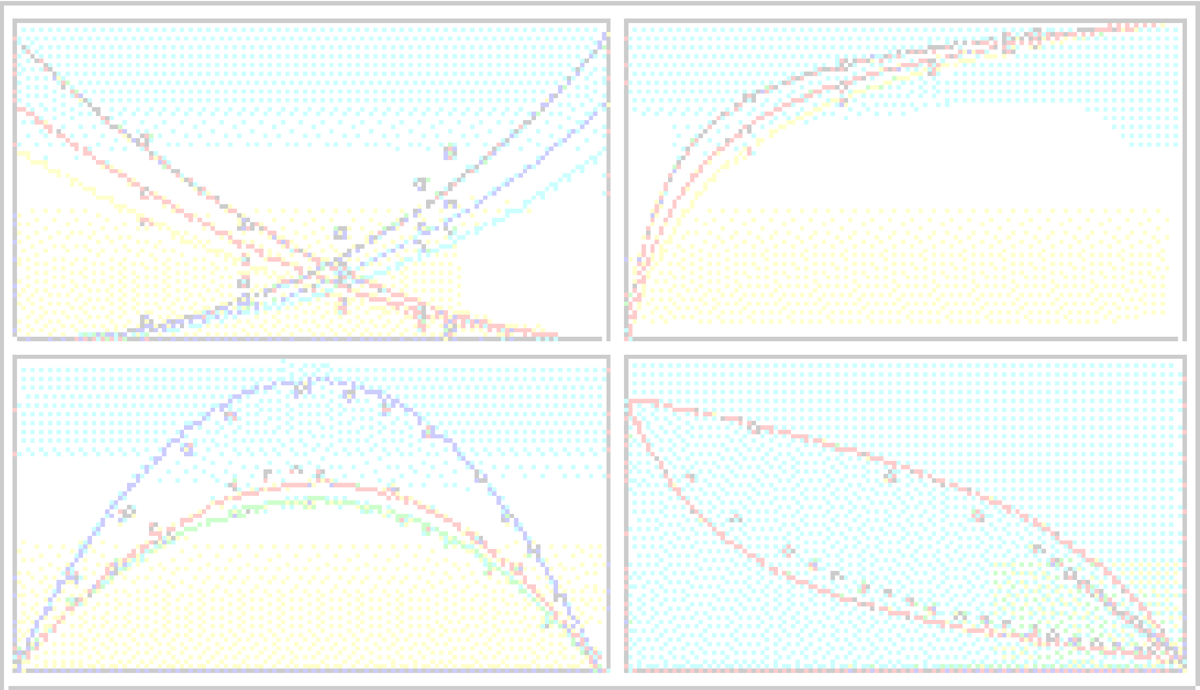
COSMO*therm* Reference Manual

Version C3.0 Release 17.01



COSMO*logic* GmbH & Co KG
Imbacher Weg 46, D-51379 Leverkusen, Germany
cosmotherm@cosmologic.de

Contents

1 Introduction

COSMO*therm* is a program that computes thermophysical data of liquids. COSMO*therm* is based on COSMO-RS theory[1,2]. A list of COSMO and COSMO-RS / COSMO*therm* related publications can be found in **Appendix A: Publications**.

1.1 How to Cite

All publications resulting from use of the COSMO*therm* program must acknowledge the following:

COSMO-RS (theory):

- Klamt, A. *J. Phys. Chem.* **99**, 2224 (1995).
- Klamt, A.; Jonas, V.; Bürger, T.; Lohrenz, J. C. *J. Phys. Chem. A* **102**, 5074 (1998).

COSMO*therm* (software)

- COSMO*therm*, Version C3.0, Release 17.01; COSMO*logic* GmbH & Co. KG, `http://www.cosmologic.de`
- Eckert, F. and A. Klamt, *AIChE Journal*, **48**, 369 (2002).

Regarding the reproducibility of COSMO*therm* results in peer-reviewed publications, we offer a limited evaluation of the program to all scientists who want to test the program and/or reproduce results published in peer-reviewed journals. Any scientist who wants to reproduce such published results is entitled to a free evaluation of the unrestricted full version of the COSMO*therm* program. Please contact our support at `cosmotherm@cosmologic.de` citing the specific article. This service aims to aid the scientific community and keep up its high standards. When publishing results that were obtained with an unrestricted full version of COSMO*therm*, you may add a corresponding note to inform the editor, reviewer, and readers of the article about this possibility.

1.2 Theory

The COSMO*therm* program is based on COSMO-RS theory of interacting molecular surface charges[1,2,3,4]. COSMO-RS is a theory of interacting molecular surfaces as computed by quantum chemical methods (QM). COSMO-RS combines an electrostatic theory of locally interacting molecular surface descriptors (which are available from QM calculations) with a statistical thermodynamics methodology.

---

[1] Eckert, F. and A. Klamt, *AIChE Journal*, **48**, 369 (2002).

[2] Klamt, A. and F. Eckert, *Fluid Phase Equilibria*, **172**, 43 (2000).

[3] Klamt, A.; V. Jonas; T. Bürger and J. C. W. Lohrenz, *J. Phys. Chem. A,* **102**, 5074 (1998).

[4] Klamt, A., *J. Phys. Chem.* **99**, 2224 (1995).

The quantum chemical basis of COSMO-RS is COSMO[5], the Conductor-like Screening Model, which belongs to the class of QM continuum solvation models (CSMs). In general, basic quantum chemical methodology describes isolated molecules at a temperature of T=0 K, allowing a realistic description only for molecules in vacuum or in the gas phase. CSMs are an extension of the basic QM methods towards the description of liquid phases. CSMs describe a molecule in solution through a quantum chemical calculation of the solute molecule with an approximate representation of the surrounding solvent as a continuum. Either by solution of the dielectric boundary condition or by solution of the Poisson-Boltzmann equation, the solute is treated as if embedded in a dielectric medium via a molecular surface or "cavity" that is constructed around the molecule. Hereby, normally the macroscopic dielectric constant of the solvent is used. COSMO is a quite popular model based on a slight approximation, which in comparison to other CSMs achieves superior efficiency and robustness of the computational methodology[5,6]. The COSMO model is available in several quantum chemistry program packages: Turbomole[7], DMOL3[8], Gaussian[9], GAMESS-US[10], PQS[11], Molpro[12], Columbus[13], ORCA[14], and Q-Chem[15]. If combined with accurate QM CSMs have been proven to produce reasonable results for properties like Henry law constants or partition coefficients. However, as has been shown elsewhere[6] the continuum description of CSMs is based on an erroneous physical concept. In addition, concepts of temperature and mixture are missing in CSMs.

COSMO-RS, the COSMO theory for "real solvents" goes far beyond simple CSMs in that it integrates concepts from quantum chemistry, dielectric continuum models, electrostatic surface interactions and statistical thermodynamics. Still, COSMO-RS is based upon the information that is evaluated by QM-COSMO calculations. Basically QM-COSMO calculations provide a discrete surface around a molecule embedded in a virtual conductor[5]. Of this surface each segment i is characterized by its area a_i and the screening charge density (SCD) σ_i on this segment which takes into account the electrostatic screening of the solute molecule by its surrounding (which in a virtual conductor is perfect screening) and the back-polarization of the solute molecule. In addition, the total energy of the ideally screened molecule E_{COSMO} is provided. Within

---

[5] Klamt, A. and G. Schüürmann, *J. Chem. Soc. Perkin Trans. II*, 799 (1993).

[6] Klamt, A., "COSMO and COSMO-RS", in *Encyclopedia of Computational Chemistry*, Schleyer, P. v. R. and L. Allinger Editors, (Wiley, New York, 1998) pages 604-615.

[7] Schäfer, A.; A. Klamt; D. Sattel; J. C. W. Lohrenz and F. Eckert, *Phys. Chem. Chem. Phys.* **2**, 2187 (2000).

[8] Andzelm, J.; C. Kölmel and A. Klamt, *J. Chem. Phys.* **103**, 9312 (1995); B. Delley, *J. Phys. Chem. A* **110**, 13632 (2006).

[9] Gaussian 09, M. J. Frisch, *et al.*, Gaussian, Inc., Pittsburgh PA, (2011).

[10] Baldridge, K. and A. Klamt, *J. Chem. Phys.*, **106**, 66622 (1997).

[11] PQS, P. Pulay, J. Baker, K. Wolinski, Paralell Quantum Solutions, Fayetteville, AR (2003).

[12] MOLPRO, a package of *ab initio* programs designed by H.-J. Werner and P. J. Knowles, version 2004.1, R. D. Amos, A. Bernhardsson, A. Berning, P. Celani, D. L. Cooper, M. J. O. Deegan, A. J. Dobbyn, F. Eckert, C. Hampel, G. Hetzer, P. J. Knowles, T. Korona, R. Lindh, A. W. Lloyd, S. J. McNicholas, F. R. Manby, W. Meyer, M. E. Mura, A. Nicklass, P. Palmieri, R. Pitzer, G. Rauhut, M. Schütz, U. Schumann, H. Stoll, A. J. Stone, R. Tarroni, T. Thorsteinsson (2004).

[13] COLUMBUS, an *ab initio* electronic structure program, release 6, by H. Lischka, R. Shepard, I. Shavitt, R. M. Pitzer, M. Dallos, Th. Müller, P. G. Szalay, F. B. Brown, R. Ahlrichs, H. J. Böhm, A. Chang, D. C. Comeau, R. Gdanitz, H. Dachsel, C. Ehrhardt, M. Ernzerhof, P. Höchtl, S. Irle, G. Kedziora, T. Kovar, V. Parasuk, M. J. M. Pepper, P. Scharf, H. Schiffer, M. Schindler, M. Schüler, M. Seth, E. A. Stahlberg, J.-G. Zhao, S. Yabushita, and Z. Zhang (2003).

[14] ORCA large scale electronic structure package, F. Neese, Max-Planck-Institut für Bioanorganische Chemie, Mülheim, Germany (2005).

[15] Q-Chem, Version 4.0, Q-Chem, Inc., 2012.

COSMO-RS theory a liquid is now considered an ensemble of closely packed ideally screened molecules. In order to achieve this close packing the system has to be compressed and thus the cavities of the molecules get slightly deformed (although the volume of the individual cavities does not change significantly). Each piece of the molecular surface is in close contact with another one. Assuming that there still is a conducting surface between the molecules, i.e. that each molecule still is enclosed by a virtual conductor, in a contact area the surface segments of both molecules have net SCDs σ and σ'. In reality there is no conductor between the surface contact areas. Thus an electrostatic interaction arises from the contact of two different SCDs. The specific interaction energy per unit area resulting from this "misfit" of SCDs is given by

$$E_{MF}(\sigma,\sigma') = a_{eff}\frac{\alpha'}{2}(\sigma+\sigma')^2 \tag{1.2-1}$$

where a_{eff} is the effective contact area between two surface segments and α' is an adjustable parameter. The basic assumption of eq. 1.1-1 which is the same as in other surface pair models like UNIQUAC[16], is that residual non-steric interactions can be described by pairs of geometrically independent surface segments. Thus, the size of the surface segments a_{eff} has to be chosen in a way that it effectively corresponds to a thermodynamically independent entity. There is no simple way to define a_{eff} from first principles and it must be considered to be an adjustable parameter. Obviously, if σ equals $-\sigma'$ the misfit energy of a surface contact will vanish. Hydrogen bonding (HB) can also be described by the two adjacent SCDs. HB donors have a strongly negative SCD whereas HB acceptors have strongly positive SCDs. Generally, a HB interaction can be expected if two sufficiently polar pieces of surface of opposite polarity are in contact. Such a behavior can be described by a functional of the form

$$E_{HB}(\sigma,\sigma') = a_{eff}c_{HB}\min\left(0;\min\left(0;\sigma_{donor}+\sigma_{HB}\right)\max\left(0;\sigma_{acceptor}-\sigma_{HB}\right)\right) \tag{1.2-2}$$

wherein c_{HB} and σ_{HB} are adjustable parameters. In addition to electrostatic misfit and HB interaction COSMO-RS also takes into account van der Waals (vdW) interactions between surface segments via

$$E_{vdW}(\sigma,\sigma') = a_{eff}\left(\tau_{vdW}+\tau'_{vdW}\right) \tag{1.2-3}$$

wherein τ_{vdW} and τ'_{vdW} are element specific adjustable parameters. The vdW energy is dependent only on the element type of the atoms that are involved in surface contact. It is spatially non-specific. E_{vdW} is an additional term to the energy of the reference state in solution. Currently nine of the vdW parameters (for elements H, C, N, O, F, S, Cl, Br and I) have been optimized. For the majority of the remaining elements reasonable guesses are available[3].

The link between the microscopic surface interaction energies and the macroscopic thermodynamic properties of a liquid is provided by statistical thermodynamics. Since in the COSMO-RS view all molecular interactions consist of local pair wise interactions of surface segments, the statistical averaging can be done in the ensemble of interacting surface pieces. Such an ensemble averaging is computationally efficient – especially in comparison to the computationally very demanding molecular dynamics or Monte Carlo approaches which require averaging over an ensemble of all possible different arrangements of all molecules in a liquid. To describe the composition of the surface segment ensemble with respect to the interactions (which depend on σ only), only the probability distribution of σ has to be known for all

---

[16] Abrams, D. S. and J. M. Prausnitz, *AIChE Journal*, **21**, 116 (1975).

compounds *i*. Such probability distributions $p_i(\sigma)$ are called "σ-profiles" . The σ-profile of the whole system/mixture $p_S(\sigma)$ is just a sum of the σ-profiles of the components *i* weighted with their mole fraction in the mixture x_i.

$$p_S(\sigma) = \sum_{i \in S} x_i p_i(\sigma) \tag{1.2-4}$$

Using $e(\sigma,\sigma') = (E_{vdW}(\sigma,\sigma') + E_{HB}(\sigma,\sigma') + E_{MF}(\sigma,\sigma'))/a_{eff}$, the chemical potential of a surface segment with SCD σ in an ensemble described by normalized distribution function $p_S(\sigma)$ is given by

$$\mu_S(\sigma) = -\frac{RT}{a_{eff}} \ln\left[\int p_S(\sigma') \exp\left(\frac{a_{eff}}{RT}\left(\mu_S(\sigma') - e(\sigma,\sigma')\right)\right) d\sigma'\right] \tag{1.2-5}$$

$\mu_S(\sigma)$ is a measure for the affinity of the system *S* to a surface of polarity σ. It is a characteristic function of each system and is called "σ-potential". Eq. 1.1-5 is an implicit equation. It must be solved iteratively. This is done in milliseconds on any PC. A detailed description and a rationale of this statistical averaging procedure are given in reference[4].

The COSMO-RS representations of molecular interactions namely the σ-profiles and σ-potentials of compounds and mixtures, respectively, contain valuable information - qualitatively as well as quantitatively. A detailed description of these features is given in sections 4.1 and 4.2.

The chemical potential (the partial Gibbs free energy) of compound *i* in system *S* is readily available from integration of the σ-potential over the surface of *i* expressed in terms of the unnormalized σ-profile $p_i(\sigma)$:

$$\mu_i^S = \mu_i^{C,S} + \int p_i(\sigma) \mu_S(\sigma) d\sigma \tag{1.2-6}$$

$\mu_i^{C,S}$ is a combinatorial contribution to the chemical potential. Starting with Version C1.2, the COSMO*therm* program includes a generic expression for the combinatorial contribution to the chemical potential. This combinatorial contribution $\mu_i^{C,S}$ results from the derivation of the combinatorial free energy expression $G_i^{C,S}$:

$$G_i^{C,S} = RT\left[(\lambda_0 + \lambda_1)\sum_i x_i \ln r_i - \lambda_1 \ln\left(\sum_i x_i r_i\right) + \lambda_2 \sum_i x_i \ln q_i - \lambda_2 \ln\left(\sum_i x_i q_i\right)\right] \tag{1.2-7}$$

The combinatorial contribution $\mu_i^{C,S}$ to the chemical potential of compound *i* is:

$$\mu_i^{C,S} = RT\left[\lambda_0 \ln r_i + \lambda_1\left(1 - \frac{r_i}{\bar{r}} + \ln\frac{r_i}{\bar{r}}\right) + \lambda_2\left(1 - \frac{q_i}{\bar{q}} + \ln\frac{q_i}{\bar{q}}\right)\right] \tag{1.2-8}$$

In eq. 1.1-8, r_i is the (dimensionless) molecular volume and q_i is the (dimensionless) molecular area of compound *i*. The total volume and area of all compounds in the mixture are defined:

$$\bar{r} = \sum_i x_i r_i \tag{1.2-9}$$

$$\bar{q} = \sum_i x_i q_i \tag{1.2-10}$$

The combinatorial contribution $\mu_i^{C,S}$ eq. 1.1-8 contains three adjustable parameters λ_0, λ_1 and λ_2. Please note, that the COSMO*therm* software allows to switch off the combinatorial contribution to the chemical potential $\mu_i^{C,S}$ – either individually for any of the given compounds or globally for all compounds ("`combi`" option, see sections 2.1 and 2.3) – with the effect that $\mu_i^{C,S}$ of eq. 1.1-8 is set to zero. This option is useful if compounds are used in COSMO*therm*, which do not have a well-defined surface area and volume such as polymers or amorphous phases. The handling of polymers and macromolecules within COSMO*therm* is explained to further detail in section 4.3.2.

Please note, that the chemical potential of eq. 1.1-6 is a pseudo-chemical potential[17], which is the standard chemical potential minus $RT \ln(x_i)$. The chemical potential μ_i^S of eq. 1.1-6 allows for the prediction of almost all thermodynamic properties of compounds or mixtures, such as activity coefficients, excess properties or partition coefficients and solubility.

In addition to the prediction of thermodynamics of liquids COSMO-RS is also able to provide a reasonable estimate of a pure compound's chemical potential in the gas phase

$$\mu_i^{Gas} = E_{Gas}^i - E_{COSMO}^i - \omega_{Ring} n_{Ring}^i + \eta_{Gas} \tag{1.2-11}$$

where E^i_{Gas} and E^i_{COSMO} are the quantum chemical total energies of the molecule in the gas phase and in the COSMO conductor respectively. The remaining contributions consist of a correction term for ring shaped molecules with n^i_{Ring} being the number of ring atoms in the molecule and ω_{Ring} an adjustable parameter as well as parameter η_{Gas} providing the link between the reference states of the system's free energy in the gas phase and in the liquid. Using equations 1.1-6 and 1.1-10 it is possible to *a priori* predict vapor pressures of pure compounds. Please note that eq. 1.1-10 is an empirical formulation. It is not part of the rigorous statistical thermodynamics approach that leads to equations 1.1-4 to 1.1-6, thus it is valid for pure compounds only.

The majority of larger and more complex compounds can be built in more than one conformation, which means that they have relevant metastable energy minima in addition to the global energy minimum. Fortunately, the conformational ambiguity can be disregarded in many cases for the calculation of chemical potentials and phase equilibria with COSMO*therm*. This is the case if the σ-profiles of the different conformations are very similar, as for bond-rotation conformations in alkane chains. In such cases the thermodynamic equilibria are unaffected by the conformational ambiguity, and the compound can be well described by its minimum energy conformation. If however, the polarity of the conformations is very different, in particular if intramolecular hydrogen bond is possible in the molecule, the free energy difference may change strongly between a polar solvent such as water and a non-polar solvent or the gas phase. In this case different molecular conformations have to be taken into account in COSMO*therm*. A compound *i* can be represented by a set of COSMO files for the conformers. Input and processing of

---

[17] Ben-Naim, A., *Solvation Thermodynamics*, Plenum Press, New York and London (1987).

conformers is decribed in sections 2.2 and 2.5. The population of a conformer *j* in solvent *S* is calculated according to the Boltzmann distribution between states of different free energy ($G_j^S = E^j_{COSMO} + \mu_j^S$), equation 1.1-11. The multiplicity w_j will be assigned to each conformer by COSMO*therm* based on molecular symmetry considerations or can be given explicitly.

$$\pi_j^S = \frac{w_j \exp\left\{ -\frac{E^j_{COSMO} + \mu_j^S}{kT} \right\}}{\sum_k \omega_k \exp\left\{ -\frac{E^k_{COSMO} + \mu_k^S}{kT} \right\}} \qquad (1.2\text{-}12)$$

If the compound *i* is a part of the system *S*, i.e. if it is present at finite dilution in *S,* the chemical potentials, μ_i^S, themselves depend on the conformational population. Therefore equation (1.1-11) has to be iterated to self-consistency, starting from an initial population guess based on $\mu_i^S = 0$. This additional self-consistency cycle on top of the COSMO-RS equation self-consistency results in a somewhat higher COSMO*therm* computation time if conformers are involved. However, starting with version C21-0111, COSMO*therm* uses a novel conformer equilibration algorithm, which significantly improves computation time of the conformer's self-consistency. This new algorithm is used by default now. The old algorithm is still available and can be toggled by a keyword (`oclp` option, see sections 2.1 and 2.3).

COSMO-RS depends on an extremely small number of adjustable parameters (the seven basic parameters of eq. (1.1-1)-(1.1-3), (1.1-8) and (1.1-10) plus nine τ_{vdW} values) some of which are physically predetermined[6]. COSMO-RS parameters are not specific of functional groups or molecule types. The parameters have to be adjusted for the QM-COSMO method that is used as a basis for the COSMO-RS calculations only. Thus the resulting parameterization is completely general and can be used to predict the properties of almost any imaginable compound mixture or system. All parameters necessary for COSMO-RS computations with the COSMO*therm* are given in the COSMO*therm* parameter files (CTDATA-files, see section 0) that are shipped with each release of COSMO*therm*.

The course of a COSMO-RS calculation is illustrated in the flow chart below. The starting point is always a QM-COSMO calculation. However, the time-consuming QM-COSMO calculations have to be done only once for each compound. The results of the QM-COSMO calculations (i.e. the charge distribution on the molecular surface) can be stored in a database. COSMO-RS then can be run from a database of stored QM-COSMO calculations. Thus COSMO-RS is well suited for the task of screening large numbers of solvents or solutes if an appropriate database of QM-COSMO calculations is available (compare section 2.5).

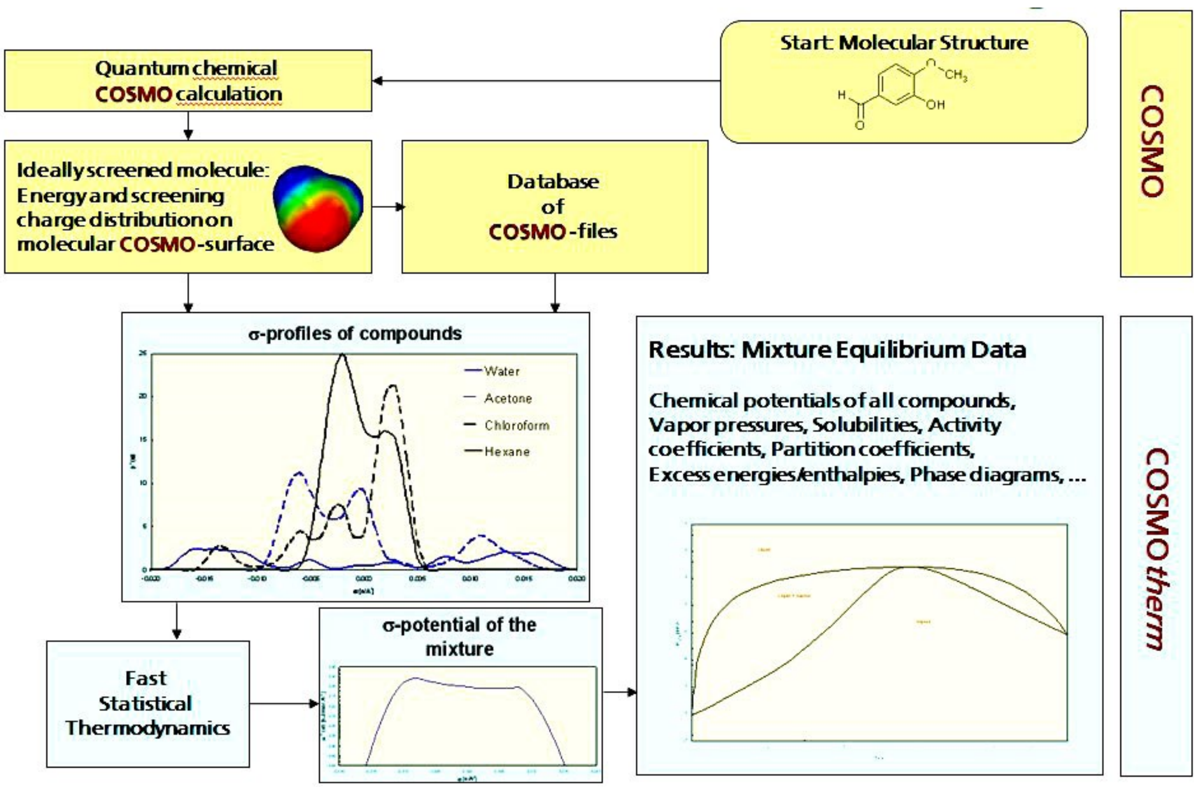


Figure 1.1-1: Flow chart of a COSMOtherm calculation.

1.3 Practical Aspects

In the current implementation, COSMO*therm* is a command-line/file driven program, which either can be used from the graphical user interface COSMO*therm*X (see COSMO*therm* User Guide for usage and details of COSMO*therm*X), or it can be run directly from a UNIX/LINUX terminal window, or in Microsoft-Windows systems from a DOS-shell (`cmd.exe`) or the Windows PowerShell (`powershell.exe`), respectively.

In a UNIX, LINUX or DOS window COSMO*therm* is invoked via

```
cosmotherm filename[.inp]
```

The file `filename[.inp]` is the COSMO*therm* input file and contains a batch of commands controlling the COSMO*therm* calculation. It is recommended to use an input file of the form `filename.inp` (i.e. with a file extension `.inp`). In this case it is possible to leave out the file extension `.inp` when invoking the COSMO*therm* program. However, any other names and file extensions are also valid. The format of and commands to be used in the input file are described in section 2.

The second file indispensable for the correct performance of the COSMO*therm* program is the COSMO*therm* parameter file. COSMO*therm* parameter files are identified by the extension **`.ctd`**. By default, a parameter file of the name `CTDATA.ctd` is read in by the program. However, it is possible to use COSMO*therm* parameter files of any other name, which can be assigned in the COSMO*therm* input file (see section 2). A detailed description of the COSMO*therm* parameter file is given in section 3, cf. also the original COSMO-RS articles[1,2,3]. Note, that previously COSMO*therm* has been introduced as COSMO-RS (i.e. the Conductor-like Screening Model for Real Solvents). However, since then the underlying physical model has been refined and largely extended (cf. reference 1 and forthcoming articles) to the representation of all kinds of thermodynamics in solution. Thus the program has been renamed COSMO*therm*.

In addition, the COSMO*therm* program requires the so-called COSMO files, which are result files from quantum chemical COSMO calculations of the compound to be processed. COSMO*therm* is extracts all the relevant information for a COSMO*therm* calculation directly from the COSMO files from quantum chemical COSMO calculations, COSMO files are identified by the extension "**`.cosmo`**" if they were computed by the quantum chemical program packages Turbomole, DMOL3, Gaussian, and others, or by the extension "**`.cos`**" if they were computed by the semi empirical program package MOPAC, see section 2.2 for details. To save disk space COSMO*therm* also allows the processing of compressed COSMO files that are identified by the extension **`.ccf`**. In such compressed COSMO files all of the information relevant to COSMO*therm* is stored in a extremely packed binary format, thus using only about 6% of the disk space of a conventional COSMO file. In practice, the handling of the compressed COSMO files is completely equivalent to the handling of conventional COSMO files. It is also possible to simply give the Chemical Abstracts / Registry Number (CAS/RN) or a short trivial name in order to identify a compound's COSMO file. In this case, an additional index file (called the COSMO database index file) which maps the CAS/RN and the trivial names to the filenames of the compound COSMO files has to be read in by COSMO*therm* (see section 2.1 for details).

By default, COSMO*therm* produces one file of output: The COSMO*therm* output file `filename.out`. Optionally, the σ-profiles of the processed compounds can be written to a file `filename.prf`, detailed information about the σ-moments and hydrogen-bonding moments can be written to a file `filename.mom` (molecular information) or `filename.moma` (molecular information divided further into atomic information) and the σ-potentials of all mixtures calculated in the COSMO*therm* job can be written to a file `filename.pot`. The σ-profiles/moments/potentials are written in a format readable by all common spreadsheet programs or by scientific graph/visualization programs like GNUplot. For the physical significance and practical use of σ-profiles, σ-moments and σ-potentials cf. Section 4.1, 4.2, and the original COSMO-RS / COSMO*therm* articles[1,2,3,4]. In the special case of the automatic calculation of phase diagrams for binary, ternary or multinary mixtures and for automatic computation of activity coefficients or partition coefficients (see below) an additional file `filename.tab` is produced, which presents the evaluated information, condensed to a single table.

1.4 Installation

COSMO*therm* release comes in one single installation form: A self-extracting **COSMO*therm*X** program installer, which includes the Java^{TM}-based **graphical user interface (GUI) COSMO*therm*X** as well as the **command line version** of **COSMO*therm***:

Windows (32bit):	`COSMOthermX-C30-1701-Installer-Win32.exe`
Windows (64bit):	`COSMOthermX-C30-1701-Installer-Win64.exe`
Linux:	`COSMOthermX-C30-1701-Installer-Linux64.sh`
Mac:	`COSMOthermX-C30-1701-Installer-Mac.sh`

The self-extracting installer files automatically install all components of COSMO*therm* and its graphical user interface COSMO*thermX*. The only additional step required is to copy the COSMO*therm* license key into the appropriate directory (see section 1.5). On first execution of the installed COSMO*therm*X you will be prompted for the license key, which will be copied into the correct location (the directory `/COSMOlogic/COSMOthermX16/licensefiles/` in your local COSMO*therm*X installation).

1.5 License

The COSMO*therm* distribution also requires a valid license key, which is necessary for the correct execution of COSMO*therm*. The license key is a text file named "**`license.txt`**". It is checked every time COSMO*therm* is executed. It is expected to be in the directory that is given by the `ldir` command in the COSMO*therm* input file (see section 2.1). Please note that the graphical user interface COSMO*thermX* on first execution will ask for the license key and copy it into the "`/licensefiles/`" directory, which is located in the installation directory `/COSMOlogic/COSMOthermX16/` in your local COSMO*therm*X installation where all current COSMOlogic products are installed. License keys already present at that location are saved and archived by COSMO*thermX*.

Please note that the COSMO*therm*X release distribution by default only holds a restricted Education/Demo license key. This means that on first execution, COSMO*therm*X will run in **restricted mode** (Education/Demo). In Education/Demo mode the program offers the full functionality with respect to property predictions, but its application is restricted to the database of compounds that comes with the release. COSMO*therm* is made fully operative by copying a valid license key into the "`/licensefiles/`" directory. The license key can be obtained from the personal user login at COSMO*logic*'s customer download site (`https://cosmologic-services.de/customer_data/`).

2 Input Structure

In this section, the commands to be included in the COSMO*therm* input file `filename.inp` are described. The syntax of the COSMO*therm* commands closely resembles the MOPAC[18] input concept: All commands are given in the form

```
command=argument
```

or

```
command={argument1 argument2 argument3}
```

i.e. if several arguments are given for a command, the arguments have to be included into curved brackets and separated by blank spaces. Note that only blanks are allowed as delimiter inside the curved brackets. The usage of commas or similar will result in an error message. There is however, no limitation in the number of blanks that are set between the commands and arguments, spare spaces are ignored. The commands do not need to be given in a special order - except for their affiliation to one of the three main areas of the input file described below. Commands are not case sensitive. Note however, that filenames given as arguments of the commands possibly will be case sensitive (for example if COSMO*therm* is run on a UNIX system). Also, note that unknown commands do not lead to an error message - they plainly will be ignored. The input of real numbers is of free format, i.e. arguments like `1.0`, `0.5d-4`, `-1.e+2`, `500` are valid. However, the latter format (input of a real number as an integer) is not recommended. The hash character '**#**' is used to identify comments in the input file. If the hash character is the first character of a line, the complete line will be ignored. In other positions, any text after the hash character will be ignored. This holds for the COSMO*therm* input file as well as for other files which hold commands that are interpreted by COSMO*therm*, such as COSMO-metafiles, vapor-pressure-files or database list files.

The COSMO*therm* input file has three main areas:

- Global command line(s) and comment line
- Compound input and options lines
- Temperature / mixture lines

---

[18] Stewart, J. J. P., MOPAC2012, Stewart Computational Chemistry, Colorado Springs, CO, USA,. `http://openmopac.net/`

2.1 Global Command and Comment Lines

The first or if necessary, the first two lines of the COSMO*therm* input file are used for global commands and general file handling commands. The second (or third) line is a comment line that is also used as a job identifier in the output file. The COSMO*therm* program automatically recognizes whether there are one or two lines of global commands, i.e. whether the comment line is the second or the third line of the input file. Note that at least one global command line is required in the input -even if it is only a 'dummy'-line. I.e. if no global commands are given, an empty line should be given as first line of the input file. Regarding the input of the directories, it is not necessary to distinguish if COSMO*therm* is run on UNIX or Windows based computers. On DOS/Windows systems COSMO*therm* can use the '/'-UNIX format for directories (for example '`C:/cosmotherm/work`') as well as the '\'-DOS format for directory input (i.e. '`C:\cosmotherm\work`').

The following (optional) commands are valid in the global command line(s):

2.1.1 File Handling

`LICENSE=name`	Optional: Give the name of the COSMO*therm* license key. The default file name of the license key is `license.txt`.
`ctd=name.ctd`	Optional: Use the file `name.ctd` as COSMO*therm* parameter file (default name: `CTDATA.ctd`).
`dbas=name.csv`	Optional: Use the file `name.csv` as COSMO database index file (default name: `DATABASE-COSMO.csv`). For purpose and description of the COSMO database index file, see section 2.5.

2.1.1 File Handling (continued)

`cdir=directory`	Optional: Sets the directory where to search for the COSMO*therm* parameter file. Default is to search in the current working directory. Alternatively, `CDIR` can be read from the UNIX or Windows environment variable `COSMOTHERM_HOME`. Note that the `cdir` command in the input file overrides the latter option. The directory name must not contain blank spaces unless it is given in quotes (e.g. `cdir="C:\Program Files\COSMOtherm\CTDATA-FILES"`).
`fdir=directory`	Optional: Sets the directory where to search for the `.cosmo`, `.cos` or `.ccf` files of the quantum chemical COSMO calculations. Default is to search in the current working directory. The directory name must not contain blank spaces unless it is given in quotes (e.g. `fdir="C:\Program Files\COSMOtherm\DATABASE-COSMO\BP-TZVP-COSMO"`).
`mdir=directory`	Optional: Sets the directory where to search for the `.cosmo` or `.ccf` files of the quantum chemical COSMO calculations that are requested from COSMO-metafiles. See section 4.3 for details. Default is to search in the current working directory or, if given, in the directory set by the `fdir` command. The directory name must not contain blank spaces unless it is given in quotes (e.g. `mdir="C:\Program Files\COSMOtherm\DATABASE-COSMO\BP-TZVP-COSMO"`).
`odir=directory`	Optional: Sets the directory for COSMO*therm* output file. Default is to use the current working directory. The `odir` option also redirects the output of all other output and table files (i.e. σ-profile/potential and `name.tab`-files). The directory name must not contain blank spaces unless it is given in quotes (e.g. `odir="C:\Program Files\COSMOtherm\Output"`).
`pdir=directory`	Optional: Sets the directory where to search for □-moment QSPR coefficient (name.prop) files. By default the CDTATA-FILES directory as given by the cdir command or as read from the UNIX or Windows environment variable COSMOTHERM_HOME is used for PDIR. Note that the pdir command in the input file overrides this default. The directory name must not contain blank spaces unless it is given in quotes (e.g. pdir="C:\Program Files\COSMOtherm\CTDATA-FILES\PROP").
`ldir=directory`	Optional: Sets the directory where to search for the COSMOtherm license keys (`license.txt`). By default the CDTATA-FILES directory as given by the cdir command or as read from the environment variable COSMOTHERM_HOME is used as LDIR. The ldir command in the input file overrides this default. The directory name must not contain blank spaces unless it is given in quotes (e.g. ldir="C:\Program Files\COSMOlogic\LICENSES").

2.1.1 File Handling (continued)

`Efile`
or
`EHfile`
or
`EJfile`
or
`ECfile`
	Optional: If this keyword is used, COSMO*therm* automatically searches for the gas phase energy files for all molecules given in the compound input section. The gas phase energy files are expected to be of the form `name.energy`, where `name` is the name from the according COSMO file (`name.cosmo`) as given in the compound input section. A description of the format of the energy-file can be found in section 2.2 (compound input) of this manual. For the commands `Efile` and `EHfile` the energy is expected in atomic units [Hartree], for `EJfile` it is expected in [kJ/mol] and for `ECfile` it is expected [kcal/mol]. By default the current working directory is searched, if the `fdir` command is used, the according path given by `fdir` is searched (cf. also the `ef[=filename]` keyword in the compound input section 2.2).

`VPfile`	Optional: If this keyword is used, COSMO*therm* automatically searches for the vapor pressure / property files for all molecules given in the compound input section. The vapor pressure / property files are expected to be of the form `name.vap`, where `name` is the name derived from the according COSMO file (`name.cosmo`) as given in the compound input section. Cf. also the `VPf[=name]` keyword in the compound input section of the COSMO*therm* input file (see section 2.2).
`Cwrl`	Optional: Create VRML files[25] of the molecular geometry of all molecules that are read in the compound input section. If this option is used, for all molecules `name.cosmo` a VRML file `name_mol.wrl` will be created. Cf. also the `Cwrl[=name.wrl]` keyword in the compound input section of the COSMO*therm* input file (see section 2.2).
`Swrl`	Optional: Create VRML files[25] of the molecular COSMO surface charges of all molecules that are read in the compound input section. If this option is used, for all molecules `name.cosmo` a VRML file `name_sig.wrl` will be created. Cf. also the `Swrl[=name.wrl]` keyword in the compound input section of the COSMO*therm* input file (see section 2.2).
`Pwrl`	Optional: Create VRML files[25] of the molecular QSPR property surface of all molecules that are read in the compound input section. If this option is used, for all molecules `name.cosmo` a VRML file `name_prop.wrl` will be created. This option is valid only if the `QSPR` or `QSPR_SI` option is given in the global command section (see section 4.5). The properties calculated by the QSPR formula for the COSMO*therm* σ-moments will be color-coded and mapped to the COSMO surface of the molecule. Cf. also the `Pwrl[=name.wrl]` keyword in the compound input section of the COSMO*therm* input file (section 2.2).
`Wrlres=res`	Optional: Set the graphical resolution of the VRML files of molecular COSMO surface charges and molecular QSPR property surfaces. If given in this section of the COSMO*therm* input file, the given resolution will be used for all VRML files of all compounds. The argument `res` is expected as a real number >0.2.

2.1.1 File Handling (continued)

`autoc[=maxautoc]`	Optional: Use all `.cosmo`, `.ccf` or `.mcos` files that are found in the directory as specified by the `fdir` command as conformers. The COSMO file filenames must follow the name convention of conformer COSMO files in COSMO*base* (i.e. conformer COSMO files are named by subsequent numbers starting with zero: `name0.cosmo`, `name1.cosmo`, … , `name_9.cosmo`, or `name_c0.cosmo`, `name_c1.cosmo`, … , `name_c9.cosmo`). By default conformer numbers ranging from 0 to 9 are considered. The optional argument `maxautoc` assigns the maximum number of conformers that are searched for, meaning that COSMOtherm will read all cosmo-files with numbers 0 to `maxautoc` that are present within the given conformation's name convention. Note that `maxautoc` has to be integer number between 1 and 999, where numbers that are larger than 9 are valid only for the `name_`**`cx`**`.cosmo` name convention. For details on conformer handling see section 2.2.4.
usec={i_1 i_2 …}	Optional, (suboption of the `autoc` command): Only use the `.cosmo`, `.ccf` or `.mcos` files with numbers i_1, i_2,… as conformers in the `autoc` option, where i_1, i_2,… are expected to be integer numbers between 0 and 9. COSMO files must follow the name convention of conformer COSMO files in COSMO*base* (i.e. conformer COSMO files are named by subsequent numbers starting with zero: `name0.cosmo`, `name1.cosmo`, … , `name9.cosmo`).
`rmic=name.mic`	Optional, (required input option for the COSMO*mic* plugin): Read micelle definition file `name.mic`. The micelle definition (file extension `.mic`) holds the micelle grid and compound definitions required for a COSMO*mic* calculation. File `name.mic` is expected to be located in the same location as the COSMO*therm* input file. Please note that this option is available only if the COSMO*mic* plugin is activated via the COSMO*therm* license key. For details and handling of the COSMO*mic* plugin, please see section 4.12 of this manual and, the COSMO*mic* documentation that is available from the COSMO*therm*X graphical user interface.
`use_tboil`	Optional for the input of boiling point temperatures: globally use the pure compound boiling points T_{Boil} as given in the compound input section or read from a compounds vapor-pressure property file as a reference point for scaling the vapor pressure prediction of the pvap option (see. section 2.3.1).
`use_pvapt`	Optional for the input of experimental pure compound vapor pressures: Use the temperature dependent experimental pure compound vapor pressure $p^0_{Exp}(T)$ as given in the compound input section or read from a compounds vapor-pressure property file as a reference point for scaling the vapor pressure prediction of the pvap option (see. section 2.3.1).

2.1.2 Print Options for the COSMO*therm* Output or Table File

`wcmn`	Optional: Write the compound/mixture number to the right side of the output file. This information can be used to efficiently process the output file, e.g. via the UNIX command `grep`.
`wconf`	Optional: If a compound input consists of several conformers (see next section), the `wconf` option toggles the output of the calculated COSMO*therm* mixture information for each individual conformer. By default, only the results for the compound are written to the output file.
`nocompw`	Optional: Do not write the pure compound information to the output file. Only notes, warnings or error messages will be printed to the compound section of the output-file.
`nomix`	Optional: Do not write the mixture information to the output file. Only notes, warnings or error messages will be printed to the mixture section of the output-file.
`long`	Optional: In the output file print all real numbers in scientific exponent number format with 15 significant digits (e.g. real number -4.8 will be printed as `-0.4800000000000000E+01`.
`pr_w`	Optional: Print complete atomic weight or real weight string to the compound section of the output file. If you toggle this option, the file line for the atomic weights may become very long.
`wtln`	Optional: Print full compound and/or molecule names to all tables in the `name.tab` table output file and the `name.mom` sigma-moments file. By default the compound/molecule names are cropped after 20 characters.
`notempty`	Optional: Print "`NA`" (Not Available) message to the `name.tab` table output file if empty table entries occur. By default an empty table entry is filled with blank spaces only.
`pzero`	Optional: Print concentrations that are zero to the table header of the `name.tab` table output file. By default only finite concentrations will be printed.
`noscreen`	Optional: Compute and print the mixture information (i.e. chemical potentials, log$_{10}$(partial pressure), total free energy in the mix, …) of all compounds for all mixtures to the mixture/property output section of the COSMO*therm* output file `name.out`. By default only the data that is actually relevant for the property computation will be printed in order to save computation time and keep the output file from getting too big.

2.1.2 Print Options for the COSMO*therm* Output or Table File (continued)

`pri1`	Optional: Print additional debug information to the output-file. Currently this option toggles the printing of the following additional information: 1) Molecular symmetry point groups. 2) Molecule principal moments of inertia. 3) Zwitterion information. 4) If additional keyword `wconf` is given, the conformer weight prefactors are printed to the mixture conformer output. 5) If the `dbas` option (use cosmo database index file) is used, the database entries for CAS-number and compound trivialname are printed to compound output.
`pri2`	Optional: Print additional debug information to the output-file (second level). Currently this option toggles the printing of the following additional information: 1) All debug information as produced by option `pri1. 2)` in combination with the **`wconf`** keyword, for each molecule the atom-wise contributions to the interaction energy terms (chemical potential μ_i, total mean interaction energy H_{tot}, misfit interaction energy H_{mf}, H-Bond interaction energy H_{HB}, and van der Waals interaction energy H_{vdW}) are printed to the conformer mixture section of the output file. 3) Molecule principal moments of inertia eigenvectors. 4) Number of irreducible representations of molecular symmetry point groups.
`pr_c`	Optional: Print concentration information in mass fractions c_i.
`pr_q`	Optional: Print concentration information in surface (area) fractions q_i.
`wcas`	Optional: Replace compound names in the output- and table-file by CAS Registry Numbers (RN). To be active, this option requires that a database index-file has been read in via the `dbas` option (use cosmo database index file). If `wcas` is used, the compounds CAS-RN associated with the given compound, as read from the database index-file will be printed to output- and table-files instead of the compound name.
`wdbn`	Optional: Replace compound names in the output- and table-file by COSMO database trivial names. To be active, this option requires that a database index-file has been read in via the `dbas` option (use cosmo database index file). If `wdbn` is used, the compounds trivial name associated with the given compound, as read from the database index-file will be printed to output- and table-files instead of the compound name.

2.1.2 Print Options for the COSMO*therm* Output or Table File (continued)

`UNIT=SI`	Optional: `UNIT=SI` in the global command section triggers the output of all values in the `.out`, `.tab`, `.mom`, `.moma`, `.prf`, `.pot` files to be in SI-units. I.e. energies in [kJ/mol], pressures in [kPa], areas and volumes in [nm^2] and [nm^3], atomic masses in [g/mol], temperatures in [K], charges in [C/mol], surface charges σ in [e/nm^2], σ-potentials μ in [kJ/mol nm^2], Antoine constants in the [kPa/K]-frame and solubility parameters in the [kJ/nm^3]-frame.
`UNIT=BRITISH`	Optional: `UNIT=BRITISH` in the global command section triggers the output of all values in the `.out`, `.tab`, `.mom`, `.moma`, `.prf`, `.pot` files to be in British units. I.e. energies in [kcal/mol], pressures in [psia] (1 psia = 6.89467 kPa) , areas and volumes in [Å^2] and [Å ^{3}], atomic masses in [g/mol], temperatures in [K], charges in [a.u.], surface charges σ in [e/ Å ^{2}], σ-potentials μ in [kcal/mol Å ^{2}], Antoine constants in the [psia/K]-frame and solubility parameters in the [kcal/ Å ^{3}]-frame.
`ctab`	Optional: Print compound and conformer mixture information to the COSMO*therm* table file. The `ctab` keyword toggles the additional tabulation of all mixture information, which by default is written to the ouput file only, to be printed to the COSMO*therm* table file. For each temperature/mixture as given in the mixture section of the input file a separate table will be created. If, in addition the `wconf` keyword is used, an additional table with the properties of the calculated COSMO*therm* mixture information for each individual conformer is written to the table file. By default all concentrations are written to the `ctab` table headers. If the the additional keyword `npzero` is used, only nonzero concentrations are written to the `ctab` table headers. To avoid an extremely large number of tables to be created, this option is only active for the plain "Temperature and Mixture" input (see section 2.3, below).
`contact`	Optional: Print statistics of the molecular surface contacts for all compounds in all mixtures to the output file. See section 0 for details.
`segment_contact`	Optional: Print statistics of the molecular surface contacts for all segments of all compounds in all mixtures to the output file to the contact statistics table file `name.contact`. See section 0 for details.
`grad`	Optional: Print the values of the temperature and composition derivatives of the chemical potentials of all compounds in all mixtures to the output file. See section 4.6 "Chemical Potential Gradients" for further information.
`pvac`	Optional: Print the value of the the chemical potential of vacuum in all mixtures to the output file.

2.1.3 General Program Control and Thresholds

`dconv=value`	Optional: Change threshold for the iterative self-consistency cycle for the determination of the chemical potential. A smaller `value` of `dconv` leads to higher accuracy of the COSMO*therm* results but also to a longer computational time due to an increasing number of iterations. Default value: `dconv=`10^{-8}.
`combi`	Optional: Switch off combinatorial contribution to the chemical potential. If given in the global command section, the `combi` command is active for the complete COSMO*therm* run (i.e. for all compounds in all of the following temperature/mixture lines).
`combi=ELBRO`	Optional: Switch on special free volume combinatorial contribution to the chemical potential. If given in the global command section, the `combi=ELBRO` command is active for the complete COSMO*therm* run (i.e. for all compounds in all of the following temperature/mixture lines). The free volume term by Elbro *et al.* is recommended for the computation of macromolecules (see section 4.3.2)
`dbco`	Optional: Use all given conformer information if using a COSMO database index file to identify compound COSMO files with the `RN=CAS` or `DBN=Trivialname` commands. For purpose and description of this command and the COSMO database index file, see section 2.5.
`nothb`	Optional: Switch off temperature dependency of the hydrogen bond contribution to the total interaction energy of the compound. If given in the global command section, the `nothb` command is active for the complete COSMO*therm* run (i.e. for all following temperature/mixture lines).
`notvdw`	Optional: Switch off temperature dependency of the van der Waals contribution to the total interaction energy of the compound. If given in the global command section, the `notvdw` command is active for the complete COSMO*therm* run (i.e. for all following temperature/mixture lines).
`nohb`	Optional: Switch off hydrogen bonding (HB) contribution to the chemical potential. If given in the global command section, the `nohb` command is active for the complete COSMO*therm* run (i.e. for all compounds in all of the following temperature/mixture lines).
`novdw`	Optional: Switch off van der Waals (vdW) interaction energy contribution to the chemical potential. If given in the global command section, the `novdw` command is active for the complete COSMO*therm* run (i.e. for all compounds in all of the following temperature/mixture lines).

2.1.3 General Program Control and Thresholds (continued)

`uqmg`	Optional for the input of external quantum chemical energies and vibrational zero point energies: Use the external quantum chemical (QM) gas phase energies and vibrational zero point energies (EZP) as given in the compound input section (`eqm` and `ezp` options) or as read from the energy file for the Boltzmann-weighting of conformers. See section 2.2.2 for details on the input and the conformer weighting procedure.
`ndgf`	Optional for the input of compound free energy of fusion: Allow negative values of ΔG_{fus} in the `DGfus` , `DGfus_SI`, `DGfus_salt`, or `DGfus_salt_SI` input. The value of ΔG_{fus} thus given can be used to compute the solubility of solid compounds or salts (see section 2.3.4).
`oclp`	Optional for the use of conformers: Use the old conformer equilibration algorithm. Starting with version C21-0111, COSMO*therm* by default uses a novel conformer equilibration algorithm, which significantly improves computation time if conformers are present. Because the resulting conformer equilibrium may differ by a few (typically <2) percent from the two algorithms, the `oclp` option offers the possibility to use the old (slow) algorithm to be able to exactly reproduce old COSMO*therm* calculations. If given in the global command section, the `oclp` command is active for the complete COSMO*therm* run (i.e. for all following temperature/mixture lines).
`Dcpfus_estimate`	Optional for the input of a temperature dependent compound free energy of fusion (via input of enthalpy or entropy of fusion ΔH_{fus} or ΔS_{fus}, and melting point T_{melt}): toggle the approximation of the heat capacity of fusion as $\Delta Cp_{fus} = \Delta S_{fus} = \Delta H_{fus}/T_{melt}$. The value of $\Delta G_{fus}(T)$ thus obtained can be used to compute the solubility of solid compounds or salts as well as solid-liquid-equilibria (SLE calculations). See sections 2.3.4 for solubility and 2.3.7 for SLE. If the `Dcpfus_estimate` keyword is given in the global input section, the approximation to ΔCp_{fus} is valid for all compounds and all mixture computations.
`E_gas_min_off`	Optional for the input of gas phase energies: switch off the usage of gas phase minimum conformer energies in single-conformer calculations (see section 2.2, gas phase energy files).

2.1.3 General Program Control (continued)

`awps`	Optional: "Allow Wrong Parameter Set", i.e. allow the usage of a parameter set that does not match the computation level of the COSMO-files used. By default such a combination would cause COSMO*therm* to stop with an error message. Please note that COSMO*therm* predictions typically will not be reasonable any more if parameterization and quantum chemical computation level do not match. Hence `awps` is an expert option that should be used with the greatest caution!
`accc`	Optional: "Allow mixture of CCF- and COSMO-files in Conformer block", i.e. allow the usage of compressed and uncompressed COSMO-files in the same conformer block. By default such a combination would cause COSMO*therm* to stop with an error message. Please note that mixing compressed and uncompressed COSMO-files can lead to inconsistent conformer energies. Hence a slight bias on the conformer distribution might be introduced into the COSMO*therm* predictions. Thus `accc` is an expert option that should be used with some caution.
`lxmx=lxmx`	Optional: increase the number of phases allowed in a multi-componen multi-phase extraction equilibrium calculation (see section 2.3.12). If given in the global input section `lxmx`, the new maximum number of phases allowed, is active for the complete COSMOtherm run (i.e. for all compounds in all of the following temperature/mixture lines). Argument `lxmx` is expected to be a integer number. Note that an increase of `lxmx` increases the memory requirements of COSMOtherm.

2.1.3 General Program Control and Thresholds (symmetry usage)

`nosym`	Optional for the use of conformers: Switch off the usage of symmetry information in the distribution of molecules among conformer sets. Details on the use of symmetry in conformer sets are given in section 2.2.4 of this manual.
`musym`	Optional: Toggle the use of absolute symmetry contributions to the chemical potential. If the global `musym` option is used, a symmetry contribution **RTln(N_{irrep})** will be added to the liquid and gas phase chemical potential of each molecule present. N_{irrep} is the symmetry number (the number of irreducible representations of the symmetry point group) of the molecule. By default N_{irrep} is determined by COSMO*therm*, but it is also possible to input a fixed symmetry number or point group in the molecule's COSMO file (see section 2.2.1), the molecule's gas phase energy file (see section 2.2.2), the compound's vapor pressure / property file, or in the compound input line (see section 2.2.3).

2.1.3 General Program Control and Thresholds (COSMO-metafile options)

`mcse`	Optional for the use of COSMO-metafiles or atomic weights: Use the sum of the fragment COSMO-file energies in the calculation of the total free energy (by default a zero value is used instead). Details on the usage of atomic weights and COSMO-metafiles are given explained in section 4.3 of this manual.
`cmet[=q]`	Optional for the use of COSMO-metafiles or atomic weights: Allow charged metafiles. If the global `cmet` option is given, all COSMO-metafiles (`.mcos`-files) and atom-weighted compounds will be “neutralized” to match the exact integer charge of the sum of the COSMO-charges of the metafile fragments (i.e. neutral metafiles will be neutralized to charge 0, singly charged cation’s metafile to charge +1, ...) or atom weighted compounds, respectively. If the global `cmet` option is given with the optional argument `q`, all given `.mcos`-files and atom weighted compounds will be “neutralized” to match the given charge `q`. The argument (neutralization charge `q`) is expected to be an integer charge number in atomic units [a.u.]. If a real number charge `q` is given, it will be rounded to the nearest integer number. By default, i.e. if no `cmet` keyword is given, all COSMO-metafiles and atom-weighted compounds will be neutralized to neutrality (i.e. `q=0` [a.u.]). Details on the usage of atomic weights and COSMO-metafiles are given in section 4.3of this manual.

2.1.4 Additional Tabulated Output Files

`Smom[=name.mom]`	Optional: Write the σ-moments of all processed compounds in tabulated form to file `name.mom`. If no argument is given, the σ-moments will be written to `filename.mom`, where `filename` is the name of the input file. In addition some other molecular information will be written to the moments file `name.mom`, including volume V, molecular weight, dielectric energy E_{diel}, average energy correction dE, van der Waals energy in continuum E_{vdw}, ring correction energy E_{ring} and the standard chemical potential of the molecule in the gas phase with respect to the ideally screened state $\mu^{QSPR}{}_{Gas} = E_{COSMO} - E_{Gas} + dE + E_{vdW} + E_{ring} - \eta_{Gas}$ (please note that $\mu^{QSPR}{}_{Gas}$ is not identical to the μ_{Gas} as computed by eq. 1.10. It is a simplified and temperature independent term intended for the use as additional descriptor of sigma-moment QSPR). For further details on sigma-moments and QSPR, please refer to section 4.5.
`Satm[=name.moma]`	Optional: Write the atomic σ-moments of all processed compounds to file `name.moma`. If no argument is given, the σ- moments will be written to `filename.moma`, where `filename` is the name of the input file. If this option is used, σ-moments will be calculated for each atom of the compounds. By default, the molecular σ-moments are written to the output-file only.
`Sprf[=name.prf]`	Optional: Write the σ-profiles of all processed compounds to file `name.prf`. If no argument is given, the σ-profiles will be written to `filename.prf`, where `filename` is the name of the input file. A summary of the σ-profiles will be written in tabulated form to the table file `filename.tab`.
`Spot[=name.pot]`	Optional: Write the σ-potentials of all calculated mixtures to file `name.pot`. If no argument is given, the σ-potentials will be written to `filename.pot`, where `filename` is the name of the input file. A summary of the σ-potential information will be written in tabulated form to the table file `filename.tab`.
`Spotc`	Optional, suboption of `spot`: In addition to the σ-potentials, also write the σ-potential coefficients of all calculated mixtures to file `name.pot`.

2.1.4.1 σ-moment QSPR Coefficient Input and Output

`QSPR=filename`
or
`QSPR={`c_1 c_2 … c_{18} `prop}`
	Optional: Read the σ-moment QSPR regression coefficients from file `filename` or directly from the input. If the coefficients are to be read from a file, the QSPR coefficient file `filename` is expected to be in the directory that also holds the COSMO*therm* parameter file, i.e. in the directory denoted by the environment variable $`COSMOTHERM_HOME` or in a directory denoted by the `cdir` command. For the format of the QSPR coefficient file, see section 4.5. If the coefficients are to be read from the input file via the `QSPR={`c_1 c_2 … c_{18} `prop}` command, the coefficients c_i are expected as real numbers separated by blank spaces. If less than 18 coefficients are given, the missing ones are assumed to be zero. The property name `prop` is expected to be a string of up to 20 characters. For a further description of the σ-moment QSPR property computation, see section 4.5.

`QSPR_SI=filename`
or
`QSPR_SI={`c_1 c_2 … c_{18} `prop}`
	Optional: Read the σ-moment QSPR regression coefficients (for σ-moments in SI-units) from file `filename` or directly from the input. Syntax: see `QSPR=filename` and `QSPR={`c_1 c_2 … c_{18} `prop}` option.

2.1.4.2 Output of molecular geometry files

`Wcar`	Optional: Write the geometries of all processed compounds `name.cosmo` to molecular structure files `name.car`. The molecular geometry will be written in (MSI) CAR-format.[19]. Cf. also the `Wcar` keyword in the compound input section of the COSMO*therm* input file (see section 2.2).
`Wml2`	Optional: Write the geometries of all processed compounds `name.cosmo` to molecular structure files `name.ml2`. The molecular geometry will be written in Tripos / Sybyl-mol2-format[20]. Cf. also the `Wml2` keyword in the compound input section of the COSMO*therm* input file (see section 2.2).
`Wsdf`	Optional: Write the geometries of all processed compounds `name.cosmo` to molecular structure files `name.sdf`. The molecular geometry will be written in MDL / ISIS-sdf-format[21]. Cf. also the `Wsdf` keyword in the compound input section of the COSMO*therm* input file (see section 2.2).
`Wmol`	Optional: Write the geometries of all processed compounds `name.cosmo` to molecular structure files `name.mol`. The molecular geometry will be written in MDL / ISIS-mol-format[22]. Cf. also the `Wmol` keyword in the compound input section of the COSMO*therm* input file (see section 2.2).
`Wxyz`	Optional: Write the geometries of all processed compounds `name.cosmo` to molecular structure files `name.xyz`. The molecular geometry will be written in Cartesian Xmol-xyz-format[23]. Cf. also the `Wxyz` keyword in the compound input section of the COSMO*therm* input file (see section 2.2).
`Wpdb`	Optional: Write the geometries of all processed compounds `name.cosmo` to molecular structure files `name.pdb`. The molecular geometry will be written in Brookhaven database pdb-format[24]. (Cartesian coordinates). Cf. also the `Wpdb` keyword in the compound input section of the COSMO*therm* input file (see section 2.2).

---

[19] MSI-car format, see Accelrys ACCL (`http://www.accelrys.com`).

[20] Tripos-mol2 format (`http://www.tripos.com/custResources/mol2Files/`).

[21] MDL / ISIS sdf-format (`http://www.mdli.com/downloads/literature/ctfile.pdf`).

[22] MDL / ISIS sdf-format (`http://www.mdli.com/downloads/literature/ctfile.pdf`).

[23] Xmol xyz-format (see e.g. `http://staff.csc.fi/~laaksone/gopenmol/help/xmol_file.html`).

[24] Brookhaven Database pdb-format. The detailed definition of the pdb model can be found on the Brookhave web site: `http://www.rcsb.org/pdb/docs/format/pdbguide2.2/guide2.2_frame.html`.

2.1.4.3 Output of COSMO files

`Wccf`	Optional: Write compressed COSMO files for all of the compounds that are read in by COSMO*therm*. The COSMO file information will be written in a compressed and encoded binary format that is only about 6% the size of the conventional COSMO files (ASCII text files). The compressed COSMO files are identified by the extension `.ccf` and can be read in by COSMO*therm* just like conventional COSMO files. Cf. also the `Wccf` keyword in the compound input section of the COSMO*therm* input file (see section 2.2).
`Wgauss`	Optional: Write converted COSMO file for all of the COSMO files computed with the Gaussian program that are read in by COSMO*therm*. The charge surface of the Gaussian COSMO files are converted into a charge surfaces that are equivalent to the charge surfaces produced by Turbomole with a molecule of the same geometry. Thus the COSMO files produced by this option can be used by COSMO*therm* in combination with a COSMO*therm* parameterization that was optimized for Turbomole (i.e. BP_TZVP or BP_SVP_AM1; see section 3.3). Cf. also the `Wgauss` keyword in the compound input section of the COSMO*therm* input file (see section 2.2).

2.2 Compound Input

The fourth and following lines of the COSMO*therm* input file are used to provide the program with the information about the compounds which shall be used in the following COSMO*therm* calculation. Each line represents the information given for one compound. I.e. all commands and options for this compound have to be given in this one line.

2.2.1 Compound and Conformer Input

The compound input lines might contain the following commands:

`f=name`
or
`f=name[.cosmo]`
or
`f=name[.cos]`
or
`f=name[.ccf]`
or
`f=name[.mcos]`
	Required: Give the filename of the results-file of the quantum chemical COSMO calculation for a molecule (`.COSMO file`, `.ccf` -file, `.cos`-file, or `.mcos`-file). The file extension `.cosmo` identifies a COSMO-file from a DMOL3, Gaussian or Turbomole calculation, whereas the extension `.cos` identifies a COSMO file from a Mopac calculation. The file extension `.ccf` identifies compressed COSMO file in binary format. The file extension `.mcos` identifies a so-called COSMO-metafile (see section 4.3). If none of these file extension are given, COSMO*therm* will try to replenish the extension and search for COSMO files or COSMO-metafiles in the following order: (1) Search for a file `name` (without extension). The file format (i.e. regular COSMO file or COSMO-metafile) of files without extension is recognized automatically. (2) Search for a file `name.cosmo` (DMOL3, Gaussian or Turbomole COSMO file). (3) Search for a file `name.cos` (Mopac COSMO file). (4) Search for a file `name.ccf` (compressed COSMO file). (5) Search for a file `name.mcos` (COSMO-metafile). Note, that the filename of the COSMO file must not contain blank spaces unless it is given in quotes (e.g. `f=`**`"name 0.cosmo"`**).

`rn=aaaaaa-bb-c`	Optional (replaces the `f=name.cosmo` command): Instead of a COSMO file filename, read in a Chemical Abstracts / Registry Number (CAS/RN). The CAS/RN will be mapped to the according filename `name.cosmo` (or `.cos` or `.ccf`) by COSMO*therm*. The use of this option requires the input of a COSMO database index file (see section 2.5).
`dbn=name`	Optional (replaces the `f=name.cosmo` command): Instead of a COSMO file filename, read in a trivial name. The trivial name "`name`" will be mapped to the according COSMO file filename `name.cosmo` (or `.cos` or `.ccf`) if the trivial name is listed somewhere in the COSMO database index file. This option is case-insensitive. The use of this option requires the input of a COSMO database index file (see section 2.5).

By default the cosmo files as requested with the `f=`, `rn=`, or, `dbn=` command are searched for in the current working directory, or in the directory given by the global or local `fdir` keyword (see section 2.1.1, and below). In addition, if the requested cosmo file is not found in the given file directory, COSMO*therm* will search for the file in the subdirectory defined by the first character of given cosmo file filename: e.g. given `fdir=”/home/DATABASE-COSMO/BP-TZVP-COSMO”` and `f=methanol.cosmo`, COSMO*therm* first will try to open the file `/home/DATABASE-COSMO/BP-TZVP-COSMO/methanol.cosmo` and, if this is not found, subsequently try to open `/home/DATABASE-COSMO/BP-TZVP-COSMO`**`/m/`**`methanol.cosmo`. This additional automatic search allows the the user to store all cosmo related files in the subdirectories defined by the first character of the filenames, to avoid having to put a large number of files into one single directory (which may cause trouble in old 32bit Linux systems).

Compound input options, continued (<u>compound list</u> and <u>file directory</u> input):

`f=`**`$DATABASE`**`=lst`
or
`rn=`**`$DATABASE`**`=lst`
or
`dbn=`**`$DATABASE`**`=lst`
	<u>Optional (replaces the `f=name.cosmo` command)</u>: Instead of a filename, read in a database list file called “`lst`“, which holds a list of COSMO filenames for the `f=$DATABASE=lst` command or a list of Chemical Abstracts / Registry Numbers (CAS/RN) for the `rn=$DATABASE=lst` command or a list trivial names for the `dbn=$DATABASE=lst` command. COSMO*therm* will then proceed with the filenames, CAS/RN or trivial names given in the database list file as if they were given in the input file. A detailed description of this option is given in section 2.5.

`fdir=directory`	Optional: Locally sets the directory where to search for the `.cosmo` , `.cos`, `.ccf` or `.mcos` files of the quantum chemical COSMO calculation for the compound in this compound input line. Given in this section of the COSMO*therm* input file, the `fdir` command is overwrites any `fdir` command given in the global input section. It is active only for the compound input line where it is given. The directory name must not contain blank spaces unless it is given in quotes (e.g. `fdir=″C:\Program Files\COSMOtherm\DATABASE-COSMO\BP-TZVP-COSMO″`).
`mdir=directory`	Optional: Locally sets the directory where to search for the `.cosmo` files of the quantum chemical COSMO calculation that are requested from a COSMO-metafile `compound.mcos` given in this compound input line. See section 4.3 for details. Default is to search in the current working directory or, if set, in the directory given by the `fdir` command. Given in this section of the COSMO*therm* input file, the `mdir` command is overwrites any `mdir` command given in the global input section. It is active only for the compound input line where it is given. The directory name must not contain blank spaces unless it is given in quotes (e.g. `mdir=″C:\Program Files\COSMOtherm\DATABASE-COSMO\BP-TZVP-COSMO″`).
`comp=name`	Optional: change name of the compound to `name` for output. By default, the filename argument of the `f=name` command is used.

Compound input options, continued (<u>automatic conformer</u> input):

`autoc[=maxautoc]`	Optional: Use all `.cosmo`, `.ccf` or `.mcos` files that are found in the directory as specified by the `fdir` command as conformers. The COSMO file filenames must follow the name convention of conformer COSMO files in COSMO*base* (i.e. conformer COSMO files are named by subsequent numbers starting with zero: `name0.cosmo`, `name1.cosmo`, … , `name_9.cosmo`, or `name_c0.cosmo`, `name_c1.cosmo`, … , `name_c9.cosmo`). By default conformer numbers ranging from 0 to 9 are considered. The optional argument `maxautoc` assigns the maximum number of conformers that are searched for, meaning that COSMOtherm will read all cosmo-files with numbers 0 to `maxautoc` that are present within the given conformation's name convention. Note that `maxautoc` has to be integer number between 1 and 999, where numbers that are larger than 9 are valid only for the `name`**`_cx`**`.cosmo` name convention. For details on conformer handling see section 2.2.4.
`usec={`i_1 i_2 `…}`	Optional, (suboption of the `autoc` command): Only use the `.cosmo`, `.ccf` or `.mcos` files with numbers i_1, i_2,… as conformers in the `autoc` option, where i_1, i_2,… are expected to be integer numbers between 0 and 9. COSMO files must follow the name convention of conformer COSMO files in COSMO*base* (i.e. conformer COSMO files are named by subsequent numbers starting with zero: `name0.cosmo`, `name1.cosmo`, … , `name9.cosmo`).

Compound input options, continued (atomic weights options):

`w={0 0 1 1 …}`	Optional: Give weights for the atoms of this compound. Weights are integer numbers `0` to `9` and are expected in the sequence of the geometry read from the `.cosmo`, `.cos` or `.ccf` file. By default, weights for all atoms are `1`. Note, in order to shorten the input for large molecules, it is possible to leave out the blanks between the weights (i.e. give `w={0011…}`). See also section 4.3.
`rw={0.1 1.d2 …}`	Optional: Give real number weights for the atoms of this compound. Weights are arbitrary real numbers divided by blank spaces and are expected in the sequence of the geometry read from the `.cosmo`, `.cos` or `.ccf` file. By default, weights for all atoms are `1`. See also section 4.3.
iw1={n_1 n_2 n_3 …}	Optional: Give weights for the atoms of this compound. For the `iw1` option by default, weights for all atoms are `0`. For atoms with the numbers n_1 n_2 n_3 … weights are set to 1. Numbers n_1 n_2 n_3 … are the atom numbers in the sequence of the geometry read from the .cosmo , .cos or .ccf file. See also section 4.3.
aw={n_1:w_1 n_2:w_2 n_3:w_3 …}	Optional: Give weights for the atoms of this compound. For the `aw` (atom numbered weights) option default weights for all atoms are `1`, or as given by the wdfl=w_0 option (see below). This default is overwritten by the weights that are given as arguments of the aw={n_1:w_1 n_2:w_2 n_3:w_3 …} option: weights have to be given in the format, n_i:w_i (i.e. separated by a colon "`:`"), where n_i is the atom number in the sequence of the geometry read from the .cosmo , .cos or .ccf file, and w_i is the atomic weight (integer or real number). All n_i:w_i entries have to be separated by blanks.
awu={n_1:w_1 n_2:w_2 n_3:w_3 …}	Optional: Give weights for the atoms of this compound. Input Syntax is the same as for the `aw` option, above. For the `aw`**u** (atom numbered weights with default unity) option default weights for all atoms are `1`.
awz={n_1:w_1 n_2:w_2 n_3:w_3 …}	Optional: Give weights for the atoms of this compound. Input Syntax is the same as for the `aw` option, above. For the `aw`**z** (atom numbered weights with default zero) option default weights for all atoms are `0`.
wdfl=w_0	Optional: Give new default value for weight strings. By default, all weight strings are assumed to be one.The wdfl= w_0 option changes this default to w_0. Argument w_0 is expected to be a positive integer or real number. See also section 4.3.

Compound input options, continued (atomic weights options):

del={q_1 q_2 ..}	Optional: Give difference charges for this compound (as real numbers q_i, separated by blanks). The difference charges q_i are expected to be in the same order as the atoms in the COSMO file of the given compound. See section 4.3 "Atomic Weights and COSMO-Metafiles" for a detailed description of the usage of difference charges. If the number of q_i thus given is lower than the number of atoms in the compound, the missing ones are assumed zero. If more q_i values are given, the superfluous ones will be ignored.
`mcse`	Optional for the use of COSMO-metafiles or atomic weights: Use the sum of the fragment COSMO file energies in the calculation of the total free energy (by default a zero value is used).
`cmet[=q]`	Optional for the use of COSMO-metafiles or atomic weights: Allow charged metafile and optionally set its "neutralization" charge. If the `cmet` option is given locally, in a compound input line, the given COSMO-metafile (`.mcos`-file) or atom-weighted compound will be "neutralized" to match the exact integer charge of the sum of the COSMO-charges of the fragments given in the `.mcos`-file (i.e. a neutral `.mcos`-file will be neutralized to charge 0, a single cation's `.mcos`-file to charge +1, ...) or by the atom weights of a given compound, respectively. If the compound `cmet` option is given with the optional argument `q`, the meta-compound defined by the `.mcos`-file or the compound's atom weights will be "neutralized" to match the given charge `q`. The argument (neutralization charge `q`) is expected to be an integer charge number in atomic units [a.u.]. If a real number charge `q` is given, it will be rounded to the nearest integer number. By default, i.e. if no cmet keyword is given, all COSMO-metafiles and atom-weighted compounds will be neutralized to neutrality (i.e. `q=0` [a.u.]).

The usage of atomic weights, COSMO-metafiles and difference charges is explained in section 4.3.

Compound input options, continued (miscellaneous options):

SMC={c_1 c_2 ..}	Optional: Give σ-moment coefficients for this compound (as real numbers c_i, separated by blanks). See section 4.5 for a detailed description of the usage of σ-moment coefficients. If less than seven (or nine, compare section 4.5) coefficients are given, the missing ones are assumed zero. If more than seven (nine) coefficients are given, the superfluous ones will be ignored.
IEI=i_1 or IEI={i_1 i_2 ..}	Optional: Give one or several interaction energy indices (IEI numbers) for this compound (as integer numbers i_i, separated by blanks). IEI numbers can be used to treat the concentration dependency if a reaction between two or more species in the mixture is taking place (e.g. the dimerization of a organic acids, the formation of a charge transfer complex between solute and solvent). See section 4.8 for a detailed description of the usage of IEI numbers.

`xw=xi`
or
`cw=ci`
	Optional: Give fixed compound concentrations for this compound. The argument `xi` and `ci` is expected to be real number between 0 and 1. For `xw=xi` the argument is assumed to be a mole fraction concentration, for `cw=ci` a mass fraction concentration. The fixed compound concentrations are multiplied with all concentrations for compound `i` given in the temperature/mixture input section of the COSMO*therm* input file. Thus, this option allows the simple handling of "mixtures of mixtures".

The usage of interaction energy indices as well as the usage of the optional input of σ-moment coefficients is explained in section 4 ''Advanced Features of COSMO*therm*''.

Control options for the visualization of molecular properties, i.e. the generation of molecular VRML-files[25]:

`Cwrl[=name.wrl]`	Optional: Create a VRML file[25] of the molecular geometry. If given in this section of the COSMO*therm* input file, the `Cwrl` command is active only for the actual compound. By default, for a molecule `molecule.cosmo` a VRML file `molecule_mol.wrl` will be created. If the command `Cwrl=name.wrl` is given with an argument, a VRML file `name.wrl` will be created.
`Swrl[=name.wrl]`	Optional: Create a VRML file[25] of the molecular COSMO surface charge. If given in this section of the COSMO*therm* input file, the `Swrl` command is active only for the actual compound. By default, for a molecule `molecule.cosmo` a VRML file `molecule_sig.wrl` will be created. If the command `Swrl=name.wrl` is given with an argument, a VRML file `name.wrl` will be created.
`Pwrl[=name.wrl]`	Optional: Create a VRML file[25] of the molecular QSPR property surface. If given in this section of the COSMO*therm* input file, the `Pwrl` command is active only for the actual compound. By default, for a molecule `molecule.cosmo` a VRML file `molecule_prop.wrl` will be created. If the command `Pwrl=name.wrl` is given with an argument, a VRML file `name.wrl` will be created. This option is valid only if the `QSPR` or `QSPR_SI` option is given in the global command section (see section 4.5). The properties calculated by the QSPR formula of the COSMO*therm* σ-moments will be color-coded and mapped to the COSMO surface of the molecule.
`Wrlres=res`	Optional: Set the graphical resolution of the VRML files of molecular COSMO surface charges and molecular QSPR property surfaces. If given in this section of the COSMO*therm* input file, the `wrlres` command is active only for the actual compound. The argument `res` is expected as a real number >0.2.

---

[25] VRML, the 'Virtual Reality Modeling Language' is a script language allowing interactively the examination of virtual three-dimensional objects (see `http://www.vrml.org`). VRML files, usually identified `name.wrl`, can be viewed with COSMO*therm*X, the graphical user interface of COSMO*therm*. Alternatively, they can be viewed within common World-Wide-Web browsers such as Mozilla Firefox or Microsoft Internet Explorer if an appropriate VRML browser plug in has been installed. Such plug-ins are available freely e.g. the Cortona VRML client by Parallel Graphics (see: `http://www.parallelgraphics.com/products/cortona`).

Control options for the visualization of molecular properties, continued:

`wrlmap=mapfile`	Optional: Create a VRML file[25] of the molecular COSMO surface property map that is given in file `mapfile`. The `wrlmap` command is active only for the actual compound and the actual property map `mapfile`. By default, for a molecule `molecule.cosmo` a VRML file `molecule_map.wrl` will be created. The property map file `mapfile` is expected to be in the same format as the surface contact statistics map file `name.contact` that can be created by COSMO*therm* via command `segment_contact`, see section 0. Alternatively, the property map file can be read in the format of an uncompressed COSMO file as computed by Turbomole. I.e. it is also possible to visualize a Turbomole COSMO file via the `wrlmap=name.cosmo` command. The `mapfile` is expected to be found in the directory where the COSMO*therm* input file is.
`namwrl=name.wrl`	Optional (Suboption of the `wrlmap=mapfile` command): Give the name of the VRML visualization file of the property map given by the `wrlmap=mapfile` command. `name.wrl` overrides the default VRML visualization file name `molec_map.wrl`.
`wrl_min=min_val`	Optional (Suboption of the `wrlmap=mapfile` command): Set the minimum value `min_val` for the property visualization of a COSMO surface property map `mapfile`. By default this value is determined automatically from the smallest value of the property given in the property column of `mapfile`. If the `wrl_min` command is used, the color management for the COSMO surface property map is adjusted according to the given `min_val`.
`wrl_max=max_val`	Optional (Suboption of the `wrlmap=mapfile` command): Set the maximum value `max_val` for the property visualization of a COSMO surface property map `mapfile`. By default this value is determined automatically from the largest value of the property given in the property column of `mapfile`. If the `wrl_max` command is used, the color management for the COSMO surface property map is adjusted according to the given `max_val`.
`map_column=`i_{col}	Optional (Suboption of the `wrlmap=mapfile` command): Choose column i_{col} of the COSMO surface property map `mapfile` that shall be visualized on the COSMO surface. By default the 6th real number column of `mapfile` is expected to hold the property that should be visualized. If property map file `mapfile` is a surface contact statistics file that was created by COSMO*therm* via command `segment_contact`, the 6th real number column holds the contact interaction statistics (see section 0). The `map_column` command allows the visualization of other properties in file `mapfile`. E.g. the surface potential on the COSMO surface that is given in the 7th column of a surface contact statistics file created by COSMO*therm* can be visualized via `map_column=7`.

Control options for the output of molecular geometry, i.e. the generation of molecular structure files:

`Wcar`	Optional: Write the geometry of the processed compound `name.cosmo` to molecular structure file `name.car`. The molecular geometry will be written in (MSI) CAR-format.. surface. If given in this section of the COSMO*therm* input file, the `Wcar` command is active only for the actual compound.
`Wml2`	Optional: Write the geometries of the processed compound `name.cosmo` to molecular structure file `name.ml2`. The molecular geometry will be written in Tripos / Sybyl-mol2-format.. If given in this section of the COSMO*therm* input file, the `Wml2` command is active only for the actual compound.
`Wsdf`	Optional: Write the geometries of the processed compound `name.cosmo` to molecular structure file `name.sdf`. The molecular geometry will be written in MDL / ISIS-sdf-format. If given in this section of the COSMO*therm* input file, the `Wsdf` command is active only for the actual compound.
`Wsdf`	Optional: Write the geometries of the processed compound `name.cosmo` to molecular structure file `name.mol`. The molecular geometry will be written in MDL / ISIS-mol-format. If given in this section of the COSMO*therm* input file, the `Wmol` command is active only for the actual compound.
`Wxyz`	Optional: Write the geometries of the processed compound `name.cosmo` to molecular structure file `name.xyz`. The molecular geometry will be written in Cartesian Xmol-xyz-format. If given in this section of the COSMO*therm* input file, the `Wxyz` command is active only for the actual compound.
`Wpdb`	Optional: Write the geometries of the processed compound `name.cosmo` to molecular structure file `name.pdb`. The molecular geometry will be written in Brookhaven database pdb-format (Cartesian coordinates). If given in this section of the COSMO*therm* input file, the `Wpdb` command is active only for the actual compound.

2.2.1.1 COSMO Files and Compressed COSMO Files (CCF Files)

Currently COSMO*therm* is able to process COSMO files created by a number of different quantum chemistry packages (see section 2.5.1). The majority of these COSMO files can be used directly, as produced from the given quantum chemistry program. However, the COSMO charge surface as produced by the Gaussian program is based on a somewhat different COSMO cavity constrcution algorithm (for more details on Gaussian COSMO files please see section 2.5.1), which requires COSMO*therm* to convert all Gaussian COSMO files to a cavity format that is compliant with the COSMO construction form generally used in COSMO*therm*. This conversion is done automatically by COSMO*therm* if a Gaussian COSMO file is detected. Because this conversion is somewhat time consuming, COSMO*therm* offers the possibility to create a COSMO file converted to the general COSMO file format created by other quamtum chemistry programs, which is read in quickly. This can be accomplished by the `Wgauss[=name.cosmo]` keyword, where the optional argument `name.cosmo` is the name of the converted COSMO file.

In addition to reading of plain text COSMO files, COSMO*therm* allows the processing (i.e. reading and writing) of Compressed COSMO files (CCF files), which are identified by file suffix **`.ccf`**. The size of CCF files is about one ninth of plain text COSMO files, thus allowing for strong savings in disc space for large databases of molecules. COSMO*therm* can create CCF files with the `Wccf[=name.ccf]` keyword, where the optional argument `name.ccf` is the name of the target CCF file.

Control options for the output of COSMO files and compressed COSMO files (`.ccf`-files):

`Wccf[=name.ccf]`	Optional: Write the compound information for the processed compound to a compressed COSMO file. The COSMO file information will be written in a compressed and encoded binary format that is only about 6% the size of the conventional COSMO file (ASCII text file). By default, for a molecule `molecule.cosmo` a compressed COSMO file `molecule.ccf` will be created. If the command `Wccf` is given with an argument `Wccf=name.ccf`, a compressed COSMO file `name.ccf` will be created. The compressed COSMO files are identified by the extension `.ccf` and can be read in by COSMO*therm* just like conventional COSMO files. If given in this section of the COSMO*therm* input file, the `Wccf` command is active only for the actual compound.
`Wgauss[=nam.cosmo]`	Optional: Write converted COSMO file for all of the COSMO files computed with the Gaussian program that are read in by COSMO*therm*. The charge surface of the Gaussian COSMO files are converted into a charge surfaces that are equivalent to the charge surfaces produced by Turbomole with a molecule of the same geometry. Thus the COSMO files produced by this option can be used by COSMO*therm* in combination with a COSMO*therm* parameterization that was optimized for Turbomole (i.e. BP_TZVP or BP_SVP_AM1; see section 3.3). By default, for a molecule `molecule.cosmo` a converted COSMO file `molecule.cosmo` will be created. If the command `Wgauss` is given with an argument `Wgauss=name.cosmo`, a converted COSMO file `name.cosmo` will be created. If given in this section of the COSMO*therm* input file, the `Wccf` command is active only for the actual compound.

COSMO and CCF files contain all of the relevant quantum chemical information that is required for a COSMO*therm* calculation of liquid thermodynamic properties. A typical COSMO file, looks like this:

```
$info
current prog.: ridft;A matrix vers.:1.0;cav. vers.: 1.0;ridft;b-p;def-TZVP;
$cosmo
  epsilon=infinity
  nppa= 1082
  nspa=   92
  disex= 10.0000
  rsolv= 1.30
  routf= 0.85
  cavity closed
  amat file=amat.cosmo
  phsran=  0.0
  ampran= 0.10E-04
$cosmo_data
  fepsi=     1.0000000
  disex2= 3538.50
  nsph=   32
  nps=  136
  npsd=  228
  npspher=   92
  area=  153.77
  volume=  172.63
$coord_rad
#atom   x
   1   0.00000070500000   0.00000000000000   0.12717137800000  o      1.72000
   2  -1.44387771300000   0.00000000000000  -1.00928443900000  h      1.30000
   3   1.44386652900000   0.00000000000000  -1.00929324200000  h      1.30000
 $screening_charge
  cosmo      =  -0.012199
  correction =   0.011725
  total      =  -0.000474
$cosmo_energy
  Total energy [a.u.]            =      -76.4781152239
  Total energy + OC corr. [a.u.] =      -76.4785388965
  Total energy corrected  [a.u.] =      -76.4783270602  Note: incorrect value contained for downward compatibility
  Dielectric energy [a.u.]       =       -0.0148636263
  Diel. energy + OC corr. [a.u.] =       -0.0152872990
$segment_information
    1    1    2.433277990   -0.061540093    2.281238018    0.002737924    0.309230275    0.008853997   -0.086607271
    2    1   -0.030670543   -2.199288003    2.520246977    0.007124829    0.343589194    0.020736475   -0.147729007
    3    1   -0.527417415    1.061508920    3.153652997    0.006402601    0.343589194    0.018634465   -0.149873711
…
```

The main information read by COSMO*therm* are the `$info`, `$cosmo`, and `$cosmo_data` sections, which denote the quantum chemical level, basis set and COSMO cavity construction algorithm the COSMO file was created with, the `$coord` section, which holds the molecular geometry, the `$cosmo_energy` section, which holds the quantum chemical energy, and the `$segment_information` section, which holds the molecules sigma surface.

In addition to these mandatory fields it is possible to pass over some accessory compound information in a COSMO file in the `$cosmo_info` section of the COSMO file. The `$cosmo_info` section may include the CAS registry number, the sum formula, or the molecular weight of the given molecule. Moreover, the `$cosmo_info` section can also be used to provide COSMO*therm* with information specific to the given molecular conformation or specific to the COSMO file of the molecule. Please note that all of entries in the `$cosmo_info` field of the COSMO file are fixed format and case sensitive. I.e. they have to given in the COSMO file in the exact formats described in the list below.

Currently the following COSMO file information fields can be processed and used by COSMO*therm*:

- `Single point energy [a.u.]      = E_GAS(SP)`

  Input of a gas phase single point energy `E_GAS(SP)`, which will be used as fallback value if no gas phase energy file (see section 2.2.2) is read in the COSMO*therm* input.

- `N**2_EST=n`^{2}

  Input of the molecules squared refraction index n^2. This input will overwrite the default refraction index estimate computed by COSMO*therm*, but it will be overwritten by the refraction index input from a vapor pressure / property file or from a compound line input (see section 2.2.3).

- `EPSILON0_EST=`ε

  Input of the molecule's dielectric constant ε. This input will overwrite the default dielectric constant estimate computed by COSMO*therm*, but it will be overwritten by the dielectric constant input from a vapor pressure / property file or from a compound line input (see section 2.2.3).

- `SYMMETRY=`N_{irrep}
  or
  `nirrep=`N_{irrep}

  Input of a fixed symmetry number N_{irrep} (number of irreducible representations) to be used in conformer equilibrium computations, or as addition to the chemical potential of a compound via global option `musym` (see section 2.1.3), replacing the symmetry computed from the COSMO file geometry. Note that for the `SYMMETRY=`N_{irrep} and the `nirrep=`N_{irrep} keywords no blanks are allowed between = and N_{irrep}.

- `pgroup=` Π

  Input of a fixed point group Π (where Π is the point group symbol, e.g. Π = `cs`, `c2v`, `D4h`, …) to be used in conformer equilibrium computations, or as addition to the chemical potential of a compound via global option `musym` (see section 2.1.3), replacing the symmetry point group computed from the COSMO file geometry. Note that for the `pgroup` keyword no blank spaces are allowed between = and Π. The given point group symbol Π will be used to determine the fixed symmetry number (number of irreducible representations) N_{irrep} of the molecule and replace the N_{irrep} number computed by COSMO*therm*.

2.2.2 Gas Phase Energy Input

To be able to compute thermodynamic properties that are related to the gas phase (i.e. vapor pressure, heat of vaporization, Gibbs free energy of solvation) COSMO*therm* requires the free energy of the gaseous phase. This can either be approximated, or preferably, computed on the basis of the quantum chemical energy of molecule in the gas phase (i.e. the the quantum chemical energy of the molecule optimized on the same level as the COSMO file but without the COSMO model). COSMO*therm* is able to process such a quantum chemical gas phase energy E_i^{Gas} via the input of gas phase energy files denoted by file suffix **`.energy`**. COSMO*therm* allows several different procedures of reading gas phase energy files.

Compound input options, continued (gas phase energy input):

`e=energy`
or
`eH=energy`
or
`eJ=energy`
or
`eC=energy`
	Optional: Give gas phase energy for this compound. For keywords `e=energy` and `eH=energy` the energy is expected in atomic units [Hartree]; for `eJ=energy` and `eC=energy` it is expected in [kJ] and [kcal].

`ef[=filename]`
or
`efH[=filename]`
or
`efJ[=filename]`
or
`efC[=filename]`
	Optional: Read gas phase energy for this compound from file `filename`. If no argument is given (i.e. just the `ef`, `efH`, `efJ` or `efC` keyword), the name of the gas phase energy file is expected to be the name of the according COSMO file with the file extension `.energy` (i.e. for COSMO file `name.cosmo`, a gas phase energy file `name.energy` is expected). In the gas phase energy file, the energy should be given as a single number (free format), without any further text or other control characters. For keywords `ef` and `efH` the energy is expected in atomic units [Hartree]; for `efJ` and `efC` it is expected in [kJ/mol] or [kcal/mol]. COSMO*therm* will search the actual working directory or if specified, the directory given by the `fdir` command for the gas phase energy files. Note, that it is also possible to automatically search for the gas phase energy files for all of the molecules given in the compound input section via the global command `Efile`) (or its variants `EHfile`, `EJfile` or `ECfile`; see section 2.1. The filename of the energy-file must not contain blank spaces unless it is given in quotes (e.g. `ef="name 0.energy"`).

`e_gas_min_off`	Optional for the input of gas phase energies: switch off the usage of gas phase minimum conformer energies in single-conformer calculations (see below).

If the gas phase energy is read from a gas phase energy file `name.energy` the COSMO*therm* file name and file format conventions have to be met. The energy files have to follow the name of the cosmo file they are associated with (e.g. `ethanol0.energy` is appendant to `ethanol0.cosmo`). The energy file itself can be provided in two alternative formats:

- Plain energy format: The gas phase energy file is expected to contain the gas phase energy as a single number in the first line of the file. E.g. `ethanol0.energy` holds:

```
-155.10644832767
```

- XYZ-geometry-file format: The gas phase energy file is expected to contain the gas phase energy as well as the compounds gas phase geometry in Cartesian xyz (xmol) format, which consists of the number of atoms `n` in the first line, a comment line as second line followed by `n` atom coordinate lines, which hold the atom elements followed by their Cartesian x, y, and z coordinates in Ångstrom. The second line, which is a info/comment line in the xmol xyz format, is used to pass over the gas phase energy information to COSMO*therm*. It is organized the following way: the info line can contain several information fields separated by colons "`;`". There is one required field holding the gas phase energy of the compound, which is identified by the keyword "**`ENERGY=`**". Several other information fields and optional keywords may follow. E.g. `ethanol0.energy` from the COSMO*therm* release database holds:

```
9
ENERGY=-155.10644832767;METHOD=b-p;BASIS=def-TZVP;
 C     -0.015981329     -1.282788176     -0.256363746
 H      0.848236555     -1.354251576     -0.936194247
 C      0.032393182      0.002051837      0.563717745
 H     -0.932167610     -1.323654004     -0.862141107
 H      0.007353355     -2.167141078      0.399090004
 H     -0.827867695      0.054160155      1.246142506
 H      0.946500407      0.021482131      1.187552880
 O     -0.058541905      1.187809328     -0.241635127
 H      0.691607816      1.176916222     -0.861486484
```

Therein, the `ENERGY=-155.1064483276` is the required gas phase energy in Hartree. The following two fields, `METHOD` and `BASIS`, are information lines created by Turbomole. They can be used to identify and check the method and basis set that were applied in the quantum chemical calculation that created the energy file (it should be the same method and basis set that were used to create the associated cosmo file). There are several additional keywords and information fields that can be stored in the xyz format gas phase energy file. The info/comment line of the xyz format may hold the gas phase energy conformer minimum energy (**`E_GAS_MIN`** option), external quantum chemical energies and zero point vibrational energies (**`EQM`** and **`EZP`** options), and symmetry information in terms of the molecules point group or number of irreducible representations (**`PGROUP`**, **`SYMMETRY`**, and **`NIRREP`** options).Please note that the keywords in the xyz info/comment line are case-sensitive. The keywords have to be separated and terminated by semicolons "`;`". No blanks are allowed between the equal sign = and the argument of the keyword.

Currently the following information fields can be read from the info/comment line of a xyz-formatted gas phase energy file:

- `E_GAS_MIN=E_GAS(min)`

  Input of the gas phase energy conformer minimum energy `E_GAS(min)`. Gas phase energy conformer minimum energy (**`E_GAS_MIN`** option). If present, the `E_GAS_MIN` information will be used in single-conformer computations of conformers with a gas phase energy that is higher than the minimum gas phase energy, thus replacing the energy read from the `ENERGY=` field. This default can be disabled by the global or compound line input option `E_GAS_MIN_OFF`.

- `EQM=`E_{QM}
  and
  `EZP=`E_{ZP}

  Input of external quantum chemical energy and zero point vibrational energy. See below for more details on the **`EQM`** and **`EZP`** options.

- `SYMMETRY=`N_{irrep}
  or
  `NIRREP=`N_{irrep}

  Input of a fixed symmetry number N_{irrep} (number of irreducible representations) to be used in conformer equilibrium computations, or as addition to the chemical potential of a compound via global option `musym` (see section 2.1.3), replacing the symmetry computed from the COSMO file geometry.

- `PGROUP=` Π

  Input of a fixed point group Π (where Π is the point group symbol, e.g. Π = `cs`, `c2v`, `D4h`, …) to be used in conformer equilibrium computations, or as addition to the chemical potential of a compound via global option `musym` (see section 2.1.3), replacing the symmetry point group computed from the COSMO file geometry. The given point group symbol Π will be used to determine the fixed symmetry number (number of irreducible representations) N_{irrep} of the molecule and replace the N_{irrep} number computed by COSMO*therm*.

Using the xyz format, the COSMO*therm* energy file allows the storage of further quantum chemistry calculation results data. Currently this information is not processed by COSMO*therm* or COSMO*therm*X. Accessory data of this kind (such as charges, vibrational frequencies, or gas phase dipole moments) may be added below the xyz geometry coordinate block, separated by one empty line:

```
9
ENERGY=…;METHOD=b-p;BASIS=def-TZVP;EQMH=…;EZP=…;E_GAS_MIN=…;SYMMETRY=…;
O  -0.454248850   1.237812268   0.000098685
C  -0.044033599   0.080935387  -0.000377161
C   1.441015399  -0.260196871   0.000034548
N  -0.893703277  -0.997075240  -0.000415213
H  -0.558769498  -1.952457654  -0.000524451
H  -1.894542884  -0.827226585   0.000850507
H   1.653497874  -1.337692988  -0.011012317
H   1.903200971   0.184834546   0.890965416
H   1.908222659   0.204516002  -0.877963952

$charge     # formal charge of the molecule in [a.u.]
0
$s2         # S2 expectation value
0
$dipole     # dipole moment in [a.u.]: abs x y z
1.5163  0.1957 -1.5036 0.00127
$symmetry   # molecular symmetry (number of irreducible representations)
1
$zpe        # zero point vibrational energy in [a.u.]
0.0711859
$vibration  # vibrational frequencies in [1/cm], 10 entries per line
17.50 220.80 417.34 510.94 526.72 651.44 815.25 945.71 1017.71 1081.92
1299.67 1350.57 1428.44 1445.31 1570.72 1712.74 2983.25 3060.46 3063.24 3495.12
3632.20
$end        # end of data blocks
```

The entries in this optional accessory data block have to be separated by `$keyword` entries which are denoted by a Dollar sign as first character. The line(s) below the `$keyword` entry are intended to hold the actual data. It is possible to give several data blocks in a row. The final data block has to be terminated by the `$end` entry. The order of the data blocks is arbitrary. It is expected that all data given in the data blocks is on the quantum chemical level as defined by `METHOD` and `BASIS` fields as defined above. It is possible to give comments in the `$keyword` lines, equivalently to the COSMO*therm* input, using a "`#`" character. Note that comments are allowed in the data block part of the energy file only. Please also note that the data blocks are intended to hold accessory information, which can not be edited by COSMO*therm* and COSMO*therm*X. In COSMO*therm*, the entries for fixed symmetry number (**`$symmetry`**) and zero point vibrational energy (**`$zpe`**) of the molecule in the gas phase are read with lower priority than the entries in the second line (the .xyz comment line). E.g. if no `EZP=value` entry for zero point vibrational energy is given in the the .xyz comment line, then COSMO*therm* will look for a `$zpe` entry in the accessory data block and, if present, read the zpe value (given in Hartree atomic units) from there. Same holds for the entry of a fixed symmetry number. Any other entries in the accessory data blocks will be ignored by COSMO*therm*.

In addition to regular gas phase energies E_i^{Gas} COSMO*therm* can also process external gas phase quantum chemical energies E_i^{QM} and vibrational zero-point energies E_i^{ZPE}. These two properties are important in describing reactions (see section 4.4). It is also possible to use such external energies to improve the relative conformer distribution (see below). The external E_i^{QM} and E_i^{ZPE}energies either can be read from the compound input lines of the COSMO*therm* input file, from the molecules vapor-pressure and property file `molecule.vap` (if the automatic vap-file reading option `VPfile` option is triggered (see section 2.1) or if the local `VPf=filename` option is used), or from the comment line (second line) of the molecules gas phase energy file `molecule.energy` (if the automatic energy-file reading option `Efile` option is triggered (see section 2.1) or if the local `Ef=filename` option is used). If several values are given (i.e. if `eqm` and `ezp` options are given in both the compound input line and the vap-file, or the energy-file) the input line has highest priority followed by vap-file and the energy file. Thus if `eqm` is present both in the input and in the vap-file, the value from the input file will be used.

Compound input options, continued ("External" Quantum Mechanical energy input):

`eqm=E`
or
`eqmH=E`
or
`eqmC=E`
or
`eqmJ=E`
or
`eqmV=E`
	Optional: input of a molecules "external" quantum chemical (gas phase) energy E_i^{QM} . Argument `E` is expected to be a real number. For the keywords `eqm` and `eqmH` the energy is expected to be in atomic units [Hartree], for the keyword `eqmC` the energy is expected to be in [kcal/mol], for the keyword `eqmJ` the energy is expected to be in [kJ/mol], and for the keyword `eqmV` the energy is expected to be in [eV].

`ezp=EZP`
or
`ezpH=EZP`
or
`ezpC=EZP`
or
`ezpJ=EZP`
or
`ezpV=EZP`
	Optional: input of a molecules vibrational zero point energy E_i^{ZPE}. Argument `EZP` is expected to be a real number. For the keywords `ezp` and `ezpH` the energy is expected to be in atomic units [Hartree], for the keyword `ezpC` the energy is expected to be in [kcal/mol], for the keyword `ezpJ` the energy is expected to be in [kJ/mol], and for the keyword `ezpV` the energy is expected to be in [eV].

The external gas phase molecular quantum chemical gas phase energies E_i^{QM} and vibrational zero point energies E_i^{ZPE} can be used for two purposes: 1) computation of a *reaction* equilibrium (see section 4.4 for details), and 2) *Boltzmann-weighting of conformers* with the given external molecular $E_i^{QM} + E_i^{ZPE}$ gas phase energies, overriding the default use of E_i^{COSMO} in conformer equilibrium. This special conformer weighting tries to (at least partly) account for some of shortcomings of the DFT methods that were used to create the COSMO and gas phase energy calculation as they are applied in a COSMO*therm* calculation. The DFT methods used are well suited to compute the charge surfaces required by COSMO*therm*, but they are less well in computing absolute energies. Thus the E_i^{COSMO} used to compute the conformer equilibrium in the liquid, and the E_i^{Gas} used to compute the conformer equilibrium in the gas phase typically are somewhat flawed. The error caused by using DFT energies should be quite small and negligible in most cases and for most applications. Nevertheless in some special cases the overall predictions can be improved if a better QM method is used to compute the energies used to weight the conformers. This special conformer weighting is activated with the global or mixture section keyword **uqmg** "Use external QM Gas phase energies" keyword. The `uqmg` option replaces the regular E_i^{COSMO} term in the Boltzmann-equation for weighting conformers (equation 1.2-12 in section 1.2) with a COSMO-energy $E_i^{COSMO,corr}$ corrected by the external QM gas phase energy and EZP:

$$E_i^{COSMO,corr} = \left(E_i^{QM} + E_i^{ZPE}\right) + \left(E_i^{COSMO} - E_i^{Gas}\right) \tag{2.2.2-1}$$

Accordingly, the regular gas phase energy term E_i^{Gas} in the Boltzmann-equation for weighting conformers in the gas phase is replaced by a corrected COSMO-energy $E_i^{COSMO,corr}$ that is corrected by the external QM gas phase energy and EZP:

$$E_i^{Gas,corr} = \left(E_i^{QM} + E_i^{ZPE}\right) \tag{2.2.2-2}$$

Applying the `uqmg` weighting of conformers with given external QM energies and EZP has some important prerequisites and conditions that have to be met before starting such a calculation:

- The external QM energy E_i^{QM} needs to be a gas phase energy and E_i^{ZPE} needs to be created in a gas phase calculation.
- The conformer sets of the COSMO-files, the energy-files, and the external QM energies and EZP are required to match with each other. This means that the external QM/EZP energy and the COSMO-file, and energy-file of a given filename (e.g. `conformer_c0.cosmo`, `conformer_c0.energy`) need to belong to the "same" conformer. This means that they need to belong to equivalent minimum energy geometries on the potential energy surfaces of the different QM levels (COSMO, Gas and external QM/EZP). If it is not possible to create matching geometries on all levels (e.g. if a geometry optimization on gas phase level leads to a different conformer, or if the external QM level does not allow for geometry optimization) it is recommended to use single-point energies on the same geometry as the COSMO or energy-file.
- It is required to have each energy E_i^{Gas}, E_i^{QM}, and E_i^{ZPE} for the full set of conformers. If e.g. the external QM energy is missing for one or more of the conformers the whole set can not be used. COSMO*therm* will stop with an error message in this case.
- It is possible to leave out any combination of gas phase energies E_i^{Gas}, E_i^{QM}, and EZP E_i^{ZPE}. If no gas phase energy $E_i^{Gas}$is given, COSMO*therm* will use an estimate. If the EZP $E_i^{ZPE}$is given but no external QM energy E_i^{QM}, COSMO*therm* will use E_i^{Gas} as estimate. If the external QM energy E_i^{QM} is given but no EZP E_i^{ZPE}, COSMO*therm* will assume that $E_i^{ZPE} = 0$.

2.2.3 Vapor Pressure / Property Input

Several automatic computation options of COSMO*therm* utilize experimental pure compound data. E.g. `binary`, `ternary` or `multinary` computations (see section 2.3.7) are able to use pure compound vapor pressures, the `solub` option for the solubility prediction of solid compounds can process experimental free energy of fusion data, and the dissociation correction to the partition coefficient calculation option `logp` utilizes experimental dissociation constants. COSMO*therm* allows several ways of reading in such compound specific experimental data. The data either can be given in the **compound input line** in the compound section of the COSMO*therm* input file, or alternative theycan be read from a **vapor pressure / property file `filename.vap`** using either the global `VPfile` option (see section 2.1) or the local `VPf=filename` option:

`VPf[=filename]`	Optional: Read the vapor pressure / property input for this compound from file `filename`. If no argument is given (i.e. only the `VPf` keyword), the name of the vapor pressure / property file is expected to be the name of the according COSMO file with the file extension `.vap` (i.e. for COSMO file `name.cosmo`, a vapor pressure / property file `name.vap` is expected). The syntax of the vapor pressure / property file is identical to the syntax of the vapor pressure input in the compound input line (i.e. it is possible to use the `VPinp`, `Vpexp`, `VPant` and other commands as described above). COSMO*therm* will search the actual working directory or if specified, the directory given by the `fdir` command for the vapor pressure / property files. Note, that it is also possible to automatically search for the vapor pressure / property files for all of the molecules given in the compound input section via the global command `VPfile` (see section 2.1). The filename of the vap-file must not contain blank spaces unless it is given in quotes (e.g. `VPf` =**"`name 0.vap`"**).

The syntax of the pure compound data input is the same in the compound input line and in the vapor pressure / property (`.vap`) file. Please note however, that if data is given in both the compound input line and in the vapor pressure / property (`.vap`) file, then the former input will be treated with higher priority, i.e. if both vap-file and compound line input are read in, the compound line input finally will be used. If several entries of the same option, or several entries of options with the same functionality are given in the same compound input line or vap-file, then only the first entry in the line will be used. All further entries in the same line or vap-file will be ignored. If one pure compound data category (such vapor pressure, or free energy of fusion) allows different possibilities of entering the property (e.g. input of Wagner, Antoine, or extended Antoine coefficients for pure compound vapor pressures) the the usage of the input is prioritized by an input hierarchy, which is given together with the input options, below. Options, which allow to input single fixed-value data (such as the `VPinp` or the `DGfus` options described in the following) have higher input priority than assembled propertied that result from the combination of several parameters (such as the `VPwag` options) or from a combination of keywords (such as free energy of fusion input via `DHfus` and `TMelt` options). The specific pure compound data input options are listed below.

The automatic computation of phase diagrams with COSMO*therm* (using the `binary`, `ternary` or `multinary`-keyword, see section 2.3.7) requires the knowledge of the pure compounds vapor pressures. These can either be estimated by COSMO*therm* or given in the COSMO*therm* input file. Vapor pressures are compound specific properties, thus they have to be given in the compound section of the input file. COSMO*therm* allows several ways of reading in a compounds vapor pressure.

Pure compound vapor pressure input options:

`VPinp=pressure`
or
`VPinp_Pa=pressure`
or
`VPinp_kPa=pressure`
or
`VPinp_psia=pressure`
	Optional: Give the fixed-value vapor pressure of this compound. It is expected in [mBar] for the `VPinp` command, in [Pa] for the `VPinp_Pa` command, in [kPa] for the `VPinp_kPa` command and in [psia] for the `VPinp_psia` command, respectively.

`VPwag={A B C D E F}`
or
`VPwag_Pa={A B C D E F}`
or
`VPwag_kPa={A B C D E F}`
	Optional: Give the coefficients of the **Wagner equation** $\ln(p_i^0) = \ln(A) + 1/(1-\tau)(C\tau + D\tau^{1.5} + E\tau^3 + F\tau^6)$ (wherein $\tau = 1 - T/B$) to be used in the calculation of the vapor pressure p_i^0 of compound *i*. Coefficients *A* and *B* correspond to the compounds critical pressure p_c and critical temperature T_c, respectively. Wagner equation coefficients for many substances are tabulated in databases such as KDB[26]. Note that the coefficients *B*, *C*, *D*, *E* and *F* are expected for temperatures *T* in [K]. Coefficient *A* is defined as vapor pressure P_c in [mbar] (or in [Pa] for the `VPwag_Pa` or in [kPa] for the `VPwag_kPa` keywords, respectively).

---

[26] Kang, J. W.; K. -P. Yoo; H. Y. Kim; H. Lee; D. R. Yang and C. S. Lee, Korea Thermophysical Properties Databank (KDB), Department of Chemical Engineering, Korea University, Seoul, Korea (2000).

Pure compound vapor pressure input options, continued:

`VPant={A B C}`
or
`VPant_Pa={A B C}`
or
`VPant_kPa={A B C}`
or
`VPKant={A B C}`
or
`VPKant_Pa={A B C}`
or
`VPKant_kPa={A B C}`
or
`VPTCant={A B C}`
	Optional: Give the coefficients of the **Antoine equation** $\ln(p_i^0) = A - B / (T + C)$ to be used in the calculation of the vapor pressure p_i^0 of compound *i*. Antoine coefficients for many substances are tabulated in the book of Reid *et al.*[27] Note that the coefficients `A`, `B` and `C` are expected for temperatures `T` in [°C] (or in [K] for `VPKant`, `VPKant_Pa` and `VPKant_kPa` keywords, respectively) and vapor pressures `P` in [mbar] (or in [Pa] for the `VPant_Pa` and `VPKant_Pa` keywords or in [kPa] for the `VPant_kPa` and `VPKant_kPa` keywords). For the `VPTCant` keyword the coefficients are expected to compute *log10(p)* with vapor pressures `p` in [Torr] (mmHg) and temperatures `T` in [°C].

`VPant1={A B C D E F G}`
or
`VPant1_Pa={A B C D E F G}`
or
`VPant1_kPa={A B C D E F G}`
or
`VPKant1={A B C D E F G}`
or
`VPKant1_Pa={A B C D E F G}`
or
`VPKant1_kPa={A B C D E F G}`
	Optional: Give the coefficients of the **extended Antoine equation** $\ln(p_i^0) = A + B / (T + C) + DT + E \ln(T) + F T^G$ to be used in the calculation of the vapor pressure p_i^0 of compound *i*. This extended Antoine equation is equivalent to the ANT1 equation of the IK-CAPE standard. Several other vapor pressure equations can be expressed with this equation if the constants are set appropriately, e.g. the regular Antoine equation (*D*,*E*,*F*,*G*=0), the Kirchhoff equation (*C*,*D*,*F*,*G*=0), the Riedel equation (*C*,*D*=0, *G*=6) or the DIPPR equation (C,*D*=0, *G*=-2). Note that the coefficients `A - G` are expected for temperatures `T` in [°C] (or in [K] for `VPKant1`, `VPKant1_Pa` and `VPKant1_kPa` keywords, respectively) and vapor pressures `p` in [mbar] (or in [Pa] for the `VPant1_Pa` and `VPKant1_Pa` keywords or in [kPa] for the `VPant1_kPa` and `VPKant1_kPa` keywords).

---

[27] Reid, R. C.; J. M. Prausnitz and B. E. Poling, *The properties of Gases and Liquids*, 4th ed., McGraw-Hill, New York (1997).

Pure compound vapor pressure input options, continued:

`VPexp={p1 T1 p2 T2 p3 T3}`
or
`VPexp_Pa={p1 T1 p2 T2 p3 T3}`
or
`VPexp_kPa={p1 T1 p2 T2 p3 T3}`
or
`VPKexp={p1 T1 p2 T2 p3 T3}`
or
`VPKexp_Pa={p1 T1 p2 T2 p3 T3}`
or
`VPKexp_kPa={p1 T1 p2 T2 p3 T3}`
	Optional: Give three pairs of vapor pressures / temperatures for this compound. The values are then used to calculate the Antoine equation's coefficients for this compound. Temperatures T_i are expected in [°C] (or in [K] for `VPKexp`, `VPKexp_Pa` and `VPKexp_kPa` keywords) and vapor pressures p_i in [mbar] (or in [Pa] for the `VPexp_Pa` and `VPKexp_Pa` keywords or in [kPa] for the `VPexp_kPa` and `VPKexp_kPa` keywords)

`TPVmin=temp`
or
`TPVmin_K=temp`
	Optional: Give a minimum temperature for which the approximation of the pure compound vapor pressures via the `vpant`, `vpant1`, `vpwag` or `vpexp` option is valid. If the temperature in a `binary`, `ternary` or `multinary` caclulation is lower than argument `temp`, a warning message will be printed to the output file. `temp` is expected in [°C] for the `TPVmin` option and in [K] for `TPVmin_K` option, respectively.

`TPVmax=temp`
or
`TPVmax_K=temp`
	Optional: Give a maximum temperature for which the approximation of the pure compound vapor pressures via the `vpant`, `vpant1`, `vpwag` or `vpexp` option is valid. If the temperature in a `binary`, `ternary` or `multinary` caclulation is higher than argument `temp`, a warning message will be printed to the output file. `temp` is expected in [°C] for the `TPVmin` option and in [K] for `TPVmin_K` option.

The temperature dependent pure compound vapor pressure input options given above are used by COSMO*therm* at the following hierarchy of priorities: `VPinp` > `VPwag` > `VPant1` > `VPant` = `VPexp`.

Pure compound boiling point temperature (T_{Boil}) input:

`Tboil=value`
or
`Tboil_C=value`
or
`Tboil_K=value`
	Optional: Input of a compounds experimental boiling point temperature. Argument `value` is the pure compound boiling point temperature value T_{Boil} of the given compound in [°C] for the `tboil` and `tboil_C` options and in [K] for the `tboil_K` option. The argument is expected to be a real number. The T_{Boil} value thus given can be used as a reference point for scaling the vapor pressure prediction of the `pvap` or the `binary`, `ternary`, or `multinary` phase diagram options via the `use_tboil` keyword (cf. sections 2.3.1 and 2.3.7).

`Tref=value`
or
`Tref_C=value`
or
`Tref_K=value`
or
`Tref_F=value`
	Optional: Input of a compounds experimental boiling point reference temperature. This option is valid only in combination with the input of a boiling point reference pressure (`pref` option, see below). Argument `value` is the pure compound boiling point reference temperature value T_{ref} of the given compound. It is expected to be given as real number. T_{ref} is expected to be in [°C] for the `tref` and `tref_C` options, in [K] for the `tref_K` option, and in [F] for the `tref_F` option. The T_{ref} value thus given can be used as a reference point for scaling the vapor pressure prediction of the `pvap` or the `binary`, `ternary`, or `multinary` phase diagram options via the `use_tboil` keyword (cf. section 2.3.1 and 2.3.7).

`pref=value`
or
`pref_Pa=value`
or
`pref_kPa=value`
or
`pref_bar=value`
or
`pref_psia=value`
	Optional: Input of a compounds experimental boiling point reference pressure. This option is valid only in combination with the input of a boiling point reference temperature (`tref` option, above). Argument `value` is the pure compound boiling point reference pressure value p_{ref} of the given compound. It is expected to be given as real number larger than zero. p_{ref} is expected to be in [mbar] for the `pref` option, in [Pa] for the `pref_Pa` option, in [kPa] for the `pref_KPA` option, in [bar] for the `pref_bar` option, and in [psia] for the `pref_psia` option. The p_{ref} value thus given can be used as a reference point for scaling the vapor pressure prediction of the `pvap` or the `binary`, `ternary`, or `multinary` phase diagram options via the `use_tboil` keyword (cf. section 2.3.1 and 2.3.7).

Pure compound critical point data input:

`Tcrit=value`
or
`Tcrit_C=value`
or
`Tcrit_K=value`
	Optional: Input of a compounds experimental critical point temperature. Argument `value` is the pure compound critical point temperature value T_C of the given compound in [°C] for the `Tcrit` and `Tboil_C` options and in [K] for the `Tcrit_K` option. The argument is expected to be a real number.

`Vcrit=value`
or
`Vcrit_SI=value`
	Optional: Input of a compounds experimental critical volume V_C (the volume that corresponds to the critical density at the compounds critical point). Argument `value` is expected to be a real number. It is the pure compound critical volume V_C of the given compound in [Å^3] for the `Vcrit` option and in [nm^3] for the `Vcrit_SI` option. On input V_C is a molecular volume. To convert a molar volume $\tilde{V}_C$ to it's molecular equivalent it has to be divided by Avogadro's constant N_A via $V_C = \tilde{V}_C / N_A$.

`Dcrit=value`
or
`Dcrit_SI=value`
	Optional: Input of a compounds experimental critical density ρ_C (the density of a compound at the critical point). Argument `value` is the pure compound critical point temperature value ρ_C of the given compound in [g/cm^3] for both the `Dcrit` option the `Dcrit_SI` options. The input of the critical density ρ_C is an alternative to the input of the critical volume V_C. Internally, ρ_C will be converted to V_C.

`pcrit=value`
or
`pcrit_Pa=value`
or
`pcrit_kPa=value`
or
`pcrit_bar=value`
or
`pcrit_psia=value`
	Optional: Input of a compounds experimental critical pressure V_C (the pressure of a compound at the critical point). Argument `value` is the pure compound critical pressure value p_C of the given compound. It is expected to be given as real number larger than zero, where p_C is expected to be in [mbar] for the `pcrit` option, in [Pa] for the `pcrit_Pa` option, in [kPa] for the `pcrit_kPa` option, in [bar] for the `pcrit_bar` option, and in [psia] for the `pcrit_psia` option.

Note that all pure compound critical data can also be estimated by the PCCP option, as described in section 2.3.13 of this manual. Any given input of experimental data however will be used in COSMO*therm* with higher internal priority, overriding the estimate.

Pure compound flash point (FP) input:

`flash_psat=value`
or
`flash_psat_bar=value`
or
`flash_psat_kPa=value`
or
`flash_psat_psia=value`
	Optional: Input of a compounds experimental flash point saturation pressure. Argument `value` is expected to be the flash point saturation pressure p_{FP} of the given compound in [mbar] for the `flash_psat` option, in [bar] for the `flash_psat_bar` option, in [kPa] for the `flash_psat_kPa` option, and in [psia] for the `flash_psat_psia` option, respectively. The argument is required to be a real number larger than zero.

`flash_temp_K=value`
or
`flash_temp_C=value`
	Optional: Input of a compounds experimental flash point temperature. Argument `value` is expected to be the flash point temperature T_{FP} of the given compound in [K] for the `flash_temp_K` option, and in [°C] for the `flash_temp_C` option. The argument is required to be a real number. In case of the `flash_temp_K` option it has to be larger than zero.

COSMO*therm* allows the processing of several other compound specific properties that can be read in the compound section of the COSMO*therm* input file:

Pure compounds heat of fusion/crystallization input:

`DGfus=value`
or
`DGfus_SI=value`
	Optional: Give the free enthalpy of fusion ΔG_{fus} for this compound. For the `DGfus=value` option, ΔG_{fus} is expected in [kcal/mol], for the `DGfus_SI` option, ΔG_{fus} is expected to be in [kJ/mol]. Argument `value` is expected to be a real number. The value of ΔG_{fus} thus given can be used to compute the solubility of solid compounds via the `SOLUB` or `NSOLUB` option (see section 2.3.4).

`DHfus=value`
or
`DHfus_SI=value`
	Optional: Give the enthalpy of fusion ΔH_{fus} for this compound. For the `DHfus=value` option, ΔH_{fus} is expected in [kcal/mol], for the `DHfus_SI` option, ΔH_{fus} is expected to be in [kJ/mol]. Argument `value` is expected to be a real number. The value of ΔH_{fus} thus given can be used in combination with a given melting point T_{melt} to compute the solubility of solid compounds via the `SOLUB` or `NSOLUB` option (see section 2.3.4).

`DSfus=value`
or
`DSfus_SI=value`
	Optional: Give the entropy of fusion ΔS_{fus} for this compound. For the `DSfus=value` option, ΔS_{fus} is expected in [kcal/mol K], for the `DSfus_SI` option, ΔS_{fus} is expected to be in [kJ/mol K]. Argument `value` is expected to be a real number. The value of ΔS_{fus} thus given can be used in combination with a given melting point T_{melt} to compute the solubility of solid compounds via the `SOLUB` or `NSOLUB` option (see section 2.3.4).

`Dcpfus=value`
or
`Dcpfus_SI=value`
	Optional: Give the heat capacity of fusion ΔCp_{fus} for this compound. For the `Dcpfus=value` option, ΔCp_{fus} is expected in [kcal/mol K], for the `Dcpfus_SI` option, ΔCp_{fus} is expected to be in [kJ/mol K]. Argument `value` is expected to be a real number. The value of ΔCp_{fus} thus given can be used to compute the solubility of solid compounds via the `SOLUB` or `NSOLUB` option (see section 2.3.4).

`Tmelt=temp`
or
`Tmelt_C=temp`
or
`Tmelt_K=temp`
	Optional: Give the melting temperature T_{melt} for this compound. For the `Tmelt=temp` and `Tmelt_C=temp` options, T_{melt} is expected in [°C], for the `Tmelt_K=temp` option, T_{melt} is expected in [K]. Argument `temp` is expected to be a real number. The value of T_{melt} thus given can be used in combination with a given enthalpy or entropy of fusion ΔH_{fus} or ΔS_{fus} to compute the solubility of solid compounds via the `SOLUB` or `NSOLUB` option (see section 2.3.4).

`Dcpfus_estimate`	Optional for the input of a temperature dependent compound free energy of fusion (via input of ΔH_{fus} or ΔS_{fus}, and melting point T_{melt}): toggle the approximation of the heat capacity of fusion as $\Delta Cp_{fus} = \Delta S_{fus} = \Delta H_{fus}/T_{melt}$. The value of $\Delta G_{fus}(T)$ thus obtained can be used to compute the solubility of solid compounds or salts as well as solid-liquid-equilibria (SLE calculations). See sections 2.3.4 and 2.3.7. If the `Dcpfus_estimate` keyword is given in the compound input section, the approximation to ΔCp_{fus} is valid for the given compound in all mixture computations.

Pure compound aqueous dissociation constant (pK_a) input:

`pK_acid=value`
or
`pK_base=value`
	Optional: Input of experimental aqueous dissociation constants for acidic (`pK_acid` keyword) or basic (`pK_base` keyword) solutes. Argument `value` is the aqueous pK_a(acid) (`pK_acid` option) or pK_a(base) (`pK_base` option) value of the given compound. The argument is expected to be a real number. The pK_a values thus given can be used to obtain a dissociation correction to partition coefficent calculations with the `logp` option (i.e. distribution coefficient log$D$).

Pure compound UNIQUAC parameter input options:

`UNIQUAC_RI=ri`	Optional: Give UNIQUAC volume parameter r_i for this compound. The given r_i value will be used in the fitting of `UNIQUAC2` parameters for `binary` VLE (see section 2.3.7). Argument `ri` is expected as a real number larger than zero.
`UNIQUAC_QI=qi`	Optional: Give UNIQUAC surface area parameter for this compound. The given q_i value will be used in the fitting of `UNIQUAC2` parameters for `binary` VLE (see section 2.3.7). Argument `qi` is expected as a real number larger than zero.

Compound dielectric constant (ε) and refraction index (n^2) input options:

`epsilon=ε`	Optional: Provide the dielectric constant ε of this compound. Argument ε is expected as a real number larger than zero.
`N2=n`^{2}	Optional: Provide the square of the refraction index n^2 of this compound. Argument n^2 is expected as a real number larger than zero.

Compound density / volume and polymer molecular weight input options:

`expdensity=` ρ or `expdensity_SI=` ρ or `expdensity_Brit=` ρ	Optional: Provide the experimental density ρ of this compound. The given density will be utilized to calculate the free volume as used in the polymer specific Elbro combinatorial contribution described in section 4.3.2. Argument ρ is expected as a real number larger than zero. For the `expdensity` option argument ρ is expected to be in [g/ml], for the `expdensity_SI` option in [g/cm^3], and for the `expdensity_Brit` option in [lbm/ft^3], respectively.
`expmolvol=`$\tilde{V}_i$ or `expmolvol_SI=`$\tilde{V}_i$	Optional: Provide the experimental molar volume $\tilde{V}_i$ of this compound. The given molar volume will be utilized to calculate the free volume as used in the polymer specific Elbro combinatorial contribution described in section 4.3.2. Argument $\tilde{V}_i$ is expected to be a real number larger than zero. For the `expmolvol` option argument $\tilde{V}_i$ is expected to be in [Å^3], and for the `expdensity_SI` option in [nm^3], respectively.
`freevol=`V_i^F or `freevol_SI=`V_i^F	Optional: Provide the free volume V_i^F of this compound. The given free volume will be used in the polymer specific Elbro combinatorial contribution described in section 4.3.2. Argument V_i^F is expected to be a real number larger than zero. For the `expmolvol` option argument V_i^F is expected to be in [Å^3], and for the `expdensity_SI` option in [nm^3], respectively.
`expmw=`$MW_{polymer}$	Optional: Provide the polymer molecular weight $MW_{polymer}$ of this compound. The given polymer weight will be used in the polymer specific Elbro combinatorial contribution described in section 4.3.2. Argument $MW_{polymer}$ in [g/mol] is expected as a real number larger than zero.
`ispolymer`	Optional: Identify compound as polymer. This flag will be used in the polymer specific Elbro combinatorial contribution described in section 4.3.2.

2.2.4 Conformer Input

Different conformers of one compound have to be given in a so-called "**Conformer Block**" denoted by square brackets `'['` and `']'`. All molecules found inside the square brackets are assigned as conformers of one compound. If no compound name is given (via the `comp=name` command), the name of the first molecule in the Conformer Block will be used. Note that the square brackets have to be in the same lines as the compound/molecule input lines. The conformers will be weighted internally by COSMO*therm* using their COSMO-energy and chemical potential. In addition, a conformer weight prefactor accounting for degeneracy or symmetry of conformer distributions is taken into account: the conformer weight prefactor accounts for cases where it is possible to form a conformer in several different ways. The conformer weight prefactor is determined automatically via comparison of the symmetry properties of all conformer geometries found in a conformer block. The automatic use of symmetry for determining the weight prefactor of the conformer distribution can be disabled by the global `nosym` option (see section 2.1.3). Note that for COSMO*therm* parameter sets C30-1701 and later, the symmetry prefactors to conformer distribution are switched off by default. Alternatively, it is possible to give a conformer weight factor in the input using the **`wc`**`=value` command, overriding the default derived from symmetry considerations. The gas phase energy confortmers can be weighted independently using the **`wg`**`=value` command. It is possible to use zero as argument of the `wc` or `wg` command, which means that the cosmo- or gas phase energy conformer of the compound input line where the `wc` or `wg` command is given, will not be used in the conformer equilibrium.

As an example of an explicit choice of the conformer weight prefactor the input lines for a mixture of the two compounds glycerol (composed of three conformers, with zero, one and two internal hydrogen bonds) and water are shown in input example 2.2.4-1.

`Example 2.2.4-1:`

```
ctd=BP_TZVP_C30_1701.ctd                            # 1st Global command line
wcmn wconf                                          # 2nd Global command line
!! Glycerol conformer equilibrium in Water !!       # Comment line
f = h2o.cosmo       comp=Water                      # Water input (no conformers)
f = glycerol0.cosmo comp=Glycerol [                 # Glycerol 1st conformer
f = glycerol1.cosmo wc=2                            # Glycerol 2nd conformer
f = glycerol2.cosmo               ]                 # Glycerol 3rd conformer
```

The first compound input line of the example contains the file input for the first compound `h2o`, which is renamed by the `comp=Water` command, which will be used to identify this compound in the output file. The following line identifies the start of a conformer block via the `'['` command. In addition, the `.cosmo` file for the first conformer and the name `Glycerol` is given. The next line contains the second conformer of this compound, which is weighted by a degeneracy factor of two via the `wc=2` command. The next line holds the third conformer and the "end of conformer block" identifier `']'`.

2.2.4.1 Automatic Conformer Search

As an alternative and / or supplement to the input of conformers with the conformer block concept it is possible to use all COSMO files that are found in the directory as specified by the `fdir` command as conformers with the global or local compound line command **`autoc`**. If `autoc` is given either in the global command line or in a local compound input line of the input file, COSMO*therm* automatically will search for conformer `.cosmo`, `.ccf` or `.mcos` files of a given COSMO filename in the `fdir` directory and – if found - use these COSMO files as conformers of the given compound! To be able to do this it is necessary that the COSMO filenames must follow the name convention of conformer COSMO files in COSMO*base*, i.e. for a given "base"-filename `name.cosmo`, the conformer COSMO files have to be named by the "base"-filename with subsequent numbers starting with zero:

```
name0.cosmo
name1.cosmo
…
name9.cosmo
```

In addition the `autoc` command also recognizes files for the alternative the name convention of conformer COSMO files in COSMO*base,* which claims that for a given "base"-filename `name.cosmo`, the conformer COSMO files have to be named by the "base"-filename with a "**`_c`**" conformer identifier followed by subsequent numbers starting with zero:

```
name_c0.cosmo
name_c1.cosmo
…
name_c9.cosmo
```

By default, the `autoc` command urges COSMO*therm* to use all conformers of numbers 0 to 9 plus the given "base"-filename COSMO-file if they are found in the given COSMO-file directory. If the **`autoc[=maxautoc]`** keyword is given with the optional argument `maxautoc` COSMO*therm* will use all conformers with conformer numbers between 0 and `maxautoc` plus the given "base"-filename COSMO-file. The value of the maximum number of conformers that will be searched for is **`0 < maxautoc < 10`** for the `namex.cosmo` name convention, and **`0 < maxautoc < 1000`** for the `name_cxxx.cosmo` name convention.

The number of conformers to be used automatically with the `autoc` command additionally can be restricted with the global or local compound line suboption `usec={`i_1 i_2 …`}`. If given, only the conformer COSMO files with the exact numbers i_1, i_2,… will be used as conformers in the `autoc` option (i_1, i_2,… are integer numbers between 0 and 9 or `maxautoc`).

2.3 Property Calculations

Following the input of the different compounds, COSMO*therm* expects information about the temperature and mixture ratio of these compounds in the third area of the input file. The temperature/mixture lines are processed subsequently. There is no limitation of the number of mixtures to be processed by COSMO*therm*.

2.3.0 Overview

The temperature/mixture input line can contain the following commands:

Temperature and mixture input

`tk=temp`
or
`tc=temp`
or
`tf=temp`
	Required: Temperature of the mixture in Kelvin [K] (`tk=temp`) or degrees Celsius [°C] (`tc=temp`, K = °C + 273.15) or degrees Fahrenheit [°F] (`tf=temp`, K = (°F + 459.67)/1.8). Note that the temperature input is not required if an `isobar` VLE computation is done (see section 2.3.7).

x={x_1 x_2 ..} or c={c_1 c_2 ..}	Required: mole fractions (`x={}`) or mass fraction concentrations (`c={}`) of the compounds in this mixture (as real numbers x_i or c_i). The arguments are expected as real numbers between zero and one in the same sequence of compounds as given in the second input area. If the values do not add up to one, COSMO*therm* will normalize them. If less mole fractions / concentrations than compounds are given, the missing ones will be assumed zero. If a negative number is given, the concentration for this compound will be inserted automatically using the normalization of the sum of mole fractions. Only one negative number is allowed per mixture input line. Note that the mole or mass fraction input is not required if only one compound is given in the compound input section or if the `binary`, the `ternary`, the `multinary` or one of the property calculation options `gamma`, `logp` , `solub`, `henry` or `pKa` (see below) is used.

Temperature and mixture input (continued)

`x_pure=i`
or
`nx_pure=namei`
	Optional (replaces `x={`x_1 x_2` ..}` and `c={`c_1 c_2` ..}` input): Compute properties of pure compound `i`. If the `x_pure=i` command is used, the mole fraction concentration of compound `i` (where `i` is the compound number in the range given in the compound input section) is set to 1, all other compound concentrations are set to zero. If the `nx_pure=namei` option is used, the mole fraction concentration for the compound of the name `namei` is set to 1, all other compound concentrations are set to zero. Note that the mole fraction input is not required if only one compound is given in the compound input section or if the `binary`, the `ternary`, the `multinary` or one of the property calculation options `gamma`, `logp` , `solub`, `henry` or `pKa` (see below) is used.

`x_pure=MICELLE`	Optional: Trigger a COSMO*mic* computation. Please note that this option is available only if the COSMO*mic* plugin is activated via the COSMO*therm* license key and if a micelle definition file has been read in the global input section via command `rmic=name.mic`. For details and handling of the COSMO*mic* plugin, please see section 4.12 of this manual and, the COSMO*mic* documentation that is available from the COSMO*therm*X graphical user interface.

Property calculation options (vapor pressure, boiling point, Henry law coefficient)

`pvap`	Optional: Toggle the automatic calculation of the total **vapor pressure** of the system at a given temperature and concentration. It is also possible to compute the vapor pressures for a given temperature range via the `tk2=temp` or `tc2=temp` commands (see section 2.3.1).

`pvap=pressure`
or
`pvap_SI=pressure`
	Optional: Toggle the iterative calculation of the **boiling point** temperature of a given mixture. The given pressure is expected as a real number in [mbar] for the `pvap=pressure` option and in [kPa] for `pvap_SI=pressure` option. The temperature of the mixture will be varied iteratively until the given value of `pressure` is met (see section 2.3.1).

`henry=i` or `nhenry=name`$_i$	Optional: Toggle the automatic calculation of the **Henry law coefficients** of all compounds in the `i`th compound, where `i` is the compound number in the range given in the compound input section. The `nhenry=name`$_i$ option computes the Henry law coefficients in the compound of the name `name`$_i$. By default, the Henry law coefficients *H* are calculated at infinite dilution in compound `i`. It is also possible to calculate *H* at finite concentrations using the `xh={}` or `ch={}` command (see section 2.3.3). If such a finite concentration input is used, arguments `i` or `name`$_i$ need not be given to the `henry` or `nhenry` option.

Property calculation options, continued (activity coefficient, solubility)

`gamma=i` or `ngamma=name`$_i$	Optional: Toggle the automatic calculation of the **activity coefficients** of all compounds in the `i`th compound, where `i` is the compound number in the range given in the compound input section. The `ngamma=name`$_i$ option computes the activity coefficients in the compound of the name `name`$_i$. By default, the natural logarithms of the activity coefficients ln(γ) are calculated at infinite dilution in compound `i`. It is also possible to calculate γ at finite concentrations using the `xg={}` or `cg={}` command (see section 2.3.2). If such a finite concentration input is used, arguments `i` or `name`$_i$ need not be given to the `gamma` or `ngamma` option.
`solub=i` or `nsolub=name`$_i$	Optional: Toggle the automatic calculation of the **solubility** of all compounds in the `i`th compound, where `i` is the compound number in the range given in the compound input section. The `nsolub=name`$_i$ option computes the solubilities in the compound of the name `name`$_i$. By default, the common logarithms of the mole fractions of the solutes $\log_{10}(x_i^{SOL})$ are calculated at infinite dilution in compound `i`. It is also possible to calculate x_i^{SOL} at finite concentrations using the `xs={}` or `cs={}` command (see section 2.3.4). If such a finite concentration input is used, arguments `i` or `name`$_i$ need not be given to the `solub` or `nsolub` option.

`solgas=p`
or
`solgas_Pa=p`
or
`solgas_kPa=p`
or
`solgas_bar=p`
or
`solgas_psia=p`
	Optional: Toggle the automatic calculation of the **gas solubility** of all compounds at the given partial pressure `p` in a given solvent (see section 2.3.9). If the `solgas` keyword is given, the argument `p` is expected to be a positive real number pressure in [mbar]. If the `solgas_Pa` keyword is given, the argument `p` is expected to be a positive real number pressure in [Pa]. If the `solgas_kPa` keyword is given, the argument `p` is expected to be a positive real number pressure in [kPa]. If the `solgas_bar` keyword is given, the argument `p` is expected to be a positive real number pressure in [bar]. If the `solgas_psia` keyword is given, the argument `p` is expected to be a positive real number pressure in [psia].

Property calculation options, continued (partition coefficient, dissociation constant,reaction energy)

`logp=`{i_1 i_2} or `nlogP=`{$name_1$ $name_2$}	Optional: Toggle the automatic calculation of the **partition coefficients** of all compounds between the compounds i_1 and i_2, where i_j are compound numbers in the range given in the compound input section. The `nlogp=`{$name_1$ $name_2$} option computes the partition coefficients between the compounds of the names $name_1$ and $name_2$. By default, the common logarithms of the partition coefficients $\log_{10}$(P) are calculated for pure compounds i_1 and i_2. It is also possible to calculate logP at finite concentration of the two solvent phases using the `xli={}` or `cli={}` commands (see section 2.3.5). If such a finite concentration input is used for both solvent phases the arguments i_j or $name_i$ need not be given to the `logp` or `nlogp` option.
`pKa=`{$i_{Solvent}$ $i_{Neutral}$ i_{Ion}} or `npKa=`{ $name_{Solvent}$ $name_{Neutral}$ $name_{Ion}$ }	Optional: Toggle the automatic calculation of the **p*K*** value of the acidity / basicity of a compound in solvent *S* (which normally is water). The `pKa=`{$i_{Solvent}$ $i_{Neutral}$ i_{Ion}} option computes the pK_a value from the free energy difference of the neutral compound $i_{Neutral}$ and the ionic compound i_{Ion} in solvent $i_{Solvent}$, where i_i are the compound numbers in the range given in the compound input section. The `npKa=`{$name_{Solvent}$ $name_{Neutral}$ $name_{Ion}$} option computes the pK_a value from the free energy difference of the neutral compound $name_{Neutral}$ and the ionic compound $name_{Ion}$ in solvent $name_{Solvent}$, where $name_i$ are the names of the compounds as given in the compound input section. By default, the pK_a is calculated for pure Solvent $i_{Solvent}$. However, it is also possible to calculate pK_a in a solvent mixture using the `xp={}` or `cp={}` commands (see section 2.3.6).
`reaction=i` or `nreaction=`$name_i$	Optional: Toggle a **reaction** equilibrium calculation in given solvent (see section 4.4). If the `reaction` keyword is given, the argument `i` is expected to be a positive integer number, which is the compound number in the sequence that was given in the compound input section. The `nreaction=`$name_i$ option computes the reaction equilibrium in the compound of the name $name_i$. By default, the reaction equilibrium is calculated at infinite dilution in compound `i/`$name_i$. It is also possible to calculate the reaction in a solvent mixture using the mixture concentration input `xr={}` or `cr={}` command (see section 4.4). If such a finite concentration input is used, arguments `i` or $name_i$ need not be given to the `reaction` or `nreaction` option.

Property calculation options, continued (phase diagram calculation)

`binary`	Optional: Toggle the automatic calculation of the **phase diagram** and the excess properties of a **binary** (two-compound) mixture (see section 2.3.7). This option is valid only if the total number of compounds is two.
`binary={i j}` or nbinary={name$_i$ name$_j$}	Optional: Toggle the automatic calculation of the **phase diagram** and the excess properties of a **binary** (two-compound) mixture (see section 2.3.7). This option is applicable if the total number of compounds is larger than two. For the `binary={i j}` option, the binary phase diagram is computed for the two compounds with the compound numbers `i` and `j` (in the order of compounds given in the compound input section). For the nbinary={name$_i$ name$_j$} option, the binary phase diagram is computed for the two compounds with the compound names name$_i$ and name$_j$ as given in the compound section of the COSMO*therm* input file (i.e. either the names of the COSMO files without extension or the name given via the `comp=name` option).
`ternary`	Optional: Toggle the automatic calculation of the phase diagram and the excess properties of a ternary (three-compound) mixture (see section 2.3.7). This option is valid only if the total number of compounds is three.
`ternary={i j k}` or nternary= {name$_i$ name$_j$ name$_k$}	Optional: Toggle the automatic calculation of the **phase diagram** and the excess properties of a **ternary** (three-compound) mixture (see section 2.3.7). This option is applicable if the total number of compounds is larger than two. For the `ternary={i j k}` option, the ternary phase diagram is computed for the three compounds with the compound numbers `i`, `j` and `k` (in the order of compounds given in the compound input section). For the nternary={name$_i$ name$_j$ name$_k$} option, the ternary phase diagram is computed for the three compounds with the compound names name$_i$, name$_j$ and name$_k$ as given in the compound section of the COSMO*therm* input file (i.e. either the names of the COSMO files without extension, or the name given via the `comp=name` option).
`multinary`	Optional: Toggle the automatic calculation of the **phase diagram** and the excess properties of a **multinary** (n-compound) mixture. Note, that the `multinary` option requires additional input (see section 2.3.7).

Property calculation options, continued (molecular similarity calculation options)

`similarity={i j}`
or
`nsimilarity={nam`$_i$ `nam`$_j$`}`
	Optional: Toggle the automatic calculation of the molecular **σ-profile similarity** of two compounds (see section 2.3.8). For the `similarity={i j}` option, the similarity factor is computed for the two compounds with the compound numbers `i` and `j` in the order of compounds given in the compound input section. For the `nsimilarity={nam`$_i$ `nam`$_j$`}` option, the similarity factor is computed for the two compounds with the compound names `nam`$_i$ and `nam`$_j$ as given in the compound input section of the COSMO*therm* input file (i.e. either the names of the COSMO files without extension, or the name given via the `comp=name` option).

`sms={i j}`
or
`nsms={nam`$_i$ `nam`$_j$`}`
	Optional: Toggle the automatic calculation of the molecular **σ-profile match similarity** of two compounds (see section 2.3.8). For the `sms={i j}` option, the similarity factor is computed for the two compounds with the compound numbers `i` and `j` in the order of compounds given in the compound input section. For the `nsms={nam`$_i$ `nam`$_j$`}` option, the similarity factor is computed for the two compounds with the compound names `nam`$_i$ and `nam`$_j$ as given in the compound input section of the COSMO*therm* input file (i.e. either the names of the COSMO files without extension, or the name given via the `comp=name` option).

Property calculation options, continued (molecular similarity calculation options)

`simpot={i j}` or nsimpot={nam_i nam_j}	Optional: Toggle the automatic calculation of the pure compound **σ-potential similarity** of two compounds (see section 2.3.8). For the `simpot={i j}` option, the similarity factor is computed for the two compounds with the compound numbers `i` and `j` in the order of compounds given in the compound input section. For the nsimpot={nam_i nam_j} option, the similarity factor is computed for the two compounds with the compound names nam_i and nam_j as given in the compound input section of the COSMO*therm* input file (i.e. either the names of the COSMO files without extension, or the name given via the `comp=name` option).
`simpot={i j k}` or nsimpot={nam_i nam_j nam_k}	Optional: Toggle the automatic calculation of the pure compound **σ-potential similarity** of two compounds weighted by the σ-profile of a third compound. Thus the computed similarity is a solute-specific σ-potential similarity (see section 2.3.8). For the `simpot={i j k}` option, the similarity factor is computed for the two compounds with the compound numbers `i` and `j` in the order of compounds given in the compound input section. The σ-potential similarity will be weighted by the σ-profile of compound `k`. For the nsimpot={nam_i nam_j nam_k} option, the similarity factor is computed for the two compounds with the compound names nam_i and nam_j as given in the compound input section of the COSMO*therm* input file (i.e. either the names of the COSMO files without extension, or the name given via the `comp=name` option). The σ-potential similarity will be weighted by the σ-profile of the compound with the name nam_k. If the third argument `k` or nam_k is not given, the unweighted, non-solute-specific σ-potential similarity (as defined above) is computed.
xs1={x_1 x_2 …} or cs1={c_1 c_2 …} and/or xs2={x_1 x_2 …} or cs2={c_1 c_2 …}	Optional for `simpot` or `nsimpot` computations: Give finite mixture concentrations for the two phases between which the sigma-potential similarity coefficient shall be computed. The input of the concentrations is possible either in mole fractions (`xs1={}`, `xs2={}`) or mass fractions (`cs1={}`, `cs2={}`) of the compounds of the mixture (as real numbers x_i and c_i). The arguments are expected as real numbers between zero and one in the same sequence of compounds as given in the second input area. If the values do not add up to one, COSMO*therm* will normalize them. If less mole fractions / concentrations than compounds are given, the missing ones will be assumed zero. If a negative number is given, the concentration for this compound will be inserted automatically using the normalization of the sum of mole fractions. Only one negative number is allowed per mixture input line.

Property calculation options, continued (QSPR property calculation options)

`PROPQSPR=filename`
or
`PROPQSPR=`
{c_1 c_2 … c_{18} prop}
or
`PROPQSPR_SI=filename`
or
`PROPQSPR_SI=`
{c_1 c_2 … c_{18} prop}
	Optional: Toggle the automatic calculation of a **σ-moment QSPR** property for all compounds in the input. The `PROPQSPR` command is closely related to the `QSPR` command of the global command section. There are some small differences in that the `PROPQSPR` command writes its results to the mixture section of the COSMO*therm* output file as well as to the COSMO*therm* table file, but not to the molecules σ-moment files. If no temperature and mixture concentration are given in the same mixture line, the `PROPQSPR` command will toggle the computation of the given QSPR property for all molecules. If temperature and mixture concentration are given, the `PROPQSPR` command will toggle the computation of the given QSPR property for all compounds, i.e. if several conformers are present for a given compound, the QSPR property will be averaged due to the Boltzmann conformer distribution of the conformers at the given temperature and mixture concentration. The `PROPQSPR=filename` option reads the σ-moment QSPR regression coefficients from file `filename` while for option `PROPQSPR=`{c_1 c_2 … c_{18} prop} coefficients are read from the input. For option `PROPQSPR_SI=` the coefficients must be defined in SI-units. If the coefficients are to be read from a file, the QSPR coefficient file `filename` is expected to be in the directory that also holds the COSMO*therm* parameter file, i.e. in the directory denoted by the environment variable `$COSMOTHERM_HOME` or in a directory denoted by the `cdir` command. For the definition of the QSPR coefficients and the format of the QSPR coefficient file, see section 4.5. If the coefficients are read from the input file via the `QSPR=`{c_1 c_2 … c_{18} prop} command, the coefficients c_i are expected as real numbers separated by blank spaces. If less than 18 coefficients are given, the missing ones are assumed to be zero. The property name `prop` is expected to be a string of up to 9 characters. For a further description of the σ-moment QSPR property computation, see section 4.5.

Property calculation options, continued (QSPR property calculation options)

`noaptab`	Optional for `PROPQSPR` computations (print option): do not print the computed Abraham parameter coefficients to the table file. Instead, only the final property computed from the Abraham coefficients will be listed as results column in the `PROPQSPR` table.
`pr_mom`	Optional for PROPQSPR computations (print option): print the most important QSPR descriptors to the table file.
`pr_allmom`	Optional for PROPQSPR computations (print option): print all possible QSPR descriptors to the table file.
`smomc[=file.momc]`	Optional for PROPQSPR computations (print option) with given temperature and mixture conditions: print averaged compound QSPR descriptors and the averaged compound QSPR properties to an additional σ-moments file with the extension `.momc`. If no argument is given the `.momc` file will get the name of the current input-file. Alternatively the filename of the `.momc` file can be given as argument of the `smomc` keyword.

Property calculation options, continued (surface activity, density, viscosity, and Liquid extraction)

`FlatSurf={i j}`
or
`NFlatSurf={nam`$_i$ `nam`$_j$`}`
	Optional: Toggle the automatic calculation of the **flat surface interaction energy** between two solvents (see section 4.10). For the `flatsurf={i j}` option, for all compounds, the surface interaction energy is computed at the surface interface of the two solvents with the compound numbers `i` and `j` (`i` and `j` being the compound numbers in the order given in the compound input section). For the `nflatsurf={nam`$_i$ `nam`$_j$`}` option, for all compounds, the surface interaction energy is computed at the surface interface of the two solvents with the compound names `nam`$_i$ and `nam`$_j$ as given in the compound input section of the COSMO*therm* input file.

`density`	Optional: Toggle the automatic calculation of the **density** ρ. This option computes the pure compound liquid density ρ for all given compounds. For details on the density calculation option see section 2.3.10.
`viscosity`	Optional: Toggle the automatic calculation of the **viscosity** η. This option computes the pure compound liquid viscosity η [cp] for all given compounds. For details on the viscosity calculation option see section 2.3.11.
`liq_ex`	Optional: Toggle the automatic calculation of a **multi-component two-phase liquid-liquid extraction equilibrium**. If toggled in a temperature/mixture line of the COSMO*therm* input file, the `LIQ_EX` option will compute the mole or mass based equilibrium partition of an arbitrary number of compounds between to given liquid phases (see section 2.3.12).

Print options for the COSMO*therm* output or table file:

`nomix`	Optional: Do not write the mixture information to the output file.
wcomp={i$_1$ i$_2$ …}	Optional: Write to the COSMO*therm* output file the evaluated information *only* for compounds i$_1$, i$_2$ ..., where i is the number of the compound as given in the compound section of the COSMO*therm* input file. The `wcomp={}` option can help to shorten the output file if not all evaluated information is required by the user. The `wcomp={}` option is active only for the temperature / mixture line where it is given.
wonly={name$_1$ name$_2$ …}	Optional: Write to the COSMO*therm* output file the evaluated information *only* for compounds name$_1$, name$_2$ ..., where name$_i$ is the name given in the compound section of the COSMO*therm* input file (i.e. it is either the name of the COSMO file without extension, or the name given via the `comp=name` option). The `wonly={}` command has the same effect as the `wcomp={}` command. It is active only for the temperature / mixture line where it is given.
`grad`	Optional: Print the values of the temperature and composition derivatives of the chemical potentials of all compounds. If given in a temperature/mixture line of the COSMO*therm* input file, the `grad` option is active only for the temperature / mixture line where it is given. See also section 4.6, "Chemical Potential Gradients" for further information.
`pvac`	Optional: Print the value of the the chemical potential of vacuum for the given mixture to the output file.
`wtln`	Optional: Print full compound and/or molecule names to all tables in the `name.tab` table output file and the `name.mom` sigma-moments file. By default the compound/molecule names are cropped after 20 characters.
`pzero`	Optional: Print concentrations that are zero to the table header of the `name.tab` table output file. By default only finite concentrations will be printed.
`noscreen`	Optional: Compute and print the mixture information (i.e. chemical potentials, log$_{10}$(partial pressure), total free energy in the mix, ...) of all compounds for all mixtures to the mixture/property output section of the COSMO*therm* output file `name.out`. By default only the data that is actually relevant for the property computation will be printed.
`pr_c`	Optional: Print concentration information in mass fractions c_i.
`pr_q`	Optional: Print concentration information in surface (area) fractions q_i.

Print options for the COSMO*therm* output or table file, continued (conformer information):

`ctab`	Optional: Print compound and conformer mixture information to the `COSMOtherm` table file. The `ctab` keyword toggles the additional tabulation of all mixture information, which by default is written to the ouput file only, to be printed to the COSMO*therm* table file. For each temperature/mixture as given in the mixture section of the input file a separate table will be created. If, in addition the `wconf` keyword is used, an additional table with the properties of the calculated COSMO*therm* mixture information for each individual conformer is written to the table file. By default all concentrations are written to the `ctab` table headers. If the the additional keyword `npzero` is used, only nonzero concentrations are written to the `ctab` table headers. To avoid an extremely large number of tables to be created, this option is only active for the plain "Temperature and Mixture" input, using `x` or `x_pure` as described in section 2.3), above.

Print options for the COSMO*therm* output file, continued (compound contact probability):

`contact` or contact={i_1 i_2 …} or ncontact={$name_1$ $name_2$}	Optional: Compute statistics of the surface contacts between the compounds in this mixture. If given in a temperature/mixture line of the COSMO*therm* input file, the `contact` option is active only for the temperature / mixture line where it is given. If the option `contact` is given without argument, the contact statistics is printed for all compounds in the mixture. If the option contact={i_1 i_2} is given, the contact statistics is printed *only* for compounds i_j, where `i` is the number of the compound in the order that is given in the compound section of the COSMO*therm* input file. If the option ncontact={$name_1$ $name_2$} is given, the contact statistics is printed *only* for compounds $name_1$, $name_2$ …, where $name_i$ is the name given in the compound section of the COSMO*therm* input file.
segment_contact={i_1 i_2 …}	Optional: Compute statistics of the surface segment contacts between the compounds in this mixture. If given in a temperature/mixture line of the COSMO*therm* input file, the `segment_contact` option is active only for the temperature / mixture line where it is given. The contact statistics is printed *only* for compounds i_j, where `i` is the number of the compound in the order that is given in the compound section of the COSMO*therm* input file. The molecular surface contacts for all segments of the given compounds are written to the contact statistics file `name.contact`, where `name` is the name of the COSMO*therm* input file. See section 0. for details.

`nohb`	Optional: Switch off hydrogen bonding (HB) contribution to the chemical potential. If used in this section of the input file, the `nohb` command is active only for the temperature / mixture line where it is given.
`novdw`	Optional: Switch off van der Waals (vdW) contribution to the chemical potential. If used in this section of the input file, the `novdw` command is active only for the temperature / mixture line where it is given.
`nothb`	Optional: Switch off temperature dependency of the hydrogen bond contribution to the total interaction energy of the compound. If used in this section of the input file, the `nothb` command is active only for the temperature / mixture line where it is given.
`notvdw`	Optional: Switch off temperature dependency of the van der Waals contribution to the total interaction energy of the compound. If used in this section of the input file, the `notvdw` command is active only for the temperature / mixture line where it is given.
`combi` or combi={i_1 i_2 …} or ncombi={$name_1$ $name_2$ …}	Optional: Switch off combinatorial contribution to the chemical potentials. If given in the mixture section of the COSMO*therm* input file, the `combi` command is active only for the temperature / mixture line where it is given. If the option `combi` is given without argument, the combinatorial contribution is switched off for all compounds in the mixture. If the option combi={i_1 i_2} is given, the combinatorial contribution is switched off *only* for compounds i_j, where `i` is the number of the compound in the order that is given in the compound section of the COSMO*therm* input file. If the option ncombi={$name_1$ $name_2$} is given, the combinatorial contribution is switched off *only* for compounds compounds $name_1$, $name_2$ …, where $name_i$ is the name given in the compound section of the COSMO*therm* input file.
`combi=ELBRO`	Optional: Switch on special free volume combinatorial contribution to the chemical potential. If given in a temperature / mixture line the `combi=ELBRO` command is active for the given mixture job option only. The free volume term by Elbro *et al.* is recommended for the computation of macromolecules (see section 4.3.2)
`uqmg`	Optional for the input of external quantum chemical energies and vibrational zero point energies: Use the external quantum chemical (QM) gas phase energies and vibrational zero point energies (EZP) as given in the compound input section (eqm and ezp options) or as read from the energy file for the Boltzmann-weighting of conformers. See section 2.2.2 for details on the input and the conformer weighting procedure.

Program control options, continued (IEI):

en_IEI={i_1 i_2 c_H c_S} or en_IEI_SI={i_1 i_2 c_H c_S}	Optional: Give the interaction energy parameters c_H (enthalpic contribution) and c_S (entropic contribution) of the interaction of two compounds identified by interaction energy indices (IEI numbers) i_1 and i_2. i_1 and i_2 are expected as integer numbers, the two parameters c_H (enthalpic contribution) and c_S (entropic contribution) are expected to be real numbers in [kcal/mol] and [kcal/mol K], respectively. If the en_IEI_SI option is used, c_H and c_S are expected to be real numbers in [kJ/mol] and [kJ/mol K], respectively. It is possible to give several en_IEI entries in one compound input line. The IEI formalism can be used to treat the concentration dependency if a reaction between two or more species in the mixture is taking place (e.g. the dimerization of a organic acids, the formation of a charge transfer complex between solute and solvent). See section 4.8 for a detailed description of the IEI method.
oclp	Optional for the use of conformers and IEI computations: Use the old conformer equilibration algorithm. Starting with version C21-0111, COSMO*therm* by default uses a novel conformer equilibration algorithm, which significantly improves computation time if conformers are present. Because the resulting conformer equilibrium may differ by a few (typically <2) percent from the two algorithms, the oclp option offers the possibility to use the old (slow) algorithm to be able to exactly reproduce old COSMO*therm* calculations. . If used in this section of the input file, the oclp command is active only for the temperature / mixture line where it is given.

2.3.1 Vapor Pressure

The **pvap** option allows for the automatic computation of vapor pressures over a given temperature range (and fixed mixture concentration). If no other input is given, the vapor pressure will be calculated only for the temperature given with the `tk=T`, `tc=T` or `tf=T` command. However, if a second temperature is given via the `tk2=T`$_2$, `tc2=T`$_2$ or `tf2=T`$_2$ commands, the vapor pressures are calculated over the range of temperatures spanned by the two values. By default, the vapor pressure then will be calculated at 10 temperature values (evenly spaced between `T` and `T`$_2$). The number of temperature points to be calculated in the temperature range can be changed via the `tstep=npoints` command. `npoints` is restricted to 101. Alternatively, the temperature points in the interval of `T` and `T`$_2$ can be determined by a given temperature stepsize via the `tstepsize=ΔT` command.

For a given pure compound or mixture composition *S*, the total vapor pressure of the system is computed from the *partial* vapor pressures of each compound *i*:

$$p_i^{S,vap}/1bar = \exp\left[\left(\mu_i^{Gas} - \mu_i^S\right)/RT\right] \tag{2.3.1-1}$$

At each temperature, for each compound *i* in the mixture *S*, the *partial* vapor pressures p_i^S, the chemical potential of the compound in the gas phase μ_i^{GAS}, and the enthalpy of vaporization ΔH_i^{vap} are written to the COSMO*therm* output file. The *total* vapor pressure of the mixture p_{tot}^S is written to the COSMO*therm* table file in tabulated form p^{VAP} vs. *T*. If the keyword `pr_pp` is given, the *partial* vapor pressures of all compounds in the mixture will be written to additional columns in the COSMO*therm* table file. In addition, the total chemical potentials of the liquid μ_{tot}^{Liquid} and of the gas phase μ_{tot}^{Gas}, as well as the heat of vaporization of the mixture ΔH^{vap} are written to the COSMO*therm* table file. Please note that in the case of mixtures, the given ΔH^{vap} value of the mixture is the sum of the partial ΔH_i^{vap} values of the contributing compounds. If the temperature conditions of a vapor pressure prediction are out of the temperature range where typical organic compounds are liquid (below 150K or over 550 K) the computed total vapor pressures will be written to the COSMO*therm* table file in square brackets (e.g. `T` and `PVtot` column look like this: "`723.15000 [0.12801479E+06]`", in the graphical user interface COSMO*therm*X such entries are marked red), indicating that the values were computed outside the core region of COSMO*therm* applicability and thus may show a higher than usual error. If the compounds melting point T_{melt} is known, i.e. if it is read from the compound input lines or the compounds .vap-file (see section 2.2.1), the predicted total vapor pressures will be written to the COSMO*therm* table file in round brackets (e.g. `T` and `PVtot` column look like this: "`173.15000 (0.51704579E-07)`" , in the graphical user interface COSMO*therm*X such entries are marked blue), if the temperature of the computation is below the melting point temperature of the given compound, or – if the vapor pressure of a mixture is computed – below the melting point of one of the components of the mixture, indicating that the vapor pressure thus computed corresponds to the vapor pressure of a subcooled melt. If three or more temperature points were calculated in a vapor pressure curve, the total vapor pressure will be fitted to Antoine's vapor pressure equation:

$$\ln(p) = A - B/(C + T) \tag{2.3.1-2}$$

where *T* is the temperature in [K] and *A*, *B,* and *C* are the coefficients of Antoine's equation. The resulting coefficients are written to COSMO*therm* output file and also to the COSMO*therm* table file. If available (i.e. if given in the input or read from a vapor pressure / property file, see section 2.2.3), experimental vapor pressures will be printed to the table file. This allows for the direct comparison of the calculated vapor pressures with experimental data. However, the output of experimental data is restricted to pure compounds, i.e. it is printed only if pure compounds vapor pressures are predicted, not for mixtures.

Furthermore the experimental values are printed only if we are inside the interpolative region of the given vapor pressure equation. Otherwise the experimental data field will be left blank. If the given temperature is below a compounds melting point, the automatic vapor pressure computation option **pvap** by default will compute the partial and total vapor pressure of the subcooled melt. However, if experimental data on a compounds solid state properties are available (in terms of the compounds melting point T_{melt} and free energy of fusion data $\Delta G_{fus}(T)$, as read from the compound input lines or the compounds vapor pressure and property .vap-file – see section 2.2.3), COSMO*therm* will compute the compounds partial sublimation pressures $p_i^{S,sub}$ and the according enthalpies of sublimation and writes tham to the COSMO*therm* output file. In addition, the total sublimation pressure of the mixture and the total enthalpy of sublimation are written to additional columns of the p^{vap} vs. T table in the COSMO*therm* table file.

$$p_i^{S,sub}/1bar = \exp\left[\left(\mu_i^{Gas} - \mu_i^{S} + \Delta G_i^{fus}\right)/RT\right] \tag{2.3.1-3}$$

The **pvap=pressure** [mbar] or **pvap_SI=pressure** [kPa] options allow for the iterative optimization of the temperature for a given vapor pressure: The temperature of the system is varied until difference of the COSMO*therm* prediction of the total vapor pressure and the given value of "`pressure`" is below a certain accuracy threshold. Thus the iterative `pvap=pressure` option allows the automatic calculation of a compounds boiling point at a given pressure. The default value for the accuracy threshold of the vapor pressure prediction is 10^{-4} mbar. It is possible to change this value using the `thresh_pvap=`Δp [mbar] or `thresh_pvap_SI=`Δp [kPa] options. Alternatively it is possible to use a relative accuracy threshold `thresh_pp=`p_p [%], which sets the accuracy of the iterative pressure optimization to p_p percent of the given reference `pressure`. During the optimization of the temperature, for each temperature and for each compound in the mixture the *partial* vapor pressures, the chemical potential of the compound in the gas phase and its enthalpy of vaporization are written to the COSMO*therm* output file. If convergence is reached, i.e. if the required accuracy threshold is met, the *total* vapor pressure of the mixture is written to the COSMO*therm* table file in tabulated form p^{vap} vs. T. In addition, the total chemical potentials of the liquid μ_{tot}^{Liquid} and of the gas phase μ_{tot}^{Gas}, as well as the heat of vaporization of the mixture ΔH^{vap} are written to the COSMO*therm* table file.

A characteristic trait of COSMO*therm*'s vapor pressure prediction is that the absolute quantitative prediction of the vapor pressure at a given temperature point shows a certain error, but that the temperature dependency of the vapor pressure, i.e. the qualitative shape of the $p^{vap}(T)$ curve is predicted very well. Thus it is possible to predict a $p^{vap}(T)$ vapor pressure curve to a very high accuracy if the predicted vapor pressure is adjusted to match a given experimental reference pressure p_{Ref} at a given reference temperature T_{Ref}. In practice, the pressure difference between the COSMO*therm* vapor pressure prediction $p_i^{S,vap}(T_{Ref})$ and the given reference pressure $p^{Ref}(T_{Ref})$ is used to determine a correction term to chemical potential in the gas phase $\mu_i^{Gas,corr}$, which subsequently is used to scale the vapor pressure prediction, yielding a corrected vapor pressure value $p_i^{S,corr}$, which replaces the regular prediction value $p_i^{S,vap}$.

$$p_i^{S,corr}/1bar = \exp\left[\left(\mu_i^{Gas} + \mu_i^{Gas,corr} - \mu_i^{S}\right)/RT\right] \tag{2.3.1-4}$$

COSMO*therm* offers several possibilities to include a reference pressure and temperature into the `pvap` prediction option: it is possible to use a reference temperature T_{Ref} and pressure p_{Ref} pair, which can be given in the mixture input line together with the `pvap` option. Input of a reference temperature T_{Ref} is possible via keywords **use_tref**, `use_tref_C`, `use_tref_K`, or `use_tref_F`, processing reference temperature values in units degree Celsius (for the first two keywords), Kelvin, and Fahrenheit, respectively. Input of reference pressure p_{Ref} is possible via keywords **use_pref**, `use_pref_Pa`, `use_pref_kPa`, `use_pref_bar`, or `use_pref_psia`, processing reference pressure values in units millibar, Pascal, kiloPascal, bar, and psia, respectively. Further, for pure compound and mixture vapor pressure predictions, there is the possibility to use the compound's experimental normal boiling point temperature as reference point(s). The compound's boiling point temperature, as read from the compound input line, or from the vapor pressure property file (see section 2.2) is used as reference temperature $T_{Ref} = T_{Boil}$ assuming normal boiling point conditions, thus the reference pressure used is $p_{Ref}(T_{Ref})$ = 1 atm = 1013.25 mbar. The usage of pure compound boiling points in a pure compounds vapor pressure prediction is toggled by the keyword **`use_tboil`**, which either can be given in the `pvap` mixture line, where it is active for this mixture only, or in the global input lines, where it will be active for all `pvap` computations given. The adjustment of the vapor pressure prediction with the given boiling point data is possible for pure compound vapor pressure computations as well as for mixtures. In the latter case COSMO*therm* will perform a number of additional computation steps in which the vapor pressures of the individual compounds are adjusted to their given boiling points data. Of course, this adjustment is only possible if an experimental boiling point temperature was provided for the compound, either in the compound input line, or, in its vapor pressure property file. In addition to the use of boiling points as reference pressures COSMO*therm* the possibility to use the compound's experimental vapor pressure as reference point. The compound's temperature dependent vapor pressure, as read from the compound input line, or from the vapor pressure property file (see section 2.2) in terms of pairs of experimental $p^0(T)$, *or,* Antoine, extended Antoine, or Wagner equation coefficients, directly can be used as reference pressures $p_{Ref} = p^0_{Exp}(T)$. The usage of temperature dependent pure compound vapor pressures is toggled by the keyword **`use_pvapt`**, which either can be given in the `pvap` mixture line, where it is active for this mixture only, or in the global input lines, where it will be active for all `pvap` computations given.

Suboptions of the `pvap` command:

`tk2=`T_2 or `tc2=`T_2 or `tf2=`T_2	Optional for `pvap` computations: Give a second temperature, which together with the temperature from the `tk=`T_2, `tc=`T_2 or `tf=`T_2 command defines the temperature range for the automatic vapor pressure computation. The `tk2=`T_2, the `tc2=`T_2 and the `tf2=`T_2 command expect as argument a temperature T_2 in [K], in [°C], and in [°F], respectively.
`tstep=npoints`	Optional for `pvap` computations: Give the number of temperature values to be computed in the given temperature range. The argument `npoints` is expected as an integer number between 1 and 101. Default value is `npoints=10`.
`tstepsize=ΔT` `tstepsize_F=ΔT`	Optional for `pvap` computations: Give a temperature step value `ΔT` to be computed in the given temperature range. Argument `ΔT`, the temperature stepsize, is expected as a positive real number in [K] for option `tstepsize` and in [°F] for option `tstepsize_F`, respectively. Note that option `tstepsize` overrides the `tstep` option. If the given temperature stepsize `ΔT` is larger than the given temperature interval or, if the given temperature stepsize `ΔT` is too small to fill up the the given temperature interval with the maximum number of allowed steps, the `tstepsize` option will be disabled and the `tstep` default will be used instead.
`logPVAP`	Optional for `pvap` computations: Output of the pressure in the table file will be written in decadic logarithmic $log_{10}[p]$ units if the additional keyword `logPVAP` is given (or if the `logPVAP` keyword is given instead of the `pvap` keyword).
`use_tboil`	Optional for pvap computations: Use the pure compound boiling points T_{Boil} (as given in the compound input section or read from a compounds vapor-pressure property file, cf. section 2.2) as a reference point for scaling the vapor pressure prediction. The `use_tboil` option is valid for pure compound and mixture vapor pressure predictions.
`use_pvapt`	Optional for `pvap` computations: Use the temperature dependent experimental pure compound vapor pressure $p^0_{Exp}(T)$ (as given in the compound input section or read from a compounds vapor-pressure property file, cf. section 2.2) as a reference point for scaling the vapor pressure prediction. The `use_pvapt` option is valid for pure compound and mixture vapor pressure predictions.
`pr_pp`	Optional for `pvap` computations of mixtures: Print partial pressures of the compounds in the mixture to the results table.

Suboptions of the `pvap` command (continued):

`use_tref=value`
or
`use_tref_C=value`
or
`use_tref_K=value`
or
`use_tref_F=value`
	Optional for `pvap` computations: Input of reference temperature T_{Ref} for the purpose of scaling the vapor pressure prediction to match a pair of given reference temperature T_{Ref} and pressure p_{Ref}. The argument of the `use_tref=value` option is expected to be a real number temperature, which is expected to be in [°C] for keywords `use_tref` and `use_tref_C`, in [K] for keyword `use_tref_K`, and in [F] for keyword `use_tref_F`.

`use_pref=value`
or
`use_pref_Pa=value`
or
`use_pref_kPa=value`
or
`use_pref_bar=value`
or
`use_pref_psia=value`
	Optional for `pvap` computations: Input of reference pressure p_{Ref} for the purpose of scaling the vapor pressure prediction to match a pair of given reference temperature T_{Ref} and pressure p_{Ref}. The argument of the `use_pref=value` option is expected to be a real number pressure, which is expected to be in [mbar] for keyword `use_pref`, in [Pa] for keyword `use_pref_Pa`, in [kPa] for keyword `use_pref_kPa`, in [bar] for keyword `use_pref_bar`, and in [psia] for keyword use_pref_`psia`.

Suboptions of the `pvap=pressure` and the `pvap_SI=pressure` commands:

`thresh_pvap=Δp`
or
`thresh_pvap_SI=Δp`
	Optional for `pvap=pressure` computations: Give the absolute accuracy threshold for the iterative vapor pressure computation. The `thresh_pvap=Δp` and the `thresh_pvap_SI=Δp` commands expect as argument a pressure value Δp in [mbar] and in [kPa], respectively.

`thresh_pp=`p_p	Optional for `pvap=pressure` computations: Give a relative accuracy threshold for the iterative vapor pressure computation. The `thresh_pp=`p_p command expects as argument a relative pressure p_p in [%].

2.3.2 Activity Coefficients

The **gamma=i** or **ngamma=name$_i$** option allows for the automatic computation of the activity coefficients in solvent `S` or a solvent mixture. By default, this option will compute the chemical potentials μ_j^P of all pure compounds `j` and subsequently the chemical potentials μ_j^S at infinite dilution in a given solvent compound `S`. The logarithmic activity coefficients are calculated as

$$\ln\left(\gamma_j\right) = \left(\mu_j^S - \mu_j^P\right) / RT \tag{2.3.2-1}$$

The ln(γ_j) values are written to the COSMO*therm* output file and to the COSMO*therm* table file. It is also possible to calculate the activity coefficients at a given finite mixture concentration via the xg={x_1 x_2 …} or cg={c_1 c_2 …} commands (see below).

For ionic species `j` the reference state for the activity coefficient is not the pure compound μ_j^P but infinite dilution of the ionic species in the solvent `i`. Thus according to Debye-Hückel limiting law, the activity coefficient for ions and the mean ionic activity coefficient of a salt (anion plus cation) will be one at infinite dilution of the salt. It is possible to override the defaults for the reference state by explicitly fiving a reference state concentration to COSMO*therm*. This is possible with the xref={x_1 x_2 …} or cref={c_1 c_2 …} suboptions of `gamma` (see table below). The explicit reference state given in the input will be applied to all compounds in the mix and override the defaults for neutral and charged compounds.

Suboptions of the `gamma=i` and the ngamma=name$_i$ option:

xg={x_1 x_2 …} or cg={c_1 c_2 …}	Optional for `gamma` computations: Give finite mixture concentration at which the activity coefficient shall be computed. The input of the concentrations is possible either in mole fractions (`xg={}`) or mass fractions (`cg={}`) of the compounds of the mixture (as real numbers x_i and c_i). The arguments are expected as real numbers between zero and one in the same sequence of compounds as given in the second input area. If the values do not add up to one, they will be normalized by COSMO*therm*. If less mole fractions / concentrations than compounds are given, the missing ones will be assumed zero. If a negative number is given, the concentration for this compound will be inserted automatically using the normalization of the sum of mole fractions. Only one negative number is allowed per mixture input line.

Note: If a finite concentration input via options `xg` or `cg` is used, no arguments need to be given to the `gamma` or `ngamma` option.

Suboptions of the `gamma=i` and the `ngamma=name`$_i$ option, continued:

xref={x_1 x_2 …} or cref={c_1 c_2 …}	Optional for `gamma` computations: Give finite mixture concentration of the reference state of the activity coefficient computation. This option overrides the default reference states (i.e. the pure compound for neutral species and infinite dilution in the solvent for ionic species) and is used for all compounds in the mixture. The input of the reference state concentrations is possible either in mole fractions (`xref={}`) or mass fractions (`cref={}`) of the compounds of the mixture (as real numbers x_i and c_i). The arguments are expected as real numbers between zero and one in the same sequence of compounds as given in the second input area. If the values do not add up to one, they will be normalized by COSMO*therm*. If less mole fractions / concentrations than compounds are given, the missing ones will be assumed zero. If a negative number is given, the concentration for this compound will be inserted automatically using the normalization of the sum of mole fractions. Only one negative number is allowed per mixture input line.

2.3.3 Henry-Law Coefficients

The **`henry=i`** or **`nhenry=name`**$_{i}$ option allows for the automatic computation of the Henry law coefficients $H^{(i)}$ in compound or solvent mixture `S`. It is also possible to calculate the Henry law coefficients at a given finite concentration (for example in a mixture of solvents) via the `xh={x1 x2 …}` or `ch={c1 c2 …}` commands. By default, this option will compute the chemical potentials μ_j^P of all pure compounds `j` and subsequently the chemical potentials μ_j^S of all compounds `j` at infinite dilution in compound or solvent mixture `S`. In addition, the vapor pressures of the pure compounds are calculated. The Henry law coefficients H_j^S for all compounds `j` are then calculated from the activity coefficients and the vapor pressures of the compounds and written to the COSMO*therm* output file and to the COSMO*therm* table file. For a given solute compound the Henry law coefficient H_j^S is computed from the compounds chemical potential difference between the ideal gas phase (μ_i^{Gas}) and the infinite dilution state in the given solvent or solvent mixture `S` ($\mu_i^{S,\infty}$), which for an incompressible liquid state is equivalent to the product of the infinite dilution activity coefficient $\gamma_i^{S,\infty}$ of `j` in `S` with compound `j`'s pure compound vapor pressure p_j^P.

$$H_j^S = \left[\left(\mu_j^{s,\infty} - \mu_j^{Gas}\right)/\,RT\right] = \gamma_j^{S,\infty} p_j^P \tag{2.3.3-1}$$

If possible, i.e. if experimental pure component vapor pressure data is available from a vap-file or given in the compound input section of the input-file, the Henry law coefficient as computed with the experimental vapor pressure $H_j^{S,\,pv\text{-}exp} = \gamma_i^{S,\infty} p_i^{P,\,exp}$ will also be computed and written to additional lines and columns of the COSMO*therm* output and table files, respectively.

In addition to the Henry law constant, it is possible to compute the closely related Gibbs free energy of solvation G_i^{Solv} for all compounds. This is possible with the keyword `Gsolv`. There are two possible reference framwork that G_i^{Solv} can be computed in:

- By default, i.e. if `Gsolv` is given without argument, G_i^{Solv} of all compounds is computed in the *"molar"* framework: i.e. G_i^{Solv} is the free energy of transfer of a solute molecule *i* from an ideal gas at *molar* (1 mol/l) concentration to an ideal solution at the same solute concentration. The reference state of this calculation is 1 l of ideal gas and 1 l of liquid solvent. In this reference state the free energy of solvation is computed as $G_i^{Solv} = (\mu_i^{S,\infty} - \mu_i^{Gas}) - RT \ln(\rho_S V_{IG} / MW_S)$, where μ_i^{Gas} is the chemical potential of the compound in the ideal gas phase, $\mu_i^{S,\infty}$ is the infinite dilution chemical potential of the compound in solution, MW_S is the molecular weight of the solvent, V_{IG} is the molar volume of the ideal gas, and ρ_S is the density of the solvent (or solvent mixture). The density of the solvent (or solvent mixture) ρ_S can be provided to COSMO*therm* via the the `solvdens` keyword. If ρ_S is not given in the input, COSMOtherm will estimate the solvents density with the liquid density/volume QSPR method as described in section 2.3.10. Please note that for solvent mixtures the density estimate is a linear interpolation between the pure compounds densities, where excess density/volume is neglected. If no density estimate is possible, a default value, the liquid density of water, about 997 [g/l], is used. If the solvent used, as given by the `henry=i` or `nhenry=name`$_{i}$ option is pure water (i.e. if the Gibbs free energy of *hydration* is calculated) COSMO*therm* automatically will fill in the exact density of the water. Thus, if the Gibbs free energy of hydration is calculated, it is not necessary to give the solvent density via the input.
- If `Gsolv` is given with argument `Gsolv=bar-mol` or `Gsolv=reference`, G_i^{Solv} of all compounds is computed in the theoretical *"COSMO-RS reference"* framework (i.e. reference state of the calculation is 1 bar of ideal gas and 1 mol of liquid solvent) via $G_i^{Solv} = (\mu_i^{S,\infty} - \mu_i^{Gas})$ where μ_i^{Gas} is the chemical potential of the compound in the ideal gas phase, and $\mu_i^{S,\infty}$ is the infinite dilution chemical potential of the compound in solution.

Suboptions of the `henry=i` and the `nhenry=name`$_i$ option are:

`xh={x1 x2 …}`
or
`ch={c1 c2 …}`
	Optional for `henry` computations: Give finite mixture concentration at which the activity coefficient shall be computed. The input of the concentrations is possible either in mole fractions (`xh={}`) or mass fractions (`ch={}`) of the compounds of the mixture (as real numbers `xi` and `ci`). The arguments are expected as real numbers between zero and one in the same sequence of compounds as given in the second input area. If the values do not add up to one, they will be normalized by COSMO*therm*. If less mole fractions / concentrations than compounds are given, the missing ones will be assumed zero. If a negative number is given, the concentration for this compound will be inserted automatically using the normalization of the sum of mole fractions. Only one negative number is allowed per mixture input line.

`logH`	Optional for `henry` computations: Output of the Henry law constant in the table file will be written in log[pressure] units if the additional keyword `logH` is given (or if the `logHenry` keyword is given instead of the `henry` keyword).
`Gsolv[=refstate]`	Optional for `henry` computations: Output of the Gibbs free energy of solvation G_i^{Solv} as additional column in the table file. If `Gsolv` is given without argument, or if it is given with arguments `Gsolv=l-l` or `Gsolv=molar`, G_i^{Solv} of all compounds is computed in the *molar* framework (i.e. reference state of the calculation is 1 l of ideal gas and 1 l of liquid solvent). If `Gsolv` is given with argument `Gsolv=bar-mol` or `Gsolv=reference`, G_i^{Solv} of all compounds is computed in the *COSMO-RS reference* framework (i.e. reference state of the calculation is 1 bar of ideal gas and 1 mol of liquid solvent).
`solvdens=value`	Optional for `henry/Gsolv` computations: Optional input of the solvent density as used in the computation of the molar Gibbs free energy of solvation. The argument `value` is expected to be the density value of the solvent in [g/l]. Note that the `solvdens` option only will effect the computation of G_i^{Solv} in the *molar* framework. If `Gsolv=reference` is used, the `solvdens` keyword will not have any effect on the computed G_i^{Solv}.

Note: If a finite concentration input via options `xh` or `ch` is used, no arguments need to be given to the `henry` or `nhenry` option.

2.3.4 Solubility of Solids and Liquids

The **`solub=i`** or **`nsolub=name`**$_i$ option allows for the automatic computation of the solubility of liquid or solid compounds in solvent `i`. It is also possible to calculate the solubilities for a mixture of solvents at a given finite concentration via the `xs={x1 x2 …}` or `cs={c1 c2 …}` commands. By default, this option will compute the chemical potentials $\mu_j^{(P)}$ of all pure compounds `j`, the chemical potentials $\mu_j^{(H2O)}$ of all compounds in water and subsequently the chemical potentials $\mu_j^{(\infty)}$ at infinite dilution in compound `i`. If the compound is solid, the energy change of a compound from the subcooled liquid state to the ordered solid state has to be taken into account. I.e. the solutes Gibbs free energy of fusion ΔG_{fus} (or equivalently its Gibbs free energy of crystallization $\Delta G_{cryst} = -\Delta G_{fus}$) has to be either given and computed from experimental data or estimated by COSMO*therm*. The solubility of a solute *j* in a solvent or solvent mixture *i* is calculated

$$\log_{10}\left(x_j^{SOL(0)}\right) = \left[\mu_j^{(P)} - \mu_j^{(\infty)} - \max\left(0, \Delta G_{fus}\right)\right] / \left(RT \ln(10)\right) \qquad (2.3.4\text{-}1)$$

The solubility thus computed is a zeroth order approximation. It corresponds to the approximation that $x^{(0)}_{SOL} \approx 1/\gamma_\infty$. This zeroth order solubility is valid only for small concentrations of the solute, i.e. if the solubility itself is small. If the solubility of the solute in the solvent is large (x_{SOL}>0.1), $x^{(0)}_{SOL}$ is a poor approximation. However, x_{SOL} can be refined iteratively: If the zeroth order $x^{(0)}_{SOL}$ is re-substituted into the solubility calculation, a better approximation for x_{SOL} is achieved:

$$\log_{10}\left(x_j^{SOL(1)}\right) = \left[\mu_j^{(P)} - \mu_j^{(i)}\left(x_j^{SOL(0)}\right) - \max\left(0, \Delta G_{fus}\right)\right] / \left(RT \ln(10)\right) \qquad (2.3.4\text{-}2)$$

I.e. the solubility now is calculated in a mixture of the solvent with solute *j* in a mole fraction concentration $x^{(0)}_{SOL}$. The $x^{(1)}_{SOL}$ value thus computed can again be re-substituted into the solubility equation and again a better guess for x_{SOL} can be achieved. This procedure can be iterated until the computed value of x_{SOL} is constant. In practice the iterations are done until the difference of $|x^{(k)}_{SOL} - x^{(k-1)}_{SOL}|$ is below a given threshold. In COSMO*therm*, this iterative procedure is toggled with the keyword **`iterative`**, which is a sub option of the `solub` or `nsolub` option. To avoid long calculation times, it is possible to limit the maximum number of iterations in an iterative solubility calculation using the keyword **`max_iterations`**. If the `iterative` solubility computation is used, it is possible to print the noniterative results to the table output file as well, using the **`pr_ni`** ("print noniterative results") keyword. If `pr_ni` is given, the *noniterative* results for the decadic logarithm of x_{SOL}, the chemical potential of the solute in the initial solvent $\mu_j^{(i,0)}$, the mass based solubility *w* [g/g] and, if possible, the molar solubility *S* in [mol/l] (for definition of the latter two, see below) will be printed to additional columns of the COSMO*therm* table file. Please note that the `iterative` solubility computation is ambiguous and physically ill-defined if the solute is part of the solvent mixture (e.g. if you want to compute the `iterative` solubility of water in a given mixture of 0.9 mole fractions of octanol and 0.1 mole fractions of water). In such a case COSMO*therm* will assume that the solute concentration in the solvent is zero and compute the `iterative` solubility in the resulting solvent mixture (in the given example COSMO*therm* will compute the `iterative` solubility of the water in the pure octanol). The additional solubility unit conversions (mass based solubility and molar solubility, see below) will be done using the original solvent concentrations, however. In general, there is no guarantee that the `iterative` solubility option will converge to a refined solubility value. If the convergence of iterative refinement of the solubility fails, the noniterative solubility value will be printed to the COSMO*therm* table file, and a warning message will be printed to the table and output files. In addition, the noniterative solubility value printed to the COSMO*therm* table file will be enclosed by square brackets `[…]`. If an oscillating behavior is detected in the iterations of the solubility concentrations, COSMO*therm* applies a sequence of techniques to force the convergence. If such a forced convergence is used, it will be noted in the table and output files. In addition, the noniterative solubility value printed to the

COSMO*therm* table file will be enclosed by parentheses `(…)`. In most cases, the "forced" solubility value thus computed will be the correct solubility. However there is one situation, where the forced convergence value might not be correct: if at a given temperature the system shows SLLE behavior, i.e. there is simultaneous phase separation in the liquid and precipitation of the solute to the solid phase. The `iterative` solubility option can not resolve this behavior and in rare cases a false solubility value at the may be optimized (although the error thus made should be quite small, only a few percent of the solubility, in general). Thus to account for this special kind of phase behavior, a SLE (and/or LLE) calculation should be done for the compound. SLE/LLE calculations either can be performed with the binary phase diagram option (see section 2.3.7) or in combination with the automatic solubility computation option `solub`.

The computation of a SLE/LLE calculation as part of a solubility computation is toggled with the keyword **`SLESOL`**, which is a sub option of the `solub` or `nsolub` option. The `SLESOL` and the `iterative` solubility options are mutually exclusive. If the `SLESOL` keyword is used, COSMO*therm* attempts to solve the actual solid-liquid (SLE), and, if present liquid-liquid (LLE), or solid-liquid-liquid (SLLE) equilibrium conditions for each solute with the given solvent or solvent mixture. The equilibrium conditions that are solved for are described in section 2.3.7.3 and 2.3.7.4. The use of the `SLESOL` option has several implications, the most important one being computational time: the explicit solution of the phase equilibrium conditions requires a considerably higher number of COSMO*therm* function evaluations than the simple recursive/algorithmic solution of the `iterative` option. Typically, with the `SLESOL` option computational time will increase by a factor of 10 or more compared to the `iterative` option. Unlike the latter there is no ambiguity in the results of the `SLESOL` option. The solubilities thus computed are actual solutions of the systems phase equilibrium conditions. Unlike the `iterative` case, complex phase situations like the SLLE case mentioned above, can be resolved correctly by the `SLESOL` option. Some care has to be taken in the interpretation of the results of the two options. While the `iterative` solubility does not distinguish between liquid and solid solubility in terms of the given results, the `SLESOL` option does: both the results of the solid-liquid equilibrium and the liquid-liquid-equilibrium conditions are written to the solubility table. There are three situations where the results of the `SLESOL` and `iterative` option may differ. First, if the solubility of a liquid is computed, i.e. if the Gibbs free energy of fusion of the solute ΔG_{fus}=0, the `iterative` option will provide just that: the liquid solubility log(x_S). In contrast, the log(x_{SLE}) solubility value predicted by `SLESOL` option will be zero, because solid-liquid equilibrium SLE conditions can not be solved for if there is no solid phase. Instead, the liquid-liquid equilibrium x'_{LLE} and x''_{LLE} values will be written to additional columns of the solubility table output. Second, if the given solute is solid, with a finite Gibbs free energy of fusion ΔG_{fus}, and the solute's virtual liquid shows a phase separation with the liquid solvent (i.e. the SLLE case described above), as noted before, the `iterative` option may show ambiguous or undetermined results. The `SLESOL` option will solve such a system correctly solving the solid-liquid phase equilibrium conditions with the boundary condition of the virtual liquid-liquid phase equilibrium. If this is the case, the solubility table output will show finite numbers for both the solid's log(x_{SLE}) solubility output as well as for the virtual liquid's x'_{LLE} and x''_{LLE} values. Third, in cases where the solubility is very high the `iterative` option may fail to converge properly or converge to "miscibility" where predicted log(x_S)=0. The `SLESOL` option may still be able to resolve such cases and predict a finite solubility value log(x_{SLE}).

In addition to the absolute solubility predictions defined by equations 2.3.4-1 and 2.3.4-2, it is possible to compute *relative* solubility values using the **`pr_rs`** ("print relative solubility") keyword. The relative solubility x_{RS} is defined as x_{RS} = exp($-\mu_j^{(i,0)}/RT$). It is important to realize that the value of the predicted x_{RS} does not make sense as a single absolute number! It only reasonably can be used in comparison with other relative solubility numbers (e.g. comparing the x_{RS} values of a given solute in several different solvents). This opens up the possibility to compare relative solubilities in cases where the absolute values can not be compared any more: e.g. if the solvent is predicted to be fully miscible with the solvent by the absolute solubility prediction – in this case the relative solubility x_{RS} can provide qualitative and even quantitative information about the solubility behavior of the given solute in a set of solvents. In addition, please note that a relative solubility computation only makes sense for noniterative solubility calculations and thus the `pr_rs` keyword is inactive if the the `iterative` refinement of the solubility computation is switched on. If the `pr_rs` keyword is given, the decadic logarithm of x_{RS}, the mass based relative solubility w_{RS} [g/g] and, if possible, the molar solubility S_{RS} in [mol/l] (for definition of the latter two, see below) will be printed to additional columns of the COSMO*therm* table file.

Because COSMO*therm* can only calculate compound in a (subcooled) liquid, for the solubility of solid compounds, the Gibbs free energy of fusion ΔG_{fus} has to be taken into account. The definition of Gibbs free energy of fusion in equations 2.3.4-1 and 2.3.4-2 assumes that ΔG_{fus} is the energy that is required to form an ordered solid from the random particle distribution in the subcooled liquid. Thus a positive value for ΔG_{fus} indicates that the solute is solid, while zero or negative vales of ΔG_{fus} indicate that the solute is liquid at the given temperature condition. Please note that this definition of ΔG_{fus} is different from the definitions used in the articles published on the solubility topic[32,33]. In these articles the negatively signed definition of the solids free energy (i.e. ΔG_{cryst}) was used, but this property was referred to as ΔG_{fus}. Thus the definitions of ΔG_{fus} in the COSMO*therm* program and in the articles[32,33] are not consistent, but can easily be translated into each other by changing their sign. There are two possibilities of incorporating ΔG_{fus} in a COSMO*therm* calculation: either experimental data of the solid-liquid phase transition may be introduced into the calculation, or COSMO*therm* may provide an estimate for ΔG_{fus}:

ΔG_{fus} can be given in the compound section of the COSMO*therm* input file via option `DGfus=value` (see section 2.2.1). Please note that by default only positive values of ΔG_{fus} are allowed to be read in from the input. In addition the free energy of fusion is used with a max(0, ΔG_{fus}) function in equations 2.3.4-1 and 2.3.4-2. Although unphysical it might be eligible to allow negative ΔG_{fus} values in certain situations. The use of negative ΔG_{fus} values can be forced by the global `ndgf` option (see section 2.1).

A temperature dependent heat of fusion can be calculated from the Schröder-van Laar equation[28] if the compounds enthalpy or entropy of fusion (ΔH_{fus} or ΔS_{fus}, respectively) and melting temperature (T_{melt}) are known:

$$\Delta G_{fus}(T) = \Delta H_{fus}\left(1-\frac{T}{T_{melt}}\right) - \Delta Cp_{fus}\left(T_{melt}-T\right) + \Delta Cp_{fus} T \ln\frac{T_{melt}}{T}$$

$$\Delta G_{fus}(T) = \Delta S_{fus}\left(T_{melt}-T\right) - \Delta Cp_{fus}\left(T_{melt}-T\right) + \Delta Cp_{fus} T \ln\frac{T_{melt}}{T} \qquad (2.3.4\text{-}3)$$

Thus the combination of a compounds experimental melting temperatures and enthalpy or entropy and optionally heat capacity of fusion (eq. 2.3.4-3 as used in eqs. 2.3.4-1 and 2.3.4-2) allows the automatic calculation of the compounds solid solubility at different temperatures with the `solub` or `nsolub` option. ΔH_{fus} or ΔS_{fus} can be given in the compound section of the COSMO*therm* input file via option `DHfus=value` or `DSfus=value`, respectively (see section 2.2.1). A compounds melting temperature T_{melt} can be given in the compound section of the COSMO*therm* input file via option `Tmelt=temp` (see section 2.2.1). Optionally the heat capacity of fusion ΔCp_{fus} can also be used to decribe the temperature dependency of the Gibbs free energy of fusion. ΔCp_{fus} can be given in the compound section of the COSMO*therm* input file with option `Dcpfus=value` (see section 2.2.1). If no `Dcpfus` value is given it is assumed zero. Because ΔCp_{fus} is a

---

[28] Prigogine, I., R Defay, R. *Chemical Thermodynamics*, Longmans Greeen, London, 1953.

property that is not readily available by experiment, but at same time quite often a non-neglible property, it has been proposed[29,30] that the estimation of $\Delta Cp_{fus} \approx \Delta S_{fus} = \Delta H_{fus} / T_{melt}$ is closer to the physical reality for nonspherical and neutral molecules than the assumption that $\Delta Cp_{fus} = 0$. Please note that the ΔCp_{fus} estimate shows unphysical behavior if the absolute zero point temperature is approached. Thus we recommend to use the ΔCp_{fus} estimate only if two prerequisites are met: 1) the solute compounds are neutral and nonspherical, and 2) the temperature is over 150 K. In COSMO*therm* the ΔCp_{fus} estimate is toggled by the keyword **`dcpfus_estimate`**, which can be given either in the global input lines (see section 2.1), or for specific compounds in the compound input line (see section 2.2.1), or in the mixture input line where the solubility calculation is defined.

A different pathway to the determination of ΔG_{fus} from experimental data is the computation of reference solubility: if the experimental solubility of a given solute in a given solvent or mixture at a given temperature is known, this reference solubility x^{REF}_{SOL} can be used to determine the compounds free energy of fusion at these conditions by solving equation 2.3.4-2 for ΔG_{fus}. In the COSMO*therm* input the reference solubility $\log_{10}(x^{REF}_{SOL})$ can be read in by the **`ref_sol_s=value`** keyword. COSMO*therm* offers several alternative unit systems for the input of the reference solubility (see "Suboptions of the `solub` option" Tables, below). In addition to the reference solubility, the reference solute has to be defined, which is possible via the **`solute=j`** or **`nsolute=name`**$_j$ keywords, which define the solute compound of number `j` or name `name`$_j$, as given in the compound section, respectively. Please note that the reference solubility calculation of ΔG_{fus} also is possible for salt solubility calculations (see below). To be effective in the salt solubility option, the arguments of the `solute=j` or `nsolute=name`$_j$ keywords may point to any of the salt components, or alternatively, the expression `nsolute=SALT` can be used. If both reference solute and solubility value are given correctly, COSMO*therm* will compute ΔG_{fus} of the given solute compound and print the computed value to the "`DG_fus`" column of the solubility table output row of the reference solute compound. Any other sources of ΔG_{fus} will be overridden by this option. Thus the solubility computation using the value of ΔG_{fus} thus computed, will reproduce the given solutes reference solubility, unless noted otherwise by eventual warning messages, that is. If there is available an experimental value of the solute compounds melting temperature (for neutral compounds from the compound vap-file or as given in the compound input section via the `Tmelt=temp` option, see section 2.2.1 above, or, for salts, from the mixture input section, see below) COSMO*therm* also will compute the enthalpy of fusion ΔH_{fus} of the solute compound and print it together with the melting temperature into additional columns of the solubility table in the COSMO*therm* table file. The ΔG_{fus} (and ΔH_{fus}) values thus computed can be used in subsequent solubility computations for this solute compound, as is taken advantage of by the solubility screening panel of COSMO*therm* graphical user interface COSMO*therm*X.

Alternatively, ΔG_{fus} can be estimated by COSMO*therm*. This is possible via a QSPR approach. The free energy of fusion can be approximated from the following COSMO*therm* descriptors using:

$$-\Delta G_{fus} = c_1 \mu_j^{(H2O)} + c_2 N_j^{Ring} + c_3 V_j + c_4 \qquad (2.3.4\text{-}4)$$

c_1 to c_5 are the QSPR parameters for the free energy of fusion. $\mu_j^{(H2O)}$ is the chemical potential of solute *j* in water, N_j^{Ring} is the number of ring atoms in compound *j* and V_j is the Volume of the compound. Please note,

---

[29] Neau, S.H., Bhandarkar, S.V., and Hellmuth, E.W., *Pharm. Res.,* **14**, 601 (1997).

[30] Hildebrand, J.H., Prausnitz, J.M., and Scott, R.L., *Regular and Related Solutions*, Van Nostrand Reinhold Co., New York, 1970.

that this QSPR approach is valid only for the estimation of heats of fusion at room temperatures (i.e. T=25°C). Please note that due to a known systematic error of COSMO and COSMO-RS methods for secondary and tertiary aliphatic amines[3] it is necessary to correct the chemical potential of such compounds by a value of about -2.5 kcal/mol times N_j^{Amino}, the number of secondary or tertiary aliphatic amino groups in the compound[33]. The N_j^{Amino} descriptor and its coefficient c_5 = 2.5 kcal/mol is needed only if the solvent in which the solubility is calculated is water. For any other solvent c_5 is zero. An approximate temperature dependency for the ΔG_{fus} term of the solubility QSPR is used. This term uses **Walden's Rule** claiming that the temperature dependency of the free energy of fusion of nonsymmetric organic molecules can be approximated by ΔS_{fus} = 0.0135 kcal/mol K. Note that the ΔCp_{fus} estimate described above (toggled by the keyword **`dcpfus_estimate`**) may also be used in combination with QSPR ΔG_{fus} and Walden's rule. The precautions noted above apply in this case too.

The different estimates or data inputs that are possible for the input of a compound's heat of fusion are processed in a hierarchical manner. The explicit input of a ΔG_{fus} value is used with the highest priority. If no explicit ΔG_{fus} value is given, then the computation of ΔG_{fus} with the Schröder-van Laar equation (eqs. 2.3.4-3 above) will be used if all necessary input data is present. If neither of these input options is present, the QSPR term of eq. 2.3.4-4 will be used to estimate ΔG_{fus}. This implies that a compound input where experimental ΔG_{fus} data fed into the compound section this experimental data due to it's higher priority, aleways overrides the QSPR estimate. This hierarchy can be overturned by the option `force_qspr`, which enforces the use of the QSPR estimate of ΔG_{fus} for all compounds in the compound list.

Quite frequently a compounds melting temperature T_{melt} is known experimentally, but none of its enthalpy of fusion ΔH_{fus}, entropy of fusion ΔS_{fus}, or heat capacity of fusion ΔCp_{fus} data. In such a case ΔG_{fus} has to be estimated by the QSPR approach of equation 2.3.4-4. If available for a given compound, COSMO*therm* will use the melting temperature information T_{melt} to determine if the compound is a solid at the given conditions of the solubility computation. This information will be written to additional columns in the table output of the `solub` computation table output. In addition, if the compound is known to be liquid at the given temperature conditions (i.e. if $T > T_{melt}$) the QSPR-derived Gibbs free of fusion ΔG_{fus} for this compound will be set to zero.

If possible, the five QSPR parameters are read from the `CTDATA` file. Currently the parameterization for single-point DFT/COSMO calculations upon semiempirically optimized MOPAC-AM1/COSMO-geometries (namely `BP_SVP_AM1_C30_1701.ctd`) and the parameterizations for fully relaxed Turbomole or DMOL3 DFT/COSMO calculations with the larger TZVP basis set (namely `BP_TZVP_C30_1701.ctd` and `DMOL3_PBE_C30_1701.ctd`) as well as the parameter set for the FINE level (`BP_TZVPD_FINE_C30_1701.ctd`) include solubility parameters that were derived from a set of solubility data of 150 solid compounds[31]. A closer description of the fitting procedure used to gain the solubility parameters is given in references 32 and 33. It is also possible to explicitly give the values of the QSPR parameters in the COSMO*therm* input file using the `solQSPR={c1 c2 c3 c4 c5}` command.

---

[31] Duffy, E. M. and W. L. Jorgensen, *J. Am. Chem. Soc.,* **122**, 2878 (2000).

[32] Klamt, A.; F. Eckert and M. Hornig, "COSMO-RS: A novel view to physiological solvation and partition questions", *J. Computer Aided Mol. Design*, **15**, 355 (2001).

[33] Klamt, A.; F. Eckert; M. Hornig; M. E. Beck and T. Bürger, "Prediction of aqueous solubility of drugs and pesticides with COSMO-RS", *J. Comput. Chem.*, **23**, 275 (2002).

The solubility of *liquid* compounds also can be calculated with the `solub` option. If the keyword **`liquid`** is given in addition to the `solub` keyword, the heats of fusion of all compounds ΔG_{fus} are assumed to be zero. In this case no additional input of ΔG_{fus} or QSPR parameters is necessary.

The computed values of $\log_{10}(x_j^{SOL})$ are written to the COSMO*therm* output file and to the COSMO*therm* table file. In addition, all of the computed chemical potentials and QSPR descriptors as well as the given or computed ΔG_{fus} value are written to the table file. For the `SLESOL` option, additional columns with the virtual or apparent liquid-liquid phase equilibrium values x'_{LLE} and x''_{LLE} will be written to the solubility results table.

Moreover, the mass based solubility w_j^{SOL} [$g_i/g_{solution}$] will be written to the COSMO*therm* table file as well as the COSMO*therm* output file. There are three possible definitions of the mass based solubility, where MW_i and $MW_{Solvent}$ are the molecular weights of the solute *i* and the solvent (or solvent mixture):

- By default (and/or if the keyword **`wfract`** is given in addition to the `solub` keyword), the *normalized mass fraction* solubility is computed: $w_i^{SOL} = w_i^{FRACT} = x_i^{SOL} * MW_i / (x_i^{SOL} * MW_i + (1- x_i^{SOL}) * MW_{Solvent})$.
- If the keyword **`wsol2`** is given in addition to the `solub` keyword), the *unnormalized* mass based solubility is computed by "Definition 2": $w_j^{SOL} = w_j^{SOL2} = x_i^{SOL} * MW_i / ((1- x_i^{SOL}) * MW_{Solvent})$.
- If the keyword **`wsol1`** is given in addition to the `solub` keyword, the *unnormalized* mass based solubility is computed by "Definition 1": $w_j^{SOL} = w_j^{SOL1} = x_i^{SOL} * MW_i / MW_{Solvent}$. Please note that, although commonly used in pharmaceutical chemistry, definition 1 is an approximation derived for small solubilities. Thus it should be used only if the solubility is expected to be small ($\log_{10}(x_j^{SOL}) < -3$).

Please note that if, at the given conditions, a compound is miscible with the solvent (i.e. if the logarithm of the mole fraction solubility is zero) the mass based solubility is not well-defined and thus it will not be printed to output and table file.

Finally, if the density of the solvent and solute is known, or can be estimated, the decadic logarithm of the *molar* solubility of the solute in the *solution* ($\log_{10}(S$ [mol/l]$) = \log_{10}(x_i^{SOL} / V_{Solution})$) will be written to an additional column in the COSMO*therm* table file. The molar volume of the solution $V_{Solution} = \rho_{Solution} / MW_{Solution}$ is calculated from the molar masses and densities of the given solvent $\rho_{Solvent}$ and solute ρ_l according to the solubility x_i^{SOL} computed. The density of the solvent (or solvent mixture) $\rho_{Solvent}$ can be provided to COSMO*therm* via the the `solvdens` keyword. If $\rho_{Solvent}$ is not given in the input, COSMOtherm will try to estimate the solvents density with the liquid density/volume QSPR method as described in section 2.3.10. Please note that for solvent mixtures the density estimate is a linear interpolation between the pure compounds densities, where excess density/volume is neglected. The density of the solute ρ_i also is estimated by the liquid density/volume QSPR. If no density estimate is available, the molar solubility of the solute in the solvent will not be computed. If the **`wsol1`** keyword is given in addition to the `solub` keyword (see above), the decadic logarithm of the *molar* solubility of the solute in the *solvent* ($\log_{10}(S$ [mol/l]$) = \log_{10}(x_i^{SOL} * \rho_{Solvent} / MW_{Solvent})$) will be written to the COSMO*therm* table file.

Suboptions of the `solub=i` or `nsolub=name`$_i$ command are:

`liquid`	Optional for `solub` computations: Computed liquid solubilities only: Heats of fusion for all compounds are assumed to be zero.
`iterative[=thresh_solub]`	Optional for `solub` computations: Refine the computed solubilities iteratively, i.e. iterate the mixture computations for each compound until the difference of the computed solubility log(x_S) of the compound differs less than threshold value `thresh_solub` from log(x_S) of the last iteration. If no argument is given with the `iterative` command, COSMO*therm* will use the default value of `thresh_solub=1.0d-5`, i.e. of 10^{-5} log(x_S) units. The optional argument `thresh_solub` is expected as a real number larger than zero.
`SLESOL`	Optional for `solub` computations: solve the solid-liquid (SLE) and/or liquid-liquid (LLE) equilibrium conditions to obtain the solubility value. The `SLESOL` option can be used as an alternative to the `iterative` refinement option. Both options are mutually exclusive.
`pr_ni`	Optional for `solub` computations with `iterative` refinement or `SLESOL`: In addition to the iteratively refined solubilities, print the *noniterative* (zero-order guess) solubility values to additional columns of the table file.
`pr_rs`	Optional for `solub` computations without `iterative` refinement: In addition to regular, absolute solubilities, print the *relative* solubility values to additional columns of the table file.
`max_iterations=i`	Optional for iterative `solub` computations: Give the maximum number of iterations `i` that are used in the iterative refinement procedure for the solubility. Default value of `max_iterations` is `999`. Argument `i` is expected as an integer number larger than one.

`solQSPR={`$c_1\ c_2\ c_3\ c_4\ c_5$`}`
or
`solQSPR_SI={`$c_1\ c_2\ c_3\ c_4\ c_5$`}`
	Optional for `solub` computations: Give the parameters for the QSPR approach for the free energy of fusion. The arguments are expected as real numbers. The parameters are expected to be used with energy values in [kcal/mol] and volumes in [Å^3]. For the `solQSPR_SI` command, they are expected to be used with energy values in [kJ/mol] and volumes in [nm^3] (i.e. in the SI-unit frame).

`force_qspr`	Optional for `solub` computations: Enforce the use of the QSPR estimate of the free energy of fusion for all compounds. This option prevents the usage of any given input data of the free energy of fusion.

Suboptions of the `solub=i` or `nsolub=name`$_i$ command, continued:

`xs={x1 x2 …}`
or
`cs={c1 c2 …}`
	Optional for `solub` computations: Give finite solvent mixture concentration at which the solubility shall be computed. The input of the concentrations is possible either in mole fractions (`xs={}`) or mass fractions (`cs={}`) of the compounds of the mixture (as real numbers `xi` and `ci`). The arguments are expected as real numbers between zero and one in the same sequence of compounds as given in the second input area. If the values do not add up to one, COSMO*therm* will normalize them. If less mole fractions / concentrations than compounds are given, the missing ones will be assumed zero. If a negative number is given, the concentration for this compound will be inserted automatically using the normalization of the sum of mole fractions. Only one negative number is allowed per mixture input line.

`wfract`	Optional for `solub` computations: Change the output of the mass based solubility w_j^{SOL} to normalized mass fraction output, which is $w_i^{FRACT} = x_i^{SOL} *MW_i / (x_i^{SOL} *MW_i + (1- x_i^{SOL})*MW_{Solvent})$.
`wsol1`	Optional for `solub` computations: Change the output of the mass based solubility w_j^{SOL} to "Definition 1" which is $w_j^{SOL1} = x_i^{SOL} *MW_i / MW_{Solvent}$.
`wsol2`	Optional for `solub` computations: Change the output of the mass based solubility w_j^{SOL} to "Definition 2" which is $w_j^{SOL2} = x_i^{SOL} *MW_i / ((1- x_i^{SOL})*MW_{Solvent})$.
`solvdens=value`	Optional for `solub` computations: Optional input of the solvent density as used in the computation of the molar solubility. The argument `value` is expected to be the density value of the solvent in [g/l]. Note that the `solvdens` option only will effect the computation of the *molar* solubility. The mole fraction solubility and the mass based solubility will not be changed.

Note: If a finite concentration input via options `xs` or `cs` is used, no arguments need to be given to the `solub` or `nsolub` option.

Suboptions of the `solub=i` or `nsolub=name`$_i$ command, continued:

`dcpfus_estimate`	Optional for the input of a temperature dependent compound free energy of fusion (via input of enthalpy or entropy of fusion ΔH_{fus} or ΔS_{fus}, and melting point T_{melt}): toggle the approximation of the heat capacity of fusion as $\Delta Cp_{fus} = \Delta S_{fus} = \Delta H_{fus}/T_{melt}$. The value of $\Delta G_{fus}(T)$ thus obtained wil be used to compute the solubility of the compounds. If the `Dcpfus_estimate` keyword is given in a solubility computation input line, the approximation to ΔCp_{fus} is valid for all compounds and the given mixture line.
`amino_corr`	Optional for `solub` computations: Toggle correction term for secondary and tertiary amino groups for the computation of aqueous solubility. By default the amino correction is used only if a solids solubility in pure water is computed using the QSPR estimate to the free energy of fusion. The `amino_corr` command overrides this default and forces the solub option to use the amino correction term (with the side condition that the solvent is water).

Suboptions of the `solub=i` or `nsolub=name`$_i$ command, continued (input of reference solubility):

`ref_sol_s=value`
or
`ref_sol_x=value`
or
`ref_sol_c=value`
or
`ref_sol_g=value`
or
`ref_sol_m=value`
or
`ref_sol_l=value`
	Optional for `solub` computations: Define the reference solubility value for a reference solubility calculation of ΔG_{fus}. The `ref_sol_s` option assumes that the given reference solubility `value` is given as the decadic logarithm of the solutes mole fraction $\log_{10}(x^{REF}_{SOL})$. The `ref_sol_x` option assumes that the given reference solubility `value` is given as the solutes mole fraction x^{REF}_{SOL}. The `ref_sol_c` option assumes that the given reference solubility `value` is given as the solutes mass fraction concentration c^{REF}_{SOL}. The `ref_sol_g` option assumes that the given reference solubility `value` is given as the solutes mass based solubility g^{REF}_{SOL} in [g/g]. By default, (or if the **`wsol2`** keyword is given), the input of the mass based solubility g^{REF}_{SOL} is assumed to be the unnormalized mass based solubility of "*Definition 2*": $g^{REF}_{SOL} = w^{REF}_{SOL2} = x_i^{SOL} *MW_i / ((1-x_i^{SOL})*MW_{Solvent})$. If the additional keyword **`wsol1`** is given, the input as well as the output of the mass based solubility g^{REF}_{SOL} is assumed to be the unnormalized mass based solubility of "*Definition 1*": $g^{REF}_{SOL} = w^{REF}_{SOL1} = x_i^{SOL} *MW_i / MW_{Solvent}$. If the additional keyword **`wfract`** is given, the input as well as the output of the mass based solubility g^{REF}_{SOL} is assumed to be the mass fraction (i.e. identical to the input of the `ref_sol_c` option). The `ref_sol_m` option assumes that the given reference solubility `value` is given as the solutes molar concentration M^{REF}_{SOL} in [mol/l]. Please note that the input of a molar reference solubility requires the input of the solvent density (**`solvdens`** option, above). If no solvent density is given, the conversion of the molar solubility to mole fraction can only be done in an approximative way. The `ref_sol_l` option assumes that the given reference solubility `value` is given as the solutes mass/volume concentration L^{REF}_{SOL} in [g/l]. Please note that the mass/volume concentration input also requires the solvent density. Along the lines of the mass based solubility input, the mass/volume concentration input assumes that L^{REF}_{SOL} is given by mass "*Definition 2*" as noted above, unless the `wsol1` or `wfract` keywords are used, in which case the input of L^{REF}_{SOL} is assumed to be a "*Definition 1*" mass solubility, or a mass fraction, respectively.

Suboptions of the `solub=i` or `nsolub=name`$_i$ command, continued (input of reference solubility):

`solute=j` or `nsolute=name`$_j$	Optional for `solub` computations: Define the solute compound for a reference solubility calculation of ΔG_{fus}. The `solute=j` option computes defines the reference solute compound `j`, which is the compound number in the collating sequence of the compound input section. The `nsolute=name`$_j$ option computes the reference solubility for the solute compound of the name `name`$_j$.

Dissociation Corrections to the Solubility of Neutral Compounds

In solubility calculations of compounds solvent in protic solvent water it is possible that acidic or basic solutes dissociate in the aqueous solvent phase. The dissociated species typically have solution properties different from the solubility of the undissociated compound, thus affecting the apparent solubility of the compound. The effect of the dissociated compound approximatively can be taken into account by applying a dissociation correction to the solubility of the neutral compound. Using the assumption that the dissociation reaction is independent from the solution process and further assuming that the dissociation process is described solely by the solutes dissociation constant pK_a and the given pH of the solvent phase, a dissociation correction term can be derived from the law of mass action. The dissociation corrected mole fraction solubility x_S^{DC} is computed from the solubility of the undissociated species x_S (as computed with the regular `solub` option) and a correction term derived from the mole fraction concentration of the dissociated ionic species x_D:

$$\log\left(x_S^{DC}\right) = \log\left(x_S\right) + \log\left(\frac{x_S + x_D}{x_S}\right) \tag{2.3.4-5}$$

If the solute is an acid HA ($x_S = x_{HA}$) showing the dissociation reaction $HA + H_2O \rightleftharpoons A^- + H_3O^+$ in water, the concentration of the dissociated ionic species $x_D = x_{A-}$ is computed from eq. 2.3.4-6, employing the acids aqueous dissociation constant **pK_a(acid)** and the **pH** of the aqueous phase:

$$c_{A^-} = -\frac{1}{2}10^{-pH} + \sqrt{\frac{1}{4}10^{-2pH} + c_S 10^{-pK_a(acid)}} \tag{2.3.4-6}$$

Molar concentrations c_i relate to mole fraction concentration via $x_S = c_S MW_{Sol}/\rho_{sol}$, where MW_{Sol} is solvent molar weight and ρ_{sol} is solvent density. If the solute is a base B ($x_S = x_B$) showing the dissociation reaction $B + H_2O \rightleftharpoons BH^+ + OH^-$ in water, the concentration of the dissociated ionic species $x_D = x_{BH+}$ is computed by equation 2.3.4-7, employing the base's aqueous dissociation constant **pK_a(base)** and the **pH** of the aqueous phase:

$$c_{BH^+} = -\frac{1}{2}10^{(pH - pK_W)} + \sqrt{\frac{1}{4}10^{(2pH - 2pK_W)} + c_S 10^{(pK_a(base) - pK_W)}} \tag{2.3.4-7}$$

Equations 2.3.4-6 and 2.3.4-7 were derived from the law of mass action of a dissociating system in a buffered solution using the self-dissociation constant of water $pK_W = 14$. By default, COSMO*therm* will apply eq. 2.3.4.2 to all solutes denoted as acids, whose aqueous pK_a(acid) is given, and eq. 2.3.4-7 to all solutes denoted as bases, whose aqueous pK_a(base) is given. Equations 2.3.4-6 and 2.3.4-7 can be simplified using the additional assumption of an unbuffered solution (i.e. the initial concentration of the dissociated water H_3O^+ and OH^- is zero), which leads to the well known Henderson-Hasselbalch equation[34], which for an acids dissociation reaction $HA + H_2O \rightleftharpoons A^- + H_3O^+$ holds:

---

[34] Henderson, L.J. *Am. J. Physiol.* **21** (1908) 173. Hasselbalch, K.A. *Biochemische Zeitschrift* **78** (1917) 112.

$$c_{A^-} = c_S 10^{(pH - pK_a(acid))} \tag{2.3.4-8}$$

For a base B showing the dissociation reaction B + $H_2O \rightleftharpoons BH^+ + OH^-$, the Henderson-Hasselbalch correction term holds:

$$c_{BH^+} = c_S 10^{(pK_a(base) - pH)} \tag{2.3.4-9}$$

Please note, that the Henderson-Hasselbalch equations 2.3.4-8 and 2.3.4-9 are based upon the side condition that the concentration of the dissociated species in the solution is negligible. If this assumption breaks down, i.e. if the system dissociates strongly at the given pH conditions, the Henderson-Hasselbalch correction term will become erroneous[35]. In such a case equations 2.3.4-6 or 2.3.4-7 should be applied instead. Accordingly, equations 2.3.4-6 and 2.3.4-7 are used by default if a dissocation corrected solubility calculation is toggled in COSMO*therm* using the `solub` option. The Henderson-Hasselbalch equations 2.3.4-8 and 2.3.4-9 can be used optionally, by means of the keyword **`use_hh`**.

Note that equations 2.3.4-5 to 2.3.4-9 are based on the implicit assumption that the dissociated species are completely dissolvable in the solvent phase. As this assumption may not hold any more for nonaqueous solvents, the DC correction of solubility calculations currently is restricted to solvent pure water.

The automatic solubility (`solub`) option of COSMO*therm* will apply the dissociation correction, to any solute of which the pK_a value is given or estimated. The solubility value thus computed will be written to the COSMO*therm* output file as well as to an additional column in the table output of the `solub` option in the COSMO*therm* table file. In addition, the pK_a(acid) or pK_a(base) value used in the dissociation correction calculation is printed to the output and table files. The dissociation corrected solubility values are identified by a "DC" (Dissociation Corrected) tag. A solutes aqueous dissociation constant pK_a(acid) or pK_a(base) value may either be given to COSMO*therm* as additional input in the vap-file, compound input line or logp option mixture input line, or, it may be estimated by COSMO*therm* using the pK_a prediction methodology described in section 2.3.6 of this manual.

There are three possibilities to enter a compounds aqueous dissociation constant pK_a(acid) or pK_a(base):

- specify the pK_a value in the compounds vapor pressure property (**`.vap`** file) using the **`pK_acid=value`**, or **`pK_base=value`** option (see section 2.2).
- specify the pK_a value in the compound input line using the **`pK_acid=value`**, or **`pK_base=value`** option (see section 2.2).
- specify the solute compound and the pK_a value in the `solub` option mixture input line using the **pKacid={i$_{\text{Solute}}$ value}**, or **pKbase={i$_{\text{Solute}}$ value}** options, where **i$_{\text{Solute}}$** is the number of the solute compound as given in the collating sequence of compounds in the compound input (see below).

As an alternative to the input of the solute pK_a values, COSMO*therm* may also compute an estimate of a solutes pK_a(acid) or pK_a(base) using the Linear Free Energy Relationship (pK_a-LFER) described in section 2.3.6. In the framework of solubility calculations, the pK_a estimate for acidic and basic solutes is toggled by the **comp_acid={i$_{\text{Neutral}}$ i$_{\text{Anion}}$}**, and **comp_base={i$_{\text{Neutral}}$ i$_{\text{Cation}}$}** options, given in the `solub` option

---

[35] Po, Henry N.; Senozan, N. M. *J. Chem. Educ.* **78** (2001) 1499.

mixture input line. Therein $\mathbf{i}_{\mathbf{Neutral}}$ is the number of the solute compound as given in the collating sequence of compounds in the compound input, and $\mathbf{i}_{\mathbf{Anion}}$ and $\mathbf{i}_{\mathbf{Cation}}$ are the compound numbers of the anion and cation compounds that are formed on acidic and basic dissociation of the solute compound, respectively. Please note that the pK_a-LFER estimate of pK_a(acid) or pK_a(base) values requires the presence of COSMO files of the dissociated species A^- or BH^+, which form from solute via the acidic or basic dissociation reaction. For more details on the pK_a-LFER methodology, see section 2.3.6.

The p*H* of the aqueous phase is assumed to be `7.0`. This value can be changed by the input option **`pH=value`**. Please note that for measurements of solubility of drugs, the pH of the aqueous phase is buffered to a specific value, most often to a value of `pH=7.4`, which is the p*H* of blood serum and thus of significance in physiological partition.

Dissociation correction ("DC") suboptions of the `solub` or `nsolub` option:

`pKacid=`
`{`i_{Solute} `value}`
or
`pKbase=`
`{`i_{Solute} `value}`
	Optional for `solub` computations: Input of experimental aqueous dissociation constants for acidic (`pKacid` option) or basic (`pKbase` option) solutes for a log*D* distribution coefficient computation. Argument i_{Solute} is the compound number of the solute. It is expected to be an integer number designating a compound in the sequence of the compound input. Argument `value` is the pK_a(acid) (`pKacid` option) or pK_a(base) (`pKbase` option) value for solute i_{Solute}. It is expected to be a real number. It is possible to give up to n_{comp} (total number of compounds) `pKacid` and `pKbase` input fields in one `logp` mixture input line.

`comp_acid=`
`{`$i_{Neutral}$ i_{Anion}`}`
or
`comp_base=`
`{`$i_{Neutral}$ i_{Cation}`}`
	Optional for `solub` computations: Input of experimental aqueous dissociation constants for acidic (`pKacid` option) or basic (`pKbase` option) solutes for a log*D* distribution coefficient computation. Argument $i_{Neutral}$ is the compound number of the solute. It is expected to be an integer number designating a compound in the sequence of the compound input. Argument i_{Anion} (for the `pKacid` option) or i_{Cation} (for the `pKbase` option) are the compound numbers of the ionic compounds that result from the acidic or basic dissociation reaction of the given solute $i_{Neutral}$. It is possible to give up to n_{comp} (total number of compounds) `comp_acid` and `comp_base` input fields in one `logp` mixture input line.

`pH=value`	Optional for `solub` computations: Input of the logarithmic acidity/basicity (p*H)* of the water phase for a log*D* distribution coefficient computation. Argument `value` is the p*H* value of the water solvent phase. It is expected to be a real number. If no `pH=value` is given, it is assumed that the aqueous phase is neutral (p*H*=7).
`use_HH`	Optional for `solub` computations: Use Henderson-Hasselbalch dissociation correction instead of default buffered solution term.

Solubility of Salts, Complexes and Cocrystals

The prediction of salt solubility involves a few complications. First, in COSMO*therm* a salt $A\nu_A C\nu_C$ is treated by means of its anion A and cation C with stoichiometries ν_A and ν_C, respectively. To obtain a salts solubility, the chemical potentials have to be determined for the individual anion A and cation C and and the heat of fusion of the salt ΔG_{fus} has to be known. The salt solubility x_{AC}^{SOL} is computed from the mean chemical potentials and the heat of fusion of the salt:

$$\log_{10}\left(x_{AC}^{SOL}\right) = \left[\left(\mu_{AC}^{(0)} - \mu_{AC}^{(s)} - \max\left(0, \Delta G_{fus}^{Salt}\right)\right)/\nu_{tot}\right]/\left(RT\ln(10)\right) \qquad (2.3.4\text{-}10)$$

where $\nu_{tot} = \nu_A + \nu_C$ is the sum of the ion stoichiometries. The chemical potential of the pure salt $\mu_{AC}^{(0)}$ is the stoichiometric sum of the chemical potentials of anion A and cation C. This means, the chemical potential of the pure salt is the sum of the chemical potentials of anion μ_A and cation μ_C determined in an stoichiometric mix of anion A and cation C ($x_A^{(0)} = \nu_A / \nu_{tot}$, $x_C^{(0)} = \nu_C / \nu_{tot}$): $\mu_{AC}^{(0)} = \nu_A \mu_A^{(0)} + \nu_C \mu_C^{(0)}$. The chemical potential of the salt in solution $\mu_{AC}^{(i)}$ is the stoichiometric sum of the chemical potentials of the anion A and cation C computed in infinite dilution in solvent *s*. This means, the chemical potentials of the soluted salt is the stoichiometric sum of the chemical potentials of anion $\mu_A^{(i)}$ and cation $\mu_C^{(i)}$ determined in pure solvent *s*: $\mu_{AC}^{(s)} = \nu_A \mu_A^{(s)} + \nu_C \mu_C^{(s)}$. The salt or, if $\Delta G_{fus} = 0$, ionic liquid solubility x_{AC}^{SOL} as calculated by equation 2.3.1a, can be considered a mean ionic solubility, a definition that is similar to the mean ionic activity coefficient of Debye-Hückel theory, but has a different reference state: whereas the mean ionic activity coefficient uses the infinite dilution of the salt in the solvent as reference state for the ions, the solubility option uses the pure salt as reference state, which is the stoichiometric mix of the anions and cations. The salt solubility as computed from eq. 2.3.4-10 assumes that the free energy of fusion value, which describes the transfer of the salt from the subcooled liquid state to the crystalline solid state, is defined as "bulk" free energy of fusion of the salt ΔG^{Salt}_{fus}, which implies that the salt is one unified compound and not made of individual ions. This is the most common definition of a salts free energy of fusion. Most experimental ΔG_{fus} data for salts is defined this way. By default COSMO*therm* uses this salt free energy of fusion ΔG^{Salt}_{fus} in combination with equation 2.3.4-10 to compute salt solubilities. However, in some cases the free energy of fusion is defined in an alternative way, as a "mean" ionic free energy of fusion ΔG^{AC}_{fus}. Using this definition of the solid phase, equation 2.3.4-10 has to be reformulated as:

$$\log_{10}\left(x_{AC}^{SOL}\right) = \left[\left(\mu_{AC}^{(0)} - \mu_{AC}^{(i)}\right)/\nu_{tot} - \max\left(0, \Delta G_{fus}^{AC}\right)\right]/\left(RT\ln(10)\right)$$

(2.3.4-11)

Thus the mean salt free energy of fusion and the bulk salt free energy of fusion are related simply as $\Delta G^{AC}_{fus} = \Delta G^{Salt}_{fus} / \nu_{tot}$. Both definitions of ΔG_{fus} as used in equations 2.3.4-10 and 2.3.4-11 are available in COSMO*therm*. Currently definition 2.3.4-10 is used by default, while use of 2.3.4-11 can be toggled by keyword (`dgfmean` keyword, see below).

A special suboption to the `solub` or `nsolub` option allows for the automatic computation of the solubility of a salt or ionic liquid compound in a given solvent or mixture: The composition of the salt has to be defined in the input. This can be done with the **`salt={i j …}`** command, where `i`, `j`,… are the numbers of the ion compounds (numbers as given in the sequence of compounds in the compound input section) or the **`nsalt={name_i name_j …}`** command, where `name_i`, `name_j`, … are the names of the ion compounds as given in the compound input section. In addition the salt stoichiometry has to be defined for the individual anion and cation compounds: The input of the salt stoichiometry factors ν_i, ν_j is possible via the **`salt_n={ν_i ν_j …}`** command, where ν_i, ν_j,… are the stoichiometry numbers of the salt compound defined by the ions as given by the `salt={i j …}` or `nsalt={name_i name_j …}` command. For example the salt sodiumsulfate (Na_2SO_4) would be defined by the combination of the option `nsalt={na so4}`, defining the salt composition from sodium cation (`na.cosmo`) and sulfate dianion (`so4.cosmo`), and the option `salt_n={2 1}`, defining the stoichiometry of the salt (i.e. two sodium ions, one sulfate ion). The definition of the salt is not restricted to simple binary salts with ionic components. It is also possible to include neutral components into the definition of the salt compound. Thus complexated salts (salt compound that include one or several molecules of a solvent in their crystal lattice, e.g. gypsum $CaSO_4 \cdot 2H_2O$) and even cocrystals (two or more distinct molecular components within the crystal lattice that are assumed to crystallize together) can be treated with the salt solubility option as well. In such a case - i.e. if a salt like gypsum is dissoluted in solvent water, which also is a component of its salt stoichiometry – COSMO*therm* will solve the thermodynamic equilibrium condition for this system and notify this in the output table of the solubility.

If the `salt` and `salt_n` options are given as additional input to an automatic solubility computation ("`solub`" option) input line, COSMO*therm* will compute the solubility of the salt of the given composition and stoichiometry via eq. 2.3.4-10 and write the results to the COSMO*therm* output and table files. If the `salt` option is used, COSMO*therm* only will compute the solubility of the given solute (salt or cocrystal) compound (this is unlike the "simple" `solub` option for individual neutral compounds, which always computes the solubilities of all compounds as given in the compound input). The options for the choice of the solvent (`solub=i` or `nsolub=name`$_i$ options) or a solvent mixture (`xs={`x_1 x_2 …`}` or `cs={`c_1 x_2 …`}` options) are as described above for the solubility of individual compounds. The iterative refinement of the computed solubility value as defined by eq. 2.3.4-2 ("`iterative`" option) is also possible for a salt solubility computation. Application and restrictions of the `iterative` salt solubility compuation are the same as for the neutral compound solubility option, above.

Although, it is possible to define a QSPR estimate for the free energy of fusion of salts similar to the QSPR model for neutral compounds, currently no parameters are available for this model[36]. Thus the default value for the heat of fusion salt solubility computation is zero (i.e. it is assumed that the salt in fact is a ionic liquid). It is possible however to provide experimental data, which can be used to compute the salt free energy of fusion via eq. 2.3.4-3 and which subsequently can be used in eq. 2.3.4-10 to compute a solid salts solubility value. The input of the required experimental data is similar to the input of the free energy of fusion related data of pure compounds in the pure compound input section (see section 2.2 of this manual) – with the difference that the salt heat of fusion data input has to be done in the same mixture line where the `solub`, `salt` and `salt_n` commands are located. The net Gibbs free energy of fusion of the salt can be given by options `DGfus_salt` or `DGfus_salt_SI`. The net enthalpy of fusion of the salt as used in eq. 2.3.4-3 can be given by options `DHfus_salt` or `DHfus_salt_SI`. The net entropy of fusion of the salt as used in eq. 2.3.4-3 can be given by options `DSfus_salt` or `DSfus_SALT_SI`. The net heat capacity of fusion of the salt, as used in eq. 2.3.4-3, can be given by options `Dcpfus_salt` or `Dcpfus_SALT_SI`. The net melting temperature of the salt, as used in eq. 2.3.4-3, can be given by options `Tmelt_salt` , `Tmelt_salt_C` or `Tmelt_salt_K`. (see below for details). All of these input options expect "net" salt data values that are defined for the salt as defined in composition and stoichiometry by the `salt` and `salt_n` commands. Thus no individual ion heat of fusion data can be used for salts. The ΔCp_{fus} estimate described above, for neutral compounds solubility (toggled by the keyword **`Dcpfus_estimate`**), may also be used in combination with salts. However, following the considerations taken above, the ΔCp_{fus} estimate should only be used with great caution when it comes to the computation of salt solubilities.

---

[36] "Prediction of Solubility with COSMO-RS", Frank Eckert, in *Developments and Applications in Solubility*, Trevor Letcher, (Ed.), The Royal Society of Chemistry, UK (2007).

Please note the ambiguity in the definition and use of given free energy of fusion data, described above: by default the definition 2.3.4-10 is used, which means that the free energy of fusion computed from experimental heat of fusion data or given explicitly in the input will be used "as is" in equation 2.3.4-10. If the keyword `dgfmean` is given in the same line as the salt solubility and salt heat of fusion data input, the alternative definition of the mean ionic free energy of fusion as given in equation 2.3.4-11 is used to compute the salt solubility.

Please note that for salt solubilities, similar to considerations taken on ionic liquids systems, some care has to be taken in the interpretation of the COSMO*therm* results for the computed salt solubility mole fractions x_{AC}^{SOL}. To be able to compare the computed salt mole fraction with experimental data, depending on the reference state of the salt solubility measurement, it may be necessary to convert the computed salt mole fraction value along the guidelines given in section 4.9 of this manual. COSMO*therm* by default will convert the mole fraction based salt solubility results from the computed multicomponent framework (where the salt is considered to consist of independent anion and cation components) to the "laboratory-binary" or "IL-binary" framework (see section 4.9), where the salt is considered to be one single compound. Because experimental solubility measurements of salt solubilities typically are done in the laboratory-binary framework, the multicomponent mole fraction soluilities are converted to the laboratory-binary framework by COSMO*therm* by default and printed to the salt solubility table accordingly. It is possible to print the multicomponent "pseudo-binary" or "IL-ternary" mole fraction of the salt (where x_{AC}^{SOL} is defined as the stoichiometric sum of the ion mole fractions $x_{AC}^{SOL} = x_{A,}+ x_C$) to the salt solubility table using the keyword `pr_ILTERN` in the salt solubility mixture input line. Both IL-binary and IL-ternary definition of the salt solubility mole fractions will be printed to the by COSMO*therm* output file.

In addition to the mole fraction output, the salt solubility option will print the mass based solubility w_{AC}^{SOL} [$g_{AC}/g_{solution}$] to the output and table file. There are three possible definitions of the mass based solubility, where MW_{AC} and $MW_{Solvent}$ are the molecular weights of the salt solute and the solvent (or solvent mixture):

- By default (and/or if the keyword **`wfract`** is given in addition to the `solub` keyword), the *normalized mass fraction* solubility is computed: $w_{AC}^{SOL} = w_{AC}^{FRACT} = x_{AC}^{SOL} * MW_{AC} / (x_{AC}^{SOL} * MW_{AC} + (1- x_{AC}^{SOL}) * MW_{Solvent})$.
- If the keyword **`wsol2`** is given in addition to the `solub` keyword), the *unnormalized* mass based solubility is computed by "Definition 2": $w_{AC}^{SOL} = w_{AC}^{SOL2} = x_{AC}^{SOL} * MW_{AC} / ((1- x_{AC}^{SOL}) * MW_{Solvent})$.
- If the keyword **`wsol1`** is given in addition to the `solub` keyword, the *unnormalized* mass based solubility is computed by "Definition 1": $w_{AC}^{SOL} = w_{AC}^{SOL1} = x_{AC}^{SOL} * MW_i / MW_{Solvent}$.
  Please note that, although commonly used in pharmaceutical chemistry, definition 1 is an approximation derived for small solubilities. Thus it should be used only if the solubility is expected to be small ($\log_{10}(x_{AC}^{SOL}) < -3$).

Please note that if, at the given conditions, a compound is miscible with the solvent (i.e. if the logarithm of the mole fraction solubility is zero) the mass based solubility is not well-defined and thus it will not be printed to output and table file. If the density of the solvent and solute is known, or can be estimated, the decadic logarithm of the *molar* solubility of the salt in the *solution* ($\log_{10}(S_{AC}$ [mol/l]$) = \log_{10}(x_{AC}^{SOL} / V_{Solution})$) will be written to an additional column in the COSMO*therm* table file. The molar volume of the solution $V_{Solution} = \rho_{Solution} / MW_{Solution}$ is calculated from the molar masses and densities of the given solvent $\rho_{Solvent}$ and salt ρ_{AC} according to the salt solubility x_{AC}^{SOL} computed. The density of the solvent (or solvent mixture) $\rho_{Solvent}$ can be provided to COSMO*therm* via the the `solvdens` keyword. If $\rho_{Solvent}$ is not given in the input, COSMOtherm will try to estimate the solvents density with the liquid density/volume QSPR method as described in section 2.3.10. If no density estimate is available, the molar solubility of the solute in the solvent will not be computed. The density of the salt ρ_{Salt} is estimated from the molecular COSMO-volume of the individual ion components of the salt. This estimate, which is somewhat poorer than the liquid density/volume QSPR, is used because the liquid density/volume QSPR is not applicable to solid salt compounds. If the **wsol1** keyword is given in addition to the solub keyword, the decadic logarithm of the *molar* solubility of the solute in the *solvent* ($\log_{10}(S_{AC}$ [mol/l]$) = \log_{10}(x_{AC}^{SOL} / V_{Solvent})$) will be written to the COSMO*therm* table file.

Salt/Ionic Liquid solubility computation suboptions to the solub or nsolub command:

salt={i j …} or nsalt={name$_i$ name$_j$ …}	Optional for solub computations: Define the salt composition from individual anion and cation compounds. The input of the salt composition is possible either via their compound number (salt={i j …} command, where i, j, … are the numbers of the ion compounds numbers as given in the sequence of compounds in the compound input section. The arguments i, j, … are expected to be positive integer numbers) or via their compound name (nsalt={name$_i$ name$_j$ …} command, where name$_i$, name$_j$, … are the names of the ion compounds as given in the compound input section).
salt_n={ν_i ν_j …}	Optional for solub computations: Define the salt stoichiometry from individual anion and cation compounds. The input of the salt stoichiometry factors ν_i, ν_j is possible via the salt_n={ν_i ν_j …} command, where ν_i, ν_j, … are the stoichiometry numbers of the salt compound defined by the ions as given by the salt={i j …} or nsalt={name$_i$ name$_j$ …} command. The stoichiometry factors ν_i, ν_j,… are expected to be positive integer numbers.
wfract	Optional for solub computations: Change the output of the mass based solubility w_{AC}^{SOL} to normalized mass fraction output, which is $w_{AC}^{FRACT} = x_{AC}^{SOL} *MW_i / (x_{AC}^{SOL} *MW_{AC} + (1-x_{AC}^{SOL})*MW_{Solvent})$.
wsol1	Optional for solub computations: Change the output of the mass based solubility w_j^{SOL} to "Definition 1" which is $w_{AC}^{SOL1} = x_{AC}^{SOL} *MW_{AC} / MW_{Solvent}$.
wsol2	Optional for solub computations: Change the output of the mass based solubility w_{AC}^{SOL} to "Definition 2" which is $w_{AC}^{SOL2} = x_{AC}^{SOL} *MW_{AC} / ((1-x_i^{SOL})*MW_{Solvent})$.
pr_ILtern	Optional for solub computations of salts: Change the output of the mole fraction solubility x_{AC}^{SOL} to "pseudo-binary" framework, where the salts anions and cations are considered to be independent species (default output of x_{AC}^{SOL} is the "laboratory-binary" framework, where the salt is considered to be one single compound, see section 4.9 for details of the conversion)

Salt heat of fusion / crystallization input:

`DGfus_salt=value`
or
`DGfus_salt_SI=value`
	Optional: Give the free enthalpy of fusion ΔG_{fus} for a salt. For the `DGfus_salt=value` option, ΔG_{fus} is expected in [kcal/mol], for the `DGfus_salt_SI` option, ΔG_{fus} is expected to be in [kJ/mol]. Argument `value` is expected to be a real number.

`DHfus_salt=value`
or
`DHfus_salt_SI=value`
	Optional: Give the enthalpy of fusion ΔH_{fus} for a salt. For the `DHfus_salt=value` option, ΔH_{fus} is expected in [kcal/mol], for the `DHfus_salt_SI` option, ΔH_{fus} is expected to be in [kJ/mol]. Argument `value` is expected to be a real number.

`DSfus_salt=value`
or
`DSfus_salt_SI=value`
	Optional: Give the entropy of fusion ΔS_{fus} for a salt. For the `DSfus_salt=value` option, ΔS_{fus} is expected in [kcal/mol], for the `DSfus_salt_SI` option, ΔS_{fus} is expected to be in [kJ/mol]. Argument `value` is expected to be a real number.

`Dcpfus_salt=value`
or
`Dcpfus_salt_SI=value`
	Optional: Give the heat capacity of fusion ΔCp_{fus} for a salt. For the `Dcpfus_salt=value` option, ΔCp_{fus} is expected in [kcal/mol], for the `Dcpfus_salt_SI` option, ΔCp_{fus} is expected to be in [kJ/mol]. Argument `value` is expected to be a real number.

`Tmelt_salt=temp`
or
`Tmelt_salt_C=temp`
or
`Tmelt_salt_K=temp`
	Optional: Give the melting temperature T_{melt} for a salt. For the `Tmelt_salt=temp` and `Tmelt_salt_C=temp` options, T_{melt} is expected in [°C], for the `Tmelt_salt_K=temp` option, T_{melt} is expected in [K]. Argument `temp` is expected to be a real number.

`dcpfus_estimate`	Optional for the input of a temperature dependent salt free energy of fusion (via input of enthalpy or entropy of fusion ΔH_{fus} or ΔS_{fus}, and melting point T_{melt}): toggle the approximation of the heat capacity of fusion as $\Delta Cp_{fus} = \Delta S_{fus} = \Delta H_{fus}/T_{melt}$. The value of $\Delta G_{fus}(T)$ thus obtained wil be used to compute the solubility of the salt compound. If the `dcpfus_estimate` keyword is given in a salt solubility computation input line, the approximation to ΔCp_{fus} is valid for the given salt and the given mixture line.
`DGfmean`	Optional: use alternative "mean" ionic free energy of fusion definition for the given ΔG_{fus} of a salt. If the `DGfmean` option is given, equation 2.3.4-11 will be used to calculate the salt solubility instead of equation 2.3.4-10. i.e. the given ΔG_{fus} data is not scaled by the salts overall stoichiometry number.

In addition to the input of salt heat of fusion / crystallization data in terms of experimental melting point and free energy, enthalpy, entropy, or heat capacity of fusion data, it is possible to compute a salts Gibbs free energy of fusion (ΔG^{Salt}_{fus} or ΔG^{AC}_{fus} as defined in eqs. 2.3.4-10 and 2.3.4-11) from a given *reference solubility* value for the salt. The input and processing of reference solubility is equivalent to the reference solubility input of the neutral compound solubility computation option described above. Please note however that there are two complications if the solute is a salt compound consisting of several individual ion components: First, the input of reference mole fraction solubilities depends on the salts mole fraction definition as described above and in section 4.9. By default COSMO*therm* assumes that the given reference mole fraction of the salt is given in the "laboratory-binary" or "IL-binary" framework, where the salt is considered to be one single compound. This value is converted to COSMO*therm*'s internal multicomponent framework, where the salt is considered to consist of independent anion and cation components. If the keyword `pr_ILTERN` is given in the salt solubility mixture input line, it is assumed that the reference mole fraction of the salt is given in the multicomponent "pseudo-binary" or "IL-ternary" framework, where x_{AC}^{SOL} is defined as the stoichiometric sum of the ion mole fractions $x_{AC}^{SOL} = x_{A,}+ x_{C}$. Second, the Gibbs free energy of fusion as computed by the reference solubility option by default is assumed to be ΔG^{Salt}_{fus} as defined in equation 2.3.4-10. I.e. by default equation 2.3.4-10 will be used to compute ΔG^{Salt}_{fus} from the reference solubility.If the keyword `DGfmean` is given in the same line as the reference salt solubility input, the alternative definition of the mean ionic free energy of fusion ΔG^{AC}_{fus} will be computed from equation 2.3.4-11.

Reference salt solubility input:

`solute=j`
or
`nsolute=name`$_j$
or
`nsolute=SALT`
	Optional for `solub` computations: Define the solute compound for a reference solubility calculation of ΔG_{fus}. The `solute=j` option computes defines the given salt as reference solute compound, where `j` may be the compound number of any of the salt components in the collating sequence of the compound input section. The `nsolute=name`$_j$ or `nsolute=SALT` options define the given salt as reference compound for the salt solubility computation, where `name`$_j$. may be the name of any of the salt components.

Reference salt solubility input (continued):

`ref_sol_s=value`
or
`ref_sol_x=value`
or
`ref_sol_c=value`
or
`ref_sol_g=value`
or
`ref_sol_m=value`
or
`ref_sol_l=value`
	Optional for `solub` computations: Define the reference solubility value for a reference solubility calculation of ΔG_{fus}. The `ref_sol_s` option assumes that the given reference solubility `value` is given as the decadic logarithm of the solutes mole fraction $\log_{10}(x^{REF}_{SOL})$. The `ref_sol_x` option assumes that the given reference solubility `value` is given as the solutes mole fraction x^{REF}_{SOL}. The `ref_sol_c` option assumes that the given reference solubility `value` is given as the solutes mass fraction concentration c^{REF}_{SOL}. The `ref_sol_g` option assumes that the given reference solubility `value` is given as the solutes mass based solubility g^{REF}_{SOL} in [g/g]. By default, (or if the **`wsol2`** keyword is given), the input of the mass based solubility g^{REF}_{SOL} is assumed to be the unnormalized mass based solubility of "*Definition 2*": $g^{REF}_{SOL} = w^{REF}_{SOL2} = x_i^{SOL} * MW_i / ((1-x_i^{SOL}) * MW_{Solvent})$. If the additional keyword **`wsol1`** is given, the input as well as the output of the mass based solubility g^{REF}_{SOL} is assumed to be the unnormalized mass based solubility of "*Definition 1*": $g^{REF}_{SOL} = w^{REF}_{SOL1} = x_i^{SOL} * MW_i / MW_{Solvent}$. If the additional keyword **`wfract`** is given, the input as well as the output of the mass based solubility g^{REF}_{SOL} is assumed to be the mass fraction (i.e. identical to the input of the `ref_sol_c` option). The `ref_sol_m` option assumes that the given reference solubility `value` is given as the solutes molar concentration M^{REF}_{SOL} in [mol/l]. Please note that the input of a molar reference solubility requires the input of the solvent density (**`solvdens`** option, above). If no solvent density is given, the conversion of the molar solubility to mole fraction can only be done in an approximative way. The `ref_sol_l` option assumes that the given reference solubility `value` is given as the solutes mass/volume concentration L^{REF}_{SOL} in [g/l]. Please note that the mass/volume concentration units also requires the input of the solvent density. Along the lines of the mass based solubility input, the mass/volume concentration input assumes that L^{REF}_{SOL} is given by mass "*Definition 2*" as noted above, unless the `wsol1` or `wfract` keywords are used, in which case the input of L^{REF}_{SOL} is assumed to be a "*Definition 1*" mass solubility, or a mass fraction, respectively.

2.3.5 Partition Coefficients

The **`logp={i1 i2}`** or **`nlogp={name1 name2}`** option allows for the automatic computation of partition coefficients between compounds `i1` and `i2`. By default, this option will compute the chemical potentials $\mu_j^{(1)}$ of all compounds `j` in infinite dilution in pure compound `i1` and subsequently the chemical potentials $\mu_j^{(2)}$ at infinite dilution in pure compound `i2`. The partition coefficients are then calculated as

$$\log_{10}\left(P_j^{(2,1)}\right) = \left(\mu_j^{(1)} - \mu_j^{(2)}\right) / RT \ln(10) + \log_{10}(VQ) \tag{2.3.5-1}$$

and written to the COSMO*therm* output file and to the COSMO*therm* table file. By default, the solvent phase volume quotient $VQ = V_1 / V_2$ will be estimated from solvent densities ρ_1 and ρ_2. If VQ is not given in the input, COSMOtherm will try estimate the solvents volumes with the liquid density/volume QSPR method as described in section 2.3.10. Please note that for solvent mixtures the density estimate is a linear interpolation between the pure compounds densities, where excess density/volume is neglected. If no density estimate is available, the solvent phase volume quotient will be estimated from the COSMO-Volumes of the given solvent compounds. However, it is also possible to read VQ from the input file using the `vq=value` command (see below). The input of VQ will be necessary if the densities of the two solvent phases differ substantially and thus the estimate from the QSPR densities or COSMO volumes (based on the assumption of an incompressible liquid) will be poor.

In addition, it is possible to give finite concentrations at which the chemical potentials $\mu_j^{(i)}$ of the compounds are computed via the `xl1={x1 x2 …}` and `cl1={c1 c2 …}` commands (for the first solvent) and `xl2={x1 x2 …}` and `cl2={c1 c2 …}` commands (for the second solvent). This will be necessary if the two solvent phases are partially soluble in each other (e.g. as for the case of octanol – water, where the octanol-rich phase contains about 0.274 mole fractions of water[37]). If the phase concentration is not known in advance it is possible to compute the equilibrated phases with the suboption `eq_phases`. This option toggels an upfront phase equilibration calculation using the multi-component - multi-phase equilibrium technique described in section 2.3.12. The pure phases given as arguments of the `logp` command, or the phase concentration input via options `xl1`, `cl1`, `xl2`, and `cl2` will be used as starting concentrations for the phase equilibration. If the `eq_phases` keyword is given, both the computation of the chemical potentials that enter the partition coefficient calculation, and the computation of the volume quotient VQ (if not given by the `vq=value` command) will be done in the phases thus equilibrated.

---

[37] Dallos, A.; Liszi, J. *J. Chem. Thermodynamics*, **27** 447 (1995)

Suboptions of the `logp` or `nlogp` command are:

`xl1={x1 x2 …}`
or
`cl1={c1 c2 …}`
and/or
`xl2={x1 x2 …}`
or
`cl2={c1 c2 …}`
	Optional for `logp` computations: Give finite mixture concentrations for the two phases between which the partition coefficient shall be computed. The input of the concentrations is possible either in mole fractions (`xl1={}`, `xl2={}`) or mass fractions (`cl1={}`, `cl2={}`) of the compounds of the mixture (as real numbers `xi` and `ci`). The arguments are expected as real numbers between zero and one in the same sequence of compounds as given in the second input area. If the values do not add up to one, COSMO*therm* will normalize them. If less mole fractions / concentrations than compounds are given, the missing ones will be assumed zero. If a negative number is given, the concentration for this compound will be inserted automatically using the normalization of the sum of mole fractions. Only one negative number is allowed per mixture input line.

`vq=value`	Optional for `logp` computations: Give the volume quotient $VQ=V_1/V_2$ between the two phases for which the partition coefficient shall be computed. The argument is expected as a non-negative non-zero real number[38].
`eq_phases`	Optional for `logp` computations: Compute the phase equilibrium between the given two phases in which the partition coefficient shall be computed.

Note: If a finite concentration input via options `xl1` or `cl1` and `xl2` or `cl2` is given for both solvent phases, no arguments need to be given to the `logp` or `nlogp` option.

---

[38] The recommended value for the volume quotient of the "wet" 1-octanol – water system with 0.274 mole fractions of water in the octanol-rich phase is `VQ=0.1505` (cf. ref. 37).

Dissociation Corrections to Partition Coefficients: Distribution Coefficient logD

In partition coefficient calculations of systems with one aqueous phase (most prominently for the 1-octanol - water system $\log P_{OW}$) it is possible that acidic or basic solutes dissociate in the water phase. The dissociated species typically have partition properties different from the partition property of the undissociated compound, thus affecting the apparent partition coefficient of the compound[39]. The effect of the dissociated compound approximatively can be taken into account by applying a dissociation correction to the partition coefficient. Following the assumption that the dissociated solute species will not migrate into the organic solvent phase, the fraction of the dissociated solute species in the aqueous phase solely is determined by the solutes aqueous dissociation constant pK_a and the pH of the aqueous phase. The resulting dissociation corrected partition coefficient is denoted as *distribution coefficient* **log*D***. If the solute is an acid HA showing the dissociation reaction $HA + H_2O \rightleftharpoons A^- + H_3O^+$ the correction term to partition coefficient logP can be approximated by equation 2.3.5-2 employing the acids dissociation constant **pK_a(acid)** and the **pH** of the aqueous phase:

$$\log D = \log P - \log\left(1 + 10^{(pH - pK_a(acid))}\right) \tag{2.3.5-2}$$

If the solute is an base B showing the dissociation reaction $B + H_2O \rightleftharpoons BH^+ + OH^-$ the correction term to partition coefficient logP can be approximated by equation 2.3.5-3 employing the base dissociation constant **pK_a(base)** and the **pH** of the aqueous phase:

$$\log D = \log P - \log\left(1 + 10^{(pK_a(base) - pH)}\right) \tag{2.3.5-3}$$

If at the given pH value, a solute is able to form a **zwitterionic structure**, the situation is somewhat more complicated, and the simple correction terms of eqs. 2.3.5-2 and 2.3.5-3 should be applied with caution. A solute S that contains acidic (AH) as well as basic (B) reactive sites can undergo the following four dissociation and protonation reactions in protic solvent water:

$S(AH)(B) + H_2O \rightarrow S(A^-)(B) + H_3O^+$	dissociation of neutral acid to anion with pK_a(AH)
$S(AH)(B) + H_2O \rightarrow S(A)(BH^+) + OH^-$	protonation of neutral base to cation with pK_a(B)
$S(AH)(BH^+) + H_2O \rightarrow S(A^-)(BH^+) + H_3O^+$	dissociation of cation to zwitterion with pK_a(BH$^+$)
$S(A^-)(B) + H_2O \rightarrow S(A^-)(BH^+) + OH^-$	protonation of anion to zwitterion with pK_a(A$^-$)

The dissociation correction to the partition coefficient now has to be determined from the chemical equilibrium between the four species of S that are possible in the solution, namely neutral form S(AH)(B), zwitterion $S(A^-)(BH^+)$, anion $S(A^-)(B)$, and cation $S(AH)(BH^+)$. Note that for the zwitterion formation reactions that start from ionic species, the appropriate pKa value that has to be used is the one of the inverse reaction. I.e. for the deprotonation of a cation to form a zwitterion, the base pK_a of species BH^+ has to be applied, and for the protonation of an anion to form the zwitterion, the acid pK_a of species A^- has to be applied, respectively. Now using the definitions $pH = -\log([H_3O^+])$, $pOH = -\log([OH^-]) = 14 - pH$, and $pK_b = 14 - pK_a$, and further assuming that the concentration of the solvent water is significantly larger than that of the solutes so that [H2O] ~ 1 we can define the equilibrium constants of the four reactions as:

---

[39] Leo A.; Hansch C.; Elkins D. *Chem. Rev.* **71** 525 (1971).

$S(AH)(B) + H_2O \rightarrow S(A^-)(B) + H_3O^+ \quad K_1 = [\ S(A^-)(B)\ ]/[\ S(AH)(B)\ ] \quad = 10^\wedge(-\mathrm{p}K_a(AH)+\mathrm{p}H)$
$S(AH)(B) + H_2O \rightarrow S(A)(BH^+) + OH^- \quad K_2 = [\ S(A)(BH^+)\ ]/[\ S(AH)(B)\ ] \quad = 10^\wedge(-\mathrm{p}K_b(B)+\mathrm{p}OH)$
$S(AH)(BH^+) + H_2O \rightarrow S(A^-)(BH^+) + H_3O^+ \quad K_3 = [\ S(A^-)(B)\ ]/[\ S(A^-)(BH^+)\ ] \quad = 10^\wedge(-\mathrm{p}K_b(BH^+)+\mathrm{p}OH)$
$S(A^-)(B) + H_2O \rightarrow S(A^-)(BH^+) + OH^- \quad K_4 = [\ S(A)(BH^+)\ ]/[\ S(A^-)(BH^+)\ ] \quad = 10^\wedge(-\mathrm{p}K_a(A^-)+\mathrm{p}H)$

The reaction free energy differences of the species can be determined from their equilibrium constants:

$S(AH)(B) + H_2O \rightarrow S(A^-)(B) + H_3O^+ \quad \Delta G_1 = -RT\ \ln(K_1)$
$S(AH)(B) + H_2O \rightarrow S(A)(BH^+) + OH^- \quad \Delta G_2 = -RT\ \ln(K_2)$
$S(AH)(BH^+) + H_2O \rightarrow S(A^-)(BH^+) + H_3O^+ \quad \Delta G_3 = -RT\ \ln(K_3)$
$S(A^-)(B) + H_2O \rightarrow S(A^-)(BH^+) + OH^- \quad \Delta G_4 = -RT\ \ln(K_4)$

The free energies of realization ΔG^r_1 of the four species (neutral, zwitterion, anion, cation) with respect to the neutral solute as reference state are:

neutral solute S(AH)(B) realization (reference state) $\Delta G^r_1 = 0$
anion $S(A^-)(B)$ realization $\Delta G^r_2 = \Delta G_1$
cation $S(AH)(BH^+)$ realization $\Delta G^r_3 = \Delta G_2$
zwitterion $S(A^-)(BH^+)$ realization $\Delta G^r_4 = ((\Delta G_2 - \Delta G_3)+(\Delta G_1 - \Delta G_4))/2$

Note that the free energy of realization of the zwitterion is approximated as the mean value of the two reaction pathways that can form the zwitterion (i.e. via anion or from the cation). The total free energy of realization of all possible species of the reacting system is determined by the partition sum of the reaction free energies:

$$\Delta G^r_S = -RT\ln(Z) \quad where \quad Z = \sum_i e^{\left(\Delta G^r_i / RT\right)} \tag{2.3.5-4}$$

The total free energy of realization ΔG^r_S thus computed corresponds to the free energy correction that results from the formation of the four different species in the solution based on the four dissociation and protonation reactions sketched above. This leads to a generalized form of the specific dissociation corrections of eqs. 2.5.3-2 and 2.5.3-3 as given above. The generalized expression for the distribution coefficient of systems that can form zwitterions now reads:

$$\log D = \log P - \log\left(1 + e^{\left(\Delta G^r_S / RT\right)}\right) \tag{2.3.5-5}$$

Note, that the general formulation of the dissociation correction to partition in equation 2.3.5-5, which is shown here for the special case of zwitterions, can be applied to many other possible dissociation or protonation scenarios as well, e.g. to solutes with multiple protonation or deprotonation levels, or to solutes with several competing protonation or deprotonation sites with similar pK_a. Also note that the general formulation of equation 2.3.5-5 typically requires the estimation of the different pK_a values involved in the different reactions. Experimental pK_a data cannot be applied in this case, because for systems with zwitterions or even more complex systems, typically only one "apparent" pK_a can be measured per dissociation stage while the "microscopic" pK_a values of the different dissociation and protonation reactions that are possible are unknown and can only be estimated or computed by COSMO*therm* (see below).

The automatic partition coefficient prediction (`logp`) option of COSMO*therm* will apply the dissociation correction, to any solute of which the pK_a value is given or estimated. The log$D$ value thus computed will be written to the COSMO*therm* output file as well as to an additional column in the table output of the `logp` option in the COSMO*therm* table file. In addition, the pK_a(acid) or pK_a(base) value used in the log*D* calculation is printed to the output and table files. A solutes aqueous dissociation constant pK_a(acid) or pK_a(base) value may either be given to COSMO*therm* as additional input in the vap-file, compound input line or logp option mixture input line, or, it may be estimated by COSMO*therm* using the pK_a prediction methodology described in section 2.3.6 of this manual.

There are three possibilities to enter a compounds aqueous dissociation constant pK_a(acid) or pK_a(base):

- specify the pK_a value in the compounds vapor pressure property (**`.vap`** file) using the **`pK_acid=value`**, or **`pK_base=value`** option (see section 2.2).
- specify the pK_a value in the compound input line using the **`pK_acid=value`**, or **`pK_base=value`** option (see section 2.2).
- specify the solute compound and the pK_a value in the `logp` option mixture input line using the **pKacid={$\mathbf{i_{Solute}}$ value}**, or **pKbase={$\mathbf{i_{Solute}}$ value}** options, where **$\mathbf{i_{Solute}}$** is the number of the solute compound as given in the collating sequence of compounds in the compound input (see below).

In addition to the use of experimental aqueous dissociation constants, COSMO*therm* is able to provide an estimate of a solutes pK_a(acid) or pK_a(base) using the Linear Free Energy Relationship (pK_a-LFER) decribed in section 2.3.6. In the framework of partition coefficient calculations, the pK_a estimate for acidic and basic solutes is toggled by the **comp_acid={$\mathbf{i_{Neutral}}$ $\mathbf{i_{Anion}}$}**, and **comp_base={$\mathbf{i_{Neutral}}$ $\mathbf{i_{Cation}}$}** options, given in the `logp` option mixture input line. Therein **$\mathbf{i_{Neutral}}$** is the number of the solute compound as given in the collating sequence of compounds in the compound input, and **$\mathbf{i_{Anion}}$** and **$\mathbf{i_{Cation}}$** are the compound numbers of the anion and cation compounds that are formed on acidic and basic dissociation of the solute compound, respectively. Please note that the pK_a-LFER estimate of pK_a(acid) or pK_a(base) values requires the presence of COSMO files of the dissociated species A^- or BH^+, which form from solute via the acidic or basic dissociation reaction. For more details on the pK_a-LFER methodology, please see section 2.3.6.

The p*H* of the aqueous phase is assumed to be `7.0`. This value can be changed by the input option **`pH=value`**. Please note that for measurements of distribution coefficient, the pH of the aqueous phase is buffered to a specific value, most often to a value of `pH=7.4`, which is the p*H* of blood serum and thus of significance in physiological partition.

Suboptions of the `logp` or `nlogp` option are:

`pKacid=`
{i_{Solute} `value`}
or
`pKbase=`
{i_{Solute} `value`}
	Optional for `logp` computations: Input of experimental aqueous dissociation constants for acidic (`pKacid` option) or basic (`pKbase` option) solutes for a log*D* distribution coefficient computation. Argument i_{Solute} is the compound number of the solute. It is expected to be an integer number designating a compound in the sequence of the compound input. Argument `value` is the pK_a(acid) (`pKacid` option) or pK_a(base) (`pKbase` option) value for solute i_{Solute}. It is expected to be a real number. It is possible to give up to n_{comp} (total number of compounds) `pKacid` and `pKbase` input fields in one `logp` mixture input line.

`comp_acid=`
{$i_{Neutral}$ i_{Anion}}
or
`comp_base=`
{$i_{Neutral}$ i_{Cation}}
	Optional for `logp` computations: Input of experimental aqueous dissociation constants for acidic (`pKacid` option) or basic (`pKbase` option) solutes for a log*D* distribution coefficient computation. Argument $i_{Neutral}$ is the compound number of the solute. It is expected to be an integer number designating a compound in the sequence of the compound input. Argument i_{Anion} (for the `pKacid` option) or i_{Cation} (for the `pKbase` option) are the compound numbers of the ionic compounds that result from the acidic or basic dissociation reaction of the given solute $i_{Neutral}$. It is possible to give up to n_{comp} (total number of compounds) `comp_acid` and `comp_base` input fields in one `logp` mixture input line.

`pH=value`	Optional for `logp` computations: Input of the logarithmic acidity/basicity (p*H)* of the water phase for a log*D* distribution coefficient computation. Argument `value` is the p*H* value of the water solvent phase. It is expected to be a real number. If no `pH=value` is given, it is assumed that the aqueous phase is neutral (p*H*=7).

2.3.6 Acidity and Basicity pK_a

The **pKa={i$_{\mathrm{Solvent}}$ i$_{\mathrm{Neutral}}$ i$_{\mathrm{Ion}}$}** or **npKa={name$_{\mathrm{Solvent}}$ name$_{\mathrm{Neutral}}$ name$_{\mathrm{Ion}}$}** option allows for the automatic computation of the pK_a value of acidity / basicity for a compound in a solvent i$_{\mathrm{Solvent}}$. (where i$_{\mathrm{Solvent}}$ normally is water). By default, this option will compute the free energy $G_{\mathrm{Neutral}}^{(i)}$ of compound i$_{\mathrm{Neutral}}$ and the free energy$G_{\mathrm{Ion}}^{(i)}$ of compound i$_{\mathrm{Ion}}$ in the solvent i$_{\mathrm{Solvent}}$ at infinite dilution of i$_{\mathrm{Neutral}}$ and i$_{\mathrm{Ion}}$. Subsequently, the pK_a value is estimated from the linear free energy relationship (LFER):

$$pK_A^{(i)} = c_0 + c_1\left(\Delta G_{Neutral}^{(i)} - \Delta G_{Ion}^{(i)}\right) \tag{2.3.6-1}$$

The pK_a value is written to the COSMO*therm* output file and to the COSMO*therm* table file. Thus, to obtain a pK_a value it is necessary to do a quantum chemical COSMO calculation of a molecule in its neutral state and as an ion. The neutral and the ionic compounds both have to be given in the input of the pK_a option as i$_{\mathrm{Neutral}}$ and i$_{\mathrm{Ion}}$. It is also possible to estimate the pK_a value of higher states of ionization. In this case, the neutral and single ionic species i$_{\mathrm{Neutral}}$ and i$_{\mathrm{Ion}}$ have to be replaced by higher ionized species. For example, the first pK_a of phosphoric acid is calculated from the free energy difference of $G(H_3PO_4)$ – $G(H_2PO_3^-)$ while the second pK_a of phosphoric acid results from the free energy difference of $G(H_2PO_3^-)$ – $G(HPO_3^{2-})$ and the third pK_a from the free energy difference of $G(HPO_3^{2-})$ – $G(PO_3^{3-})$. The pK_a -LFER denoted in eq. 2.3.6 does not include a temperature dependency term, which implies that it is valid for the temperature of the LFER fit (i.e. room temperature) only. By default, LFER parameters c_0 and c_1 for acidic solutes in solvent water at room temperature are read from the COSMO*therm* parameter file. However, it is also possible to read the LFER parameters from the COSMO*therm* input file via the pKaLFER={c$_0$ c$_1$} or pKaLFER_SI={c$_0$ c$_1$} commands (see below). The input of of the LFER paramaters will be necessary if solvents other than water or temperatures other than room temperature are used. In addition, it is possible to give finite concentrations of a solvent mixture where the free energies $G_{\mathrm{Neutral}}^{(i)}$ and $G_{\mathrm{Ion}}^{(i)}$ are computed. This is possible via the commands xp={x$_1$ x$_2$ …} or cp={c$_1$ c$_2$ …}.

Suboptions of the `pKa` or `npKa` command are:

xp={x_1 x_2 …} or cp={c_1 c_2 …}	Optional for `pKa` computations: Give finite mixture concentrations for the solvent phase in which the pK_a value shall be computed. The input of the concentrations is possible either in mole fractions (xp={x_1 x_2 …}) or mass fractions (cp={c_1 c_2 …}) of the compounds of the mixture (as real numbers x_i and c_i). The arguments are expected as real numbers between zero and one in the same sequence of compounds as given in the second input area. If the values do not add up to one, COSMO*therm* will normalize them. If less mole fractions / concentrations than compounds are given, the missing ones will be assumed zero. If a negative number is given, the concentration for this compound will be inserted automatically using the normalization of the sum of mole fractions. Only one negative number is allowed per mixture input line.
pKaLFER={c_0 c_1} or pKaLFER_SI={c_0 c_1}	Optional for `pKa` computations: Give the LFER parameters c_0 and c_1 required in the pK_a computation. The arguments c_0 and c_1 are expected as real numbers. If the pKaLFER={c_0 c_1} option is used the LFER parameters are expected in [kcal/mol]. If the pKaLFER_SI={c_0 c_1} option is used the LFER parameters are expected in [kJ/mol].

COSMO*therm* pK_a prediction is not restricted to acid pK_a. It is also possible to compute **aqueous base pK_a**. However, base pK_a prediction requires a reparameterization of the pK_a LFER parameters. The LFER parameters for aqueous base pK_a at room temperature are shipped within COSMO*therm* parameter files `BP_TZVP_C30_1701.ctd` and `BP_SVP_AM1_C30_1701.ctd`. The usage of the aqueous base pK_a LFER parameters is toggled with the **`WATER-BASE`** suboption of the `pKa={`$i_{Solvent}$ $i_{Neutral}$ i_{Ion}`}` command. Alternatively, the `pKaLFER={`c_0 c_1`}` command may be used to give the aqueous base pK_a LFER parameters in the COSMO*therm* input. Note: For secondary and tertiary aliphatic amines COSMO*therm* systematically underestimates the base pK_a. This underestimation is the result of a well known[3,33] problem of continuum solvation models like COSMO with aliphatic amines and amino-cations in polar solvents. Because the error is systematic, it can be accounted for by a simple correction term:

The pK_a prediction of tertiary aliphatic amines should be corrected: pK_a = p$K_a^{predict}$ + 2.0
The pK_a prediction of secondary aliphatic amines should be corrected: pK_a = p$K_a^{predict}$ + 1.0

The given corrections are valid for COSMO*therm* version C2.1 revision 01.07 to 01.11. If you use an older version of COSMO*therm* amine pK_a correction values of 3.8 (for tertiary aliphatic amines) and 1.7 (for secondary aliphatic amines) should be applied. **For COSMO*therm* version C3.0 revision 12.01 and later, no correction is necessary.** Please keep in mind that the systematic amine error only occurs for secondary and tertiary aliphatic amines that are not constrained geometrically. For aromatic amines or aliphatic amines that are sterically hindered (such as tertiary bridge nitrogens in bicyclic rings) the error does not occur and thus the COSMO*therm* pK_a prediction is applicable without correction!

COSMO*therm* pK_a predictions are not restricted to solvent water. It is also possible to compute an acid or base pK_a for nonaqueous solvents. This requires a reparameterization of the pK_a LFER parameters. The pK_a LFER parameters can be passed over to COSMO*therm* with the `pKaLFER={`c_0 c_1`}`. For some common pK_a solvents the LFER parameters are shipped within the COSMO*therm* parameter files and can be used by a simple keyword of the form `SOLVENT-ACID` or `SOLVENT-BASE`. Currently, pK_a LFER parameters for acids in the solvents dimethylsulfoxide and acetonitrile are available from the COSMO*therm* parameter files.

The LFER parameters for acids pK_a at room temperature in solvent **dimethylsulfoxide (DMSO)** are shipped within COSMO*therm* parameter files `BP_TZVP_C30_1701.ctd` and `BP_SVP_AM1_C30_1701.ctd`. The usage of the DMSO acid pK_a LFER parameters is toggled with the **`DMSO-ACID`** suboption of the `pKa={`$i_{Solvent}$ $i_{Neutral}$ i_{Ion}`}` command. Note that the solvent $i_{Solvent}$ or $name_{Solvent}$ that is used in the `pKa` or `npKa` command, is required to be dimethylsulfoxide if the `DMSO-ACID` keyword is used.

The LFER parameters for acids pK_a at room temperature in solvent **acetonitrile (MeCN)** are shipped within COSMO*therm* parameter files `BP_TZVP_C30_1701.ctd` and `BP_SVP_AM1_C30_1701.ctd`. The usage of the acetonitrile acid pK_a LFER parameters[40] is toggled with the **`ACETONITRILE-ACID`** suboption of the `pKa={`$i_{Solvent}$ $i_{Neutral}$ i_{Ion}`}` command. Note that the solvent $i_{Solvent}$ or $name_{Solvent}$ that is used in the `pKa` or `npKa` command, is required to be acetonitrile if the `ACETONITRILE-ACID` keyword is used. The usage of the acetonitrile base pK_a LFER parameters is toggled with the **`ACETONITRILE-BASE`** suboption of the

---

[40] Frank Eckert, Ivo Leito, Ivari Kaljurand, Agnes Kütt, Andreas Klamt, Michael Diedenhofen *Journal of Computational Chemistry* **30**, 799-810 (2009). Please note that the "**`ACETONITRILE-ACID`**" LFER as given in the COSMO*therm* parameterization corresponds to the "complete fit" (eq. 4) of the cited article.

`pKa={`$i_{Solvent}$ $i_{Neutral}$ i_{Ion}`}` command. Note that the solvent $i_{Solvent}$ or $name_{Solvent}$ that is used in the `pKa` or `npKa` command, is required to be acetonitrile if the `ACETONITRILE-BASE` keyword is used.

The LFER parameters for acids pK_a at room temperature in solvent **n-heptane (HEPTANE)** are shipped within COSMO*therm* parameter files `BP_TZVP_C21_0109.ctd` and `BP_SVP_AM1_C21_0109.ctd`. The usage of the n-heptane acid pK_a LFER parameters[41] is toggled with the **`HEPTANE-ACID`** suboption of the `pKa={`$i_{Solvent}$ $i_{Neutral}$ i_{Ion}`}` command. Note that the solvent $i_{Solvent}$ or $name_{Solvent}$ that is used in the `pKa` or `npKa` command, is required to be n-heptane if the `HEPTANE-ACID` keyword is used.

The LFER parameters for base pK_a at room temperature in solvent **tetrahydrofuran (THF)** are shipped within COSMO*therm* parameter files `BP_TZVP_C21_0109.ctd` and `BP_SVP_AM1_C21_0109.ctd`. The usage of the tetrahydrofuran acid pK_a LFER parameters is toggled with the **`THF-BASE`** suboption of the `pKa={`$i_{Solvent}$ $i_{Neutral}$ i_{Ion}`}` command. Note that the solvent $i_{Solvent}$ or $name_{Solvent}$ that is used in the `pKa` or `npKa` command, is required to be n-tetrahydrofuran if the `THF-BASE` keyword is used.

Suboptions of the `pKa` or `npKa` command are:

`WATER-ACID`	Default for `pKa` computations: Use aqueous acid pK_a LFER parameters c_0 and c_1 from the COSMO*therm* parameter file. This is the default setting for `pKa` computations.
`WATER-BASE`	Optional for `pKa` computations: Use aqueous base pK_a LFER parameters c_0 and c_1 from the COSMO*therm* parameter file.
`DMSO-ACID`	Optional for `pKa` computations: Use dimethylsulfoxide (DMSO) acid pK_a LFER parameters c_0 and c_1 from the COSMO*therm* parameter file.
`ACETONITRILE-ACID`	Optional for `pKa` computations: Use acetonitrile (MeCN) acid pK_a LFER parameters c_0 and c_1 from the COSMO*therm* parameter file.
`ACETONITRILE-BASE`	Optional for `pKa` computations: Use acetonitrile (MeCN) base pK_a LFER parameters c_0 and c_1 from the COSMO*therm* parameter file.
`HEPTANE-ACID`	Optional for `pKa` computations: Use n-heptane (HEPTANE) acid pK_a LFER parameters c_0 and c_1 from the COSMO*therm* parameter file.
`THF-BASE`	Optional for `pKa` computations: Use tetrahydrofuran (THF) base pK_a LFER parameters c_0 and c_1 from the COSMO*therm* parameter file.

---

[41] Frank Eckert, Ivo Leito, *unpublished results*.

2.3.7 Phase Diagrams

2.3.7.1 Phase Diagrams: Binary and Ternary Mixtures

The **`binary`** and **`ternary`** options allows for the automatic computation of phase diagrams of two- and three-component mixtures, respectively. Phase diagrams of higher dimensionsonality can be computed with the **`multinary`** phase diagram option, which is described in section 2.3.7.7, below. If one of the `binary`, `ternary`, or `multinary` options is applied, no mole (`x={}`), or mass (`c={}`) fraction input is required. Instead, the program automatically computes a list of concentrations covering the whole range of possible mole fractions of the binary or ternary mixture. Then, for a given temperature, COSMO*therm* automatically calculates the excess properties, the phase diagram as well as other useful information like azeotropic points.

The total pressures used in the computation of a phase diagram are obtained from

$$p_{tot} = \sum_i p_i^0 x_i \gamma_i \tag{2.3.7-1}$$

The p_i^0 are the pure compound vapor pressures for compounds *i* (*i*=1,2 for binary and *i*=1,2,3 for ternary computations, respectively). x_i are the mole fractions of the compounds in the liquid, and γ_i are the activity coefficients of the compounds as predicted by COSMO*therm*. Ideal behaviour of the gas phase is assumed. Vapor mole fractions y_i are obtained from the ratio of partial and total vapor pressures:

$$y_i = p_i^0 x_i \gamma_i / p_{tot} \tag{2.3.7-2}$$

Thus, the computation of phase diagrams requires the knowledge of the vapor pressures of the pure compounds p_i^0 at a given temperature. For each compound, there are several possibilities to calculate or approximate this property.

In order of increasing accuracy you might

- Use the COSMO*therm* approximation of the vapor pressure, using the approximated gas phase energy of the compound. This is the default and requires no additional input.
- Use the COSMO*therm* approximation of the vapor pressure, using the exact gas phase energy of the compound, given via the `e=energy` or `ef=filename` command in the compound input and options section of the input file
- Use COSMO*therm* approximation of the vapor pressure in combination with an experimental boiling point value (either normal boiling point T_{Boil} at 1 atm or a reference boiling temperature T_{ref} at a given reference pressure p_{ref}) via the via the `use_tboil` option.
- Use a vapor pressure correlation equation such as Wagner, DIPPR or Antoine equation to compute the vapor pressure at the given temperature. The vapor pressure equation coefficients either can be given directly in the compound input section of the input file, they can be read from a vapor pressure / property file (vap-file), or they can be calculated from three temperature/vapor pressure pairs given via the `vpexp` command (see section 2.2.3).
- Give the exact value of the vapor pressure for this temperature via the `vpinp` command in the compound input and options section of the input file.

The given sequence of the vapor pressure estimates also indicates the hierarchy in which the vapor pressure estimates are processed in COSMO*therm*. I.e. if several options are used simultaneously, a `vpinp` input value for the vapor pressure will override any vapor pressure equation data given, which itself overrides the `use_tboil` option, which overrides COSMO*therm*'s native vapor pressure estimates.

By default the mole fractions of the compounds in the gas phase y_i are written to the output and the table file for each mixture. The computed partial pressures of compounds $p_i = y_i / p_{tot}$ by default are written to the output file only. The additional keyword **`pr_pp`** (given in the same line as the `binary`, `ternary` or `multinary` option), toggles the printing of the partial pressures of compounds p_i to the COSMO*therm* table file, replacing the entries for mole fractions of the compounds in the gas phase y_i.

By default the excess Gibbs free energy G^E and the excess enthalpy, H^E are written to the output and the table file for each mixture. The additional keyword **`HE_SPLIT`** (given in the same line as the binary or ternary command), toggles the printing of the three contributions to the total excess enthalpy to the COSMO*therm* table and output files The three contributions to H^E are the misfit excess enthalpy H^E(MF), a contribution from hydrogen bonding H^E(HB) and a contribution from van der Waals interactions H^E(vdW).

If several conformers are present for a given compound, the computed values of the phase diagram properties of the conformers will be averaged due to the Boltzmann distribution of the conformers total free energies at the given temperature and mixture concentration (see section 2.2.2). By default only the average property of the compound is printed to the output and table file.The global keyword `wconf` (see section 2.1) toggles the printing of all conformer thermodynamic properties as well as conformer weights for all temperatures and mixtures to the output file. For the often large number of temperature and mixture state points that are computed in the course of an automatic phase diagram calculation, it might be difficult to extract the conformer information from the output file. Thus an additional suboption has been introduced into the automatic phase diagram calculation options, which allows the printing of the conformer weight factors to the phase diagram table in the COSMO*therm* table file. The keyword **`confweight`** (given in the same line as the `binary`, `ternary` or `multinary` command) toggles the printing of the conformer weights to the binary, ternary or multinary phase diagram table output. For each of the conformers of the given compounds (or all compounds in the case of a `multinary` computation) there will be added a column with the conformers Boltzmann weight factor added to the phase diagram table. If no conformers are present for a compound only one additional column will be written and the conformer weight factors in this column will all be equal to one.

General printing suboptions of the `binary`, `ternary` or `multinary` phase diagram computations:

`pr_pp`	Optional for `binary`, `ternary` or `multinary` computations: in the `binary`, `ternary` or `multinary` phase diagram table replace the default output of the compound mole fraction in the gas phase y_i with the compounds partial vapor pressure p_i.
`HE_SPLIT`	Optional for `binary`, `ternary` or `multinary` computations: print the three contributions to the total excess enthalpy to the COSMO*therm* table and output files.
`confweight`	Optional for `binary`, `ternary` or `multinary` computations: print, as additional columns, the Boltzmann weight factors of molecular conformers to the `binary`, `ternary` or `multinary` phase diagram table.

If no additional phase diagram calculation suboptions are given in the mixture input, i.e. if a simple vapor-liquid equilibrium (VLE) calculation is performed, the `binary` and `ternary` options will compute a default grid of 29 and 231 mole fraction concentration points, respectively. The default grid values span the complete concentration range of the two and three dimensional binary and ternary phase space. The concentrations of the VLE default grid are defined in mole fractions, which are unevenly spaced: the concentrations steps in the grid are becoming smaller if a compound approaches infinite dilution. This specific grid accounts for the fact that properties such as activity coefficients γ_i, or concentrations in the vapor phase y_i typically show their strongest changes at low x_i concentrations. Thus being uneven the default VLE grid tries to cover the phase space of the computed properties as comprehensive and effective as possible. If a liquid-liquid-equilibrium (LLE) calculation is performed using the `LLE` or `LLE_NEW` options (see below, section 2.3.7.3 for a closer description of these options), the default grid values are modified. The default VLE grid of 29 concentrations is used for `binary` calculations with the `LLE` option. The iterative `LLE` search algorithm (see below, section 2.3.7.3 for details on the LLE algorithms) starts from an initial guess derived from the coarse VLE default grid. If the `binary` calculation is performed with the `LLE_NEW` option, a considerably finer default grid of 323 evenly spaced mole fraction concentrations is used. The iterative `LLE_NEW` algorithm starts from an initial guess derived from this fine grid. A `ternary` calculation with `LLE` search option behaves slightly different. If no other concentration options are given, the `ternary LLE` option will compute a default grid of 33 mole fraction concentration, which strides across the ternary phase space in three evenly spaced grids, which all follow the same slicing pattern: the phase space is crossed from one of the pure compounds towards an even 1:1 mixture of the remaining two components of the ternary phase space. Unlike the `binary` LLE options, which only perform one LLE search per `binary` calculation, the `ternary LLE` option will perform LLE searches for each grid point. I.e. each of the 33 grid concentrations is used as individual starting point for a LLE search. If a solid-liquid-equilibrium (SLE) calculation is performed via the `SLE` option (see below, section 2.3.7.4 for a closer description of SLE calculations), the default grid is modified. If a `binary` calculation is performed with the `SLE` option, a considerably finer default grid of 323 evenly spaced mole fraction concentrations (same as for the `LLE_NEW` option, above) is used. The iterative `SLE` algorithm will initiate from a first guess that is derived from this fine grid.

The default binary and ternary VLE, LLE, and SLE concentration grids are given in mole fraction concentrations x_i. This may not always be the best choice for an optimal coverage of the phase space with as little grid points as possible. If the compounds used show large differences in their size or shape, it may be better if the grid concentrations are scaled in a way that reflects these size or shape differences. Such can be achieved by defining the grid point concentration in terms of mass fraction concentrations c_i or surface area fraction concentrations q_i, both of which somehow reflect apparent size and shape differences of the compounds involved. COSMO*therm* offers two possibilities to modify the grid concentrations in the framework of the `binary` and `ternary` calculation options. First, it is possible to modify the default grids with the options **`xgrid`**, **`cgrid`**, and **`qgrid`**. These option will change the basis of the binary or ternary VLE, LLE, or SLE concentration grid into mole fraction concentrations (`xgrid` keyword, default), into mass fraction concentrations (`cgrid` keyword), or into surface fraction concentrations (`qgrid` keyword), respectively. Second, it is possible to define a custom concentration grid via the definition of a start-concentration vector, an end-concentration vector, and the number of grid points to be computed between the two concentration vectors. The custom grid definition is possible in mole fractions, using options **`xstart`**, **`xend`**, and **`xstep`**, in mass fractions, using options **`cstart`**, **`cend`**, and **`cstep`**, and in surface fractions, using options **`qstart`**, **`qend`**, and **`qstep`**, respectively. The concept of custom grids is described to further detail in section 2.3.7.7, below. Please note that the conversion of the grid concentrations via `[x/c/q]grid` or the custom grid options `[x/c/q]start`, `[x/c/q]end`, and `[x/c/q]step` will not change the units of the concentrations in the output and table files of COSMO*therm*. On output all concentrations by default will be converted into mole fractions x_i.

Optional concentration grid input for the `binary` and `ternary` options:

`xgrid`
or
`cgrid`
or
`qgrid`
	Optional for `binary` and `ternary` computations: Change default concentration grid of the VLE, LLE, or SLE calculation to mole fraction (`xgrid`, default), mass fraction (`cgrid`), or surface fraction (`qgrid`) concentrations.

xstart={x_1 x_2 [x_3]} or cstart={c_1 c_2 [c_3]} or qstart={q_1 q_2 [q_3]}	Optional for `binary` and `ternary` computations: Mole fraction concentration grid input of initial mole fraction (`xstart`), mass fraction (`cstart`), or surface fraction (`qstart`) concentration vector of the compounds of the `binary` or `ternary` mixture (as real numbers x_i, c_i,and q_i). The arguments are expected as real numbers between zero and one. If the values do not add up to one, COSMO*therm* will normalize them. If less concentration values than defined phases are given, the missing ones will be assumed zero. If a negative number is given, the concentration for this component will be inserted automatically using the normalization of the sum of the remaining concentrations. Only one negative number is allowed per concentration input.
xend={x_1 x_2 [x_3]} or cend={c_1 c_2 [c_3]} or qend={q_1 q_2 [q_3]}	Optional for `binary` and `ternary` computations: Mole fraction concentration grid input of final mole fraction (`xstart`), mass fraction (`cstart`), or surface fraction (`qstart`) concentration vector of the compounds of the `binary` or `ternary` mixture (as real numbers x_i, c_i,and q_i).. Properties of the `xend`, `cend`, and `qend` commands are the same as described for the `xstart`, `cstart`, and `qstart` keywords.
xstep=n_{points} or cstep=n_{points} or qstep=n_{points}	Optional for `binary` and `ternary` computations: Input of the number of grid points to be calculated in the concentration grid computation. Argument n_{points} is expected to be an integer number between 2 and 235. Default is n_{points}=10. If `xstep` is used, the concentration grid points will be chosen evenly spaced between the given start- and end-concentration vectors in the frame of mole fraction concentrations. If `cstep` is used, the points will be chosen evenly spaced in the frame of mass fraction concentrations. If `qstep` is used, the points will be chosen evenly spaced in the frame of surface fraction concentrations. The `xstep`, `cstep`, or `qstep` options are independent of the surface, mass or mole fraction frame of the start- and end-concentration vectors – i.e. any combination of `xstep`, `cstep`, or `qstep` with `xstart`, `cstart`, or `qstart` and `xend`, `cend`, or `qend` is possible.

In addition to the general print options, it is possible to toggle an automatic search for Liquid-Liquid equilibria (LLE's) for binary, ternary, or multidimensional mixtures:

`LLE`	Optional for `binary`, `ternary` or `multinary` computations: Search for points of phase separation (liquid-liquid-equilibria). This option is searching the LLE with an iterative optimization procedure. See section 2.3.7.3.
`threshold=value`	Optional for `binary`, `ternary` or `multinary` computations with iterative **`LLE`** search: give the accuracy threshold for the iterative refinement of the LLE calculation. Argument value is expected to be a nonzero positive real number.Default is `threshold=1.0E-5`.
`maxiter=value`	Optional for `binary`, `ternary` or `multinary` computations with iterative **`LLE`** search: give the maximum number of iterations for the iterative LLE point calculation. Argument value is expected to be a nonzero positive integer number.Default is `maxiter=1000`.
`LLE_NEW`	Optional for `binary` computations: Search the computed binary mixture for points of phase separation (liquid-liquid-equilibria). This option is searching the LLE on a fine grid using additional mixture concentrations points. In addition to the binodal LLE found by the regular LLE option, the LLE_NEW option also computes the spinodal miscibility gap (see section 2.3.7.3).
`conf_dx=value`	Optional for `binary`, computations with `LLE_NEW` search: give the accuracy threshold for the iterative refinement of the `LLE_NEW` calculation. Argument value is expected to be a nonzero positive real number. Default is `conf_dx=1.0E-5`.
`maxiter_dx=value`	Optional for `binary` computations with `LLE_NEW` search: give the maximum number of iterations for the iterative `LLE_NEW` point calculation. Argument value is expected to be a nonzero positive integer number. Default is `maxiter_dx=400`.

Other general control options of the `binary`, `ternary`, and `multinary` options:

`use_tboil`	Optional for the input of boiling point temperatures: use the pure compound boiling points T_{Boil} or a pair of reference pressure and temperature $p_{ref,}$ T_{ref} as given in the compound input section or read from a compounds vapor-pressure / property file (see section 2.2.3) as a reference point for scaling the vapor pressure prediction of the `binary`, `ternary`, or `multinary` option.

The **`binary`** option allows further automatic post-processing of the computed properties such as fitting of the computed activity coefficients to activity coefficient models, special LLE search, and solid-liquid equilibrium (SLE) detection.

Suboptions specific to the `binary` option are:

`NRTL`	Optional for `binary` computations: Correlate the computed activity coefficients with the three parameter `NRTL` model. See section 2.3.7.2.
`NRTL2`	Optional for `binary` computations: Correlate the computed activity coefficients with the two parameter `NRTL2` model. Only the two binary interaction parameters τ_{12} and τ_{21} are fitted to the computed activity coefficient data. Factor α is held at a constant value. See section 2.3.7.2.
`WILSON`	Optional for `binary` computations: Correlate the computed activity coefficients with the `WILSON` model. See section 2.3.7.2.
`UNIQUAC2`	Optional for `binary` computations: Correlate the computed activity coefficients with the UNIQUAC model. The two binary interaction parameters τ_{12} and τ_{21} are fitted to the computed activity coefficient data. The compounds volumes and surfaces r_i and q_i are not fitted. See section 2.3.7.2.
`UNIQUAC4`	Optional for `binary` computations: Correlate the computed activity coefficients with the UNIQUAC model. The two binary interaction parameters τ_{12} and τ_{21} and the compound volume and surface parameters r_i and q_i are fitted to the computed activity coefficient data. See section 2.3.7.2.
`UNIQUAC6`	Optional for `binary` computations: Correlate the computed activity coefficients with the UNIQUAC model. The two binary interaction parameters τ_{12} and τ_{21} and the compound volume and surface parameters r_i and q_i are fitted to the computed activity coefficient data. See section 2.3.7.2.
`UNIQUAC8`	Optional for `binary` computations: Correlate the computed activity coefficients with the UNIQUAC model. The two binary interaction parameters τ_{12} and τ_{21} and the compound volume and surface parameters r_i, $q_i^{residual}$, and $q_i^{combinatorial}$ are fitted to the computed activity coefficient data. See section 2.3.7.2.
`SLE`	Optional for `binary` computations: Search the computed binary mixture for points of solid phase separation (solid-liquid-equilibria). This option is searching the LLE on a fine grid using additional mixture concentrations points. See section 2.3.7.4.

In binary phase diagrams, COSMO*therm* also looks for possible azeotropes.. If an azeotrope is found, the mole fraction concentration of the azeotropic point will be written to the COSMO*therm* output and table files. By default the azeotrope concentration as found on the given concentration grid is used.With the keyword `search-azeotrope` COSMO*therm* will perform an additional iterative refinement of the azeotropic point(s) found. By default the iterative refinement will compute the azeotropc points up to an accuracy threshold of $x_i = 10^{-5}$ mole fractions within a maximum number of 500 iterations. These defaults can be changed by options `azeo-iter-thresh=value` for the accuracy threshold value, and `maxiter-azeo=value` for the maximum number of iterations.

Azeotrope-related suboptions specific to the `binary` option:

`search-azeotrope`	Optional for `binary` computations.: Toggle iterative refinement of the azeotropic point(s) in the computed binary mix. This option solves the thermodamic equilibrium of the goven binary VLE for Azeotropes using an iterative self-consistent algorithm, which computes additional mixture concentrations points.
`azeo-iter-thresh=value`	Optional for `binary` computations with iterative azeotrope search: give the accuracy threshold for the iterative refinement of the azeotropic point calculation. Argument value is expected to be a nonzero positive real number.Default is `azeo-iter-thresh=1.0E-5`.
`maxiter-azeo=value`	Optional for `binary` computations with iterative azeotrope search: give the maximum number of iterations for the iterative refinement of the azeotropic point calculation. Argument value is expected to be a nonzero positive integer number.Default is `maxiter-azeo=500`.

2.3.7.2 Activity Coefficient Model Computation for Binary Mixtures

If using the **`binary`** option COSMO*therm* offers the possibility to correlate the evaluated data with activity coefficient models, such as the non random two-liquid (NRTL) model[42], Wilson's equation[43] and the UNIQUAC model[16] simply by giving the keywords `NRTL`, `WILSON`, `UNIQUAC2` or `UNIQUAC4` in combination with the `binary` command.

If the **`NRTL`** command is used, selected activity coefficients of the binary computation are fitted to the NRTL equation[42]:

$$\ln(\gamma_i) = \frac{\sum_j x_j \tau_{ji} G_{ji}}{\sum_k x_k G_{ki}} + \sum_j \frac{x_i G_{ij}}{\sum_k x_k G_{kj}} \left[\tau_{ij} - \frac{\sum_m x_m \tau_{mj} G_{mj}}{\sum_k x_k G_{kj}} \right] \qquad (2.3.7\text{-}3)$$

$$G_{ij} = \exp\left(-\alpha_{ij}\tau_{ij}\right)$$

Per binary pair three adjustable parameters of the NRTL equation (τ_{ij}, τ_{ji}, and $\alpha = \alpha_{ij} = \alpha_{ji}$) are fitted to the COSMO*therm* activity coefficients in a least squares procedure. The optimized values of the parameters are written to the COSMO*therm* output and table files, below the output and table of the `binary` option, respectively. If the **`NRTL2`** option is used only the binary interaction parameters τ_{ij} and τ_{ji} are fitted. Factor α is held at a constant value. By default α=0.3 is used for the `NRTL2` option. However, it is also possible to give the value of α in the COSMO*therm* input via option `NRTL_ALPHA=value`. Reasonable values of α are α=0.3 for common organic mixtures, α=0.2 for mixtures that show a miscibility gap and α=0.46 for "associating" systems that show strong interactions between the two compounds.

Suboptions of the `binary` / `NRTL2` option are:

`NRTL_ALPHA=value`	Optional for `binary` / `NRTL2` computations: Give NRTL parameter α. Argument `value` is expected as a real number.

If the **`WILSON`** command is given, the COSMO*therm* activity coefficients are adjusted in a least squares procedure to Wilson's equation[43]:

$$\ln(\gamma_i) = 1 - \ln\left(\sum_k x_k \Lambda_{ik}\right) - \sum_j \frac{x_k \Lambda_{ji}}{\sum_k x_k \Lambda_{jk}} \qquad (2.3.7\text{-}4)$$

$$\Omega_{ij} = \Lambda_{ji} = \frac{V_i}{V_j} \exp\left[-a_{ji} / RT\right] \qquad (2.3.7\text{-}5)$$

---

[42] Renon, H.; Prausnitz J. M. *AIChE Journal*, **14**, 135 (1968).

[43] Wilson, G. M. *J. Am. Chem. Soc.*, **86**, 127 (1964).

The adjustable binary interaction parameters of Wilson's equation $\Omega_{ij} = \Lambda_{ji}$ are written to the COSMO*therm* output file below the output of the `binary` option.

If the **`UNIQUAC2`** or the **`UNIQUAC4`** command is given, the COSMO*therm* activity coefficients are adjusted in a least squares procedure to the UNIQUAC equation[16,44]. The activity coefficient of a species *i* in a mixture is built from two contributions:

$$\ln \gamma_i = \ln \gamma_i^C + \ln \gamma_i^R \tag{2.3.7-6}$$

The first term, the *combinatorial* contribution $\ln \gamma_i^C$, accounts for the entropic size and shape differences of the compounds. The coordination number *z* is set equal to 10.

$$\ln \gamma_i^C = \ln \frac{\Phi_i}{x_i} + \frac{z}{2} q_i \ln \frac{\Theta_i}{\Phi_i} + l_i - \frac{\Phi_i}{x_i} \sum_j x_j l_j \tag{2.3.7-7}$$

Φ_i and Θ_i are the normalized volume and surface area fraction of species *i* in the mixture. Each species *i* is characterized by its mole fraction concentration in the mixture x_i, its volume r_i and its surface area q_i.

$$\Phi_i = \frac{x_i r_i}{\sum_j r_j x_j}$$

$$\Theta_i = \frac{x_i q_i}{\sum_j q_j x_j}$$

$$l_i = \frac{z}{2}(r_i - q_i) - (r_i - 1)$$

The second contribution to the activity coefficient (the *residual* contribution $\ln \gamma_i^R$) describes the enthalpic interactions between the different compounds in the mixture.

$$\ln \gamma_i^R = q_i \left[1 - \ln\left(\sum_j \Theta_j \tau_{ji} \right) - \sum_j \frac{\Theta_j \tau_{ij}}{\sum_k \Theta_k \tau_{kj}} \right] \tag{2.3.7-8}$$

For any species *i* – species *j* interaction it is also common to express the UNIQUAC interaction parameters τ_{ij} by a temperature-dependent average interaction energy parameter $\Delta u_{ij} = a_{ij}/R$:

$$\ln \tau_{ij} = -\frac{\Delta u_{ij}}{RT} = -\frac{a_{ij}}{T} \tag{2.3.7-9}$$

Thus, for binary mixtures the UNIQUAC model contains only two adjustable parameters τ_{12} and τ_{21}.

---

[44] Anderson, T. F.; Prausnitz, J. M. *Ind. Eng. Chem. Process Des. Dev.*, **17**, 552 (1978).

If the **UNIQUAC2** option is used within the framework of a binary calculation, only the two UNIQUAC interaction parameters τ_{12} and τ_{21} are adjusted to match the activity coefficient data computed by COSMO*therm*. By default, the compound-specific UNIQUAC volume and surface area parameters are approximated by scaled volumes and areas of the compounds COSMO surface:

$$\begin{aligned} r_i &= V_i^{COSMO}/30 \\ q_i &= A_i^{COSMO}/40 \end{aligned} \tag{2.3.7-10}$$

Alternatively, the UNIQUAC volume and area parameters r_i and q_i can be given in the input file using the options `UNIQUAC_R1=r1`, `UNIQUAC_R2=r2`, `UNIQUAC_Q1=q1` and `UNIQUAC_Q2=q2` or, in the compound input section, the options `UNIQUAC_RI=ri` and `UNIQUAC_QI=qi`.

Suboptions of the `binary` / `UNIQUAC2` option are:

`UNIQUAC_R1=r1`	Optional for `binary` / `UNIQUAC2` computations: Give UNIQUAC volume parameter for compound 1. Argument `r1` is expected as a real number larger than zero.
`UNIQUAC_R2=r2`	Optional for `binary` / `UNIQUAC2` computations: Give UNIQUAC volume parameter for compound 2. Argument `r2` is expected as a real number larger than zero.
`UNIQUAC_Q1=q1`	Optional for `binary` / `UNIQUAC2` computations: Give UNIQUAC surface area parameter for compound 1. Argument `q1` is expected as a real number larger than zero.
`UNIQUAC_Q2=q2`	Optional for `binary` / `UNIQUAC2` computations: Give UNIQUAC surface area parameter for compound 2. Argument `q2` is expected as a real number larger than zero.

Please note, that the input of UNIQUAC volume and surface parameters in the mixture section of the COSMO*therm* input file (i.e. as a suboption of the the `binary` / `UNIQUAC2` command) overrides any values given in the compound input section of the COSMO*therm* input file via `UNIQUAC_RI` and `UNIQUAC_QI`.

If the **UNIQUAC4** option is used within the framework of a binary calculation, the two UNIQUAC interaction parameters τ_{12} and τ_{21} are adjusted to match the activity coefficient data computed by COSMO*therm*. In addition, the compound-specific UNIQUAC volume and surface area parameters are adjusted to optimally match the activity coefficient data computed by COSMO*therm*. This is achieved by introducing an additional compound-specific scaling factor s_i to the computation of the UNIQUAC volume and area parameters.

$$\begin{aligned} r_i &= s_i V_i^{COSMO}/30 \\ q_i &= s_i A_i^{COSMO}/40 \end{aligned} \tag{2.3.7-11}$$

Thus, for binary mixture computations the `UNIQUAC4` option depends upon four adjustable parameters τ_{12}, τ_{21}, s_1 and s_2.

If the **UNIQUAC6** option is used within the framework of a binary calculation, the two UNIQUAC interaction parameters τ_{12} and τ_{21} are adjusted to match the activity coefficient data computed by COSMO*therm*. In addition, the compound-specific UNIQUAC volume and surface area parameters are adjusted to optimally match the activity coefficient data computed by COSMO*therm*. This is achieved by introducing two additional compound-specific scaling factor s_i^q and s_i^r to the computation of the UNIQUAC volume and area parameters.

$$\begin{aligned} r_i &= s_i^r V_i^{COSMO} / 30 \\ q_i &= s_i^q A_i^{COSMO} / 40 \end{aligned} \qquad \text{(2.3.7-11a)}$$

Thus, for binary mixture computations the `UNIQUAC6` option depends upon four adjustable parameters τ_{12}, τ_{21}, s_1^q, s_1^r, s_2^q, and s_2^r.

If the **`UNIQUAC8`** option is used within the framework of a binary calculation, the two UNIQUAC interaction parameters τ_{12} and τ_{21} are adjusted to match the activity coefficient data computed by COSMO*therm*. In addition, the compound-specific UNIQUAC volume and surface area parameters are adjusted to optimally match the activity coefficient data computed by COSMO*therm*. This is achieved by introducing three additional compound-specific scaling factor $s_i^{q(residual)}$, $s_i^{q(combinatorial)}$, and s_i^r to the computation of the UNIQUAC volume and area parameters.

$$\begin{aligned} r_i &= s_i^r V_i^{COSMO} / 30 \\ q_i^r &= s_i^{q(residual)} A_i^{COSMO} / 40 \\ q_i^c &= s_i^{q(combinatorial)} A_i^{COSMO} / 40 \end{aligned} \qquad \text{(2.3.7-11b)}$$

In this case equations 2.3.7-6 to 2.3.7-11 correspond to the extended UNIQUAC equation of Prausnitz *et al.*[44]. The extended UNIQUAC method uses separate area parameters q_i^r and q_i^c for the residual and combinatorial part of the UNIQUAC equation (eqs. 2.3.7-7 and 2.3.7-8, respectively). Thus, for binary mixture computations the `UNIQUAC8` option depends upon four adjustable parameters τ_{12}, τ_{21}, $s_1^{q(residual)}$, $s_1^{q(combinatorial)}$, s_1^r, $s_2^{q(residual)}$, $s_2^{q(combinatorial)}$, and s_2^r.

2.3.7.3 Liquid-Liquid Equilibrium Computation for Binary Mixtures

For the `binary`, `ternary`, and `multinary` computation options, COSMO*therm* offers the possibility to detect miscibility gaps, i.e. points of liquid-liquid-equilibrium (LLE) phase separation via the keyword **`LLE`**. If this command is given in the same line as the `binary`, `ternary`, or `multinary` command, COSMO*therm* will search the computed mixtures for possible points of separation and if found, writes them to the COSMO*therm* output and table file. The LLE properties are calculated from the liquid phase equilibrium condition eq. 2.3.7-12, where indices ′′ and ′ denote the two liquid phases, and *i* denotes the compound:

$$x_i'\gamma_i' = x_i''\gamma_i'' \qquad \text{for all compounds } i \qquad (2.3.7\text{-}12)$$

If the **`LLE`** option is used with the **`binary`** option, the LLE tie points will be written to the COSMO*therm* table file. In addition, the `binary` property table will be modified according to any LLE that has been detected. Within the points of LLE, the vapor pressures (or for `isobar` calculations the temperatures) and the mole fractions in the gas phase y_i will be replaced by the values of the LLE points. In practice, any miscibility gap as found by COSMO*therm* will be visible as a straight horizontal line in the *x-y* and $xy\text{-}p_{tot}$ phase diagram. The iterative LLE search for `binary` systems by defaults starts at the LLE points eventually found on the binary concentration grid. However it is also possible to give an explicit starting concentration via the `xstart` keyword (see section 2.3.7.7).

If the command **`LLE_NEW`** is given in the same line as the **`binary`** command, COSMO*therm* will compute the thermodynamic properties of the binary mixture at 325 additional mixture concentrations that are distributed on an even spaced grid. In addition to the *binodal* LLE defined by equation 2.3.7-12 COSMO*therm* also computes the *spinodal* LLE points, which distinguish the unstable region of a liquid mixture [$(\partial^2\Delta G_{mix}/\partial x^2)_{T,p} < 0$] from the metastable region [$(\partial^2\Delta G_{mix}/\partial x^2)_{T,p} > 0$]. If the mole fraction of the binary mixture falls within the unstable region, spontaneous phase separation occurs when going from the one-phase to the two-phase region[45]. For an LLE thus found, the points of spinodal phase separation are also written to the COSMO*therm* output file. The `LLE_NEW` option is only possible for isothermal binary computations. If given for an isobaric system, or for a system with more than two phases, COSMO*therm* will use the regular `LLE` search procedure instead. The thermodynamic properties of the additional mixtures are written to the COSMO*therm* output file and, in tabulated form to the COSMO*therm* table file. Subsequently, COSMO*therm* will search the computed mixtures for possible points of liquid phase separation and if found, writes them to the COSMO*therm* output file. If the commands `LLE` or `LLE_NEW` are given in the same line as the `binary` command, COSMO*therm* will compute the thermodynamic properties of the binary mixture on a grid (the coarse 29 point grid is used for option `LLE` and the fine 325 point grid is used for the `LLE_NEW` option) and afterwards refine the value of the binodal LLE miscibility gap found on the grid with an iterative optimization procedure. The iteratively optimized binodal LLE value will be written to output and table files and replaces the value found on the grid.

If the **`LLE`** option is used with the **`ternary`** or **`multinary`** option, additional tabulated output with the LLE tie points and related information will be printed to the COSMO*therm* table file. In the case of the `ternary` option, a different default grid will be used if the LLE search is toggled. Instead of the irregular grid of 231 concentrations in the ternary phase space that is used for VLE calculations, the LLE search uses a grid of 3*11 concentrations scanning the three-dimensional phase space along the concentration vectors

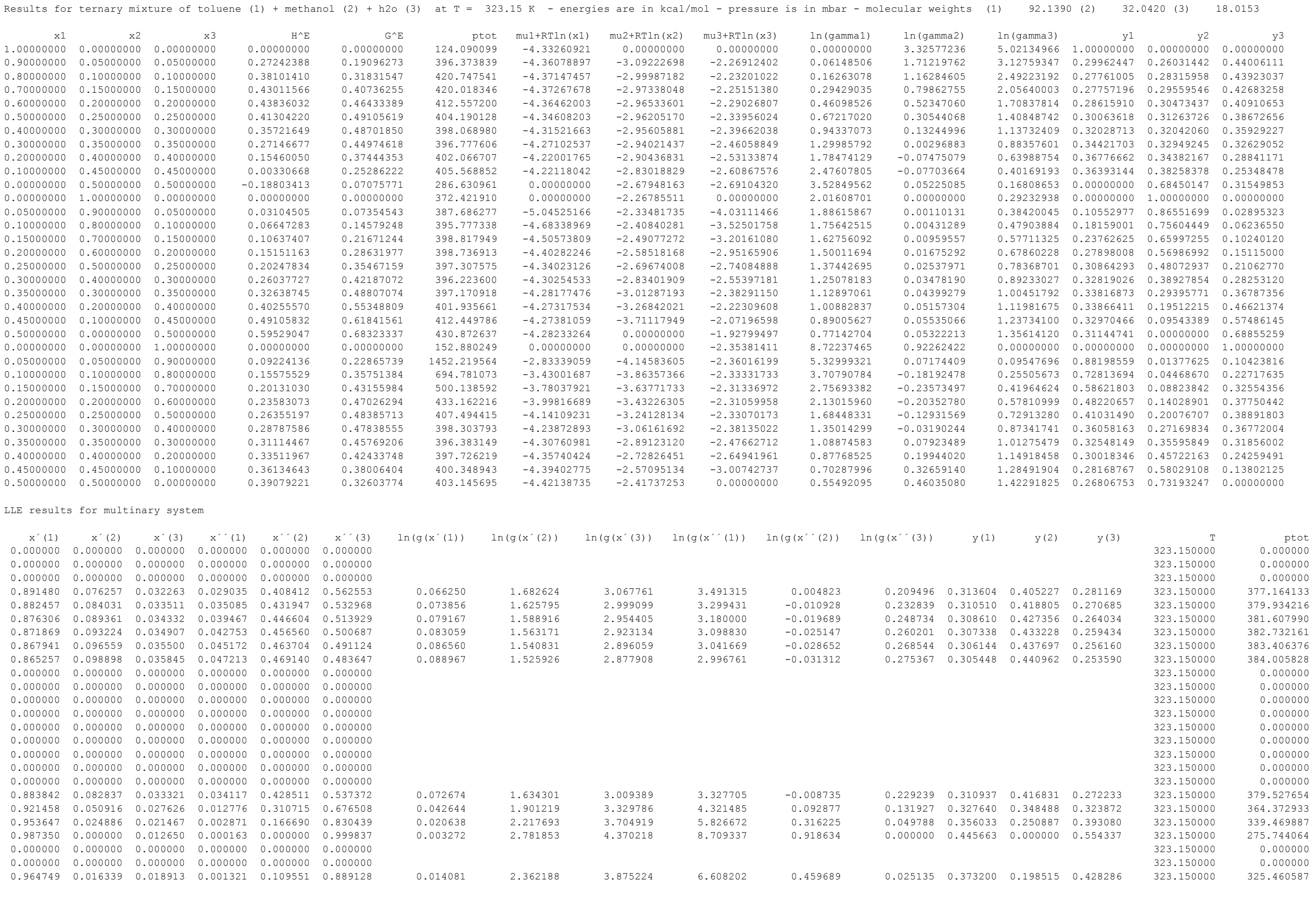
---

[45] Prausnitz, J. M.; Lichtenthaler, R. M.; Azevedo, E.G. *Molecular Thermodynamics of Fluid Phase Equilibria*, 3rd ed., Prentice Hall PTR, Upper Saddle River, NJ (1999).

that lead from each pure compound to an equimolar mix of the two other compounds, respectively. Each of the concentration points of the `ternary` phase vector given in the input will be used as starting point for an LLE search. Thus by default COSMO*therm* will perform 33 LLE searches and compute 33 possible LLE tie points. As an alternative it is possible to compute the LLE tie points on a given grid of concentration points along a concentratin vector. The input format of this is the same as for the `multinary` option and is described in section 2.3.7.7. The `ternary` computation in combination with the `LLE` option will produce two table blocks in the COSMO*therm* table file: first the regular `multinary` VLE table for 33 default or the given concentrations. Second a table with the LLE tie points and related properties. It is possible to use one or more Ionic Liquids phases in a `ternary` computation (see section 2.3.7.5). If one or more Ionic Liquids phases are defined in the input then, in addition to the VLE and LLE table blocks a third table block will printed to the table file. This third table holds the information of the LLE tie points of the second table converted to the "laboratory-ternary" framework of concentrations (see sections 2.3.7.5 and 4.9).

In the case of the `multinary` option, first the regular n-dimensional phase diagram information according to the `multinary` phase vector input is printed to a table block in the table file. Each of the concentration points of the `multinary` phase vector given in the input will be used as starting point for an LLE search. Thus for *k* given concentrations (steps along the `multinary` phase vector), COSMO*therm* will perform *k* LLE searches and compute *k* possible LLE tie points. The LLE tie points then will be printed to a second table block below the `multinary` VLE table. For details on the computation of n-dimensional phase diagrams with the `multinary` option, see section 2.3.7.7.

A typical `ternary` or `multinary` LLE computation output is shown below. First the regular VLE output table with the mole fraction concentrations of the grid points (and all computed themrdynamic properties such as H^E, G^E, p^{tot} at the grid points) is printed, followed by a table with the results of the LLE searches, each started at the concentration grid points given in the upper table. The LLE results table holds the two tie point concentrations *x´(i)* and *x´´(i)* for the three phases *i* = 1, 2, 3. If no LLE was found the values of *x´(i)* and *x´´(i)* are set to zero. The following columns of the LLE results table hold the logarithm of the activity coefficients at the tie points (blank if no LLE was found) and the temperature and total pressure at the tie points:



```
Results for ternary mixture of toluene (1) + methanol (2) + h2o (3)  at T =  323.15 K  - energies are in kcal/mol - pressure is in mbar - molecular weights  (1)    92.1390 (2)    32.0420 (3)    18.0153

        x1         x2         x3           H^E           G^E          ptot  mu1+RTln(x1)  mu2+RTln(x2)  mu3+RTln(x3)    ln(gamma1)    ln(gamma2)    ln(gamma3)         y1         y2         y3
1.00000000 0.00000000 0.00000000    0.00000000    0.00000000    124.090099   -4.33260921    0.00000000    0.00000000    0.00000000    3.32577236    5.02134966 1.00000000 0.00000000 0.00000000
0.90000000 0.05000000 0.05000000    0.27242388    0.19096273    396.373839   -4.36078897   -3.09222698   -2.26912402    0.06148506    1.71219762    3.12759347 0.29962447 0.26031442 0.44006111
0.80000000 0.10000000 0.10000000    0.38101410    0.31831547    420.747541   -4.37147457   -2.99987182   -2.23201022    0.16263078    1.16284605    2.49223192 0.27761005 0.28315958 0.43923037
0.70000000 0.15000000 0.15000000    0.43011566    0.40736255    420.018346   -4.37267678   -2.97338048   -2.25151380    0.29429035    0.79862755    2.05640003 0.27757196 0.29559546 0.42683258
0.60000000 0.20000000 0.20000000    0.43836032    0.46433389    412.557200   -4.36462003   -2.96533601   -2.29026807    0.46098526    0.52347060    1.70837814 0.28615910 0.30473437 0.40910653
0.50000000 0.25000000 0.25000000    0.41304220    0.49105619    404.190128   -4.34608203   -2.96205170   -2.33956024    0.67217020    0.30544068    1.40848742 0.30063618 0.31263726 0.38672656
0.40000000 0.30000000 0.30000000    0.35721649    0.48701850    398.068980   -4.31521663   -2.95605881   -2.39662038    0.94337073    0.13244996    1.13732409 0.32028713 0.32042060 0.35929227
0.30000000 0.35000000 0.35000000    0.27146677    0.44974618    396.777606   -4.27102537   -2.94021437   -2.46058849    1.29985792    0.00296883    0.88357601 0.34421703 0.32949245 0.32629052
0.20000000 0.40000000 0.40000000    0.15460050    0.37444353    402.066707   -4.22001765   -2.90436831   -2.53133874    1.78474129   -0.07475079    0.63988754 0.36776662 0.34382167 0.28841171
0.10000000 0.45000000 0.45000000    0.00330668    0.25286222    405.568852   -4.22118042   -2.83018829   -2.60867576    2.47607805   -0.07703664    0.40169193 0.36393144 0.38258378 0.25348478
0.00000000 0.50000000 0.50000000   -0.18803413    0.07075771    286.630961    0.00000000   -2.67948163   -2.69104320    3.52849562    0.05225085    0.16808653 0.00000000 0.68450147 0.31549853
0.00000000 1.00000000 0.00000000    0.00000000    0.00000000    372.421910    0.00000000   -2.26785511    0.00000000    2.01608701    0.00000000    0.29232938 0.00000000 1.00000000 0.00000000
0.05000000 0.90000000 0.05000000    0.03104505    0.07354543    387.686277   -5.04525166   -2.33481735   -4.03111466    1.88615867    0.00110131    0.38420045 0.10552977 0.86551699 0.02895323
0.10000000 0.80000000 0.10000000    0.06647283    0.14579248    395.777338   -4.68338969   -2.40840281   -3.52501758    1.75642515    0.00431289    0.47903884 0.18159001 0.75604449 0.06236550
0.15000000 0.70000000 0.15000000    0.10637407    0.21671244    398.817949   -4.50573809   -2.49077272   -3.20161080    1.62756092    0.00959557    0.57711325 0.23762625 0.65997255 0.10240120
0.20000000 0.60000000 0.20000000    0.15151163    0.28631977    398.736913   -4.40282246   -2.58518168   -2.95165906    1.50011694    0.01675292    0.67860228 0.27898008 0.56986992 0.15115000
0.25000000 0.50000000 0.25000000    0.20247834    0.35467159    397.307575   -4.34023126   -2.69674008   -2.74084888    1.37442695    0.02537971    0.78368701 0.30864293 0.48072937 0.21062770
0.30000000 0.40000000 0.30000000    0.26037727    0.42187072    396.223600   -4.30254533   -2.83401909   -2.55397181    1.25078183    0.03478190    0.89233027 0.32819026 0.38927854 0.28253120
0.35000000 0.30000000 0.35000000    0.32638745    0.48807074    397.170918   -4.28177476   -3.01287193   -2.38291150    1.12897061    0.04399279    1.00451792 0.33816873 0.29395771 0.36787356
0.40000000 0.20000000 0.40000000    0.40255570    0.55348809    401.935661   -4.27317534   -3.26842021   -2.22309608    1.00882837    0.05157304    1.11981675 0.33866411 0.19512215 0.46621374
0.45000000 0.10000000 0.45000000    0.49105832    0.61841561    412.449786   -4.27381059   -3.71117949   -2.07196598    0.89005627    0.05535066    1.23734100 0.32970466 0.09543389 0.57486145
0.50000000 0.00000000 0.50000000    0.59529047    0.68323337    430.872637   -4.28233264    0.00000000   -1.92799497    0.77142704    0.05322213    1.35614120 0.31144741 0.00000000 0.68855259
0.00000000 0.00000000 1.00000000    0.00000000    0.00000000    152.880249    0.00000000    0.00000000   -2.35381411    8.72237465    0.92262422    0.00000000 0.00000000 0.00000000 1.00000000
0.05000000 0.05000000 0.90000000    0.09224136    0.22865739   1452.219564   -2.83339059   -4.14583605   -2.36016199    5.32999321    0.07174409    0.09547696 0.88198559 0.01377625 0.10423816
0.10000000 0.10000000 0.80000000    0.15575529    0.35751384    694.781073   -3.43001687   -3.86357366   -2.33331733    3.70790784   -0.18192478    0.25505673 0.72813694 0.04468670 0.22717635
0.15000000 0.15000000 0.70000000    0.20131030    0.43155984    500.138592   -3.78037921   -3.63771733   -2.31336972    2.75693382   -0.23573497    0.41964624 0.58621803 0.08823842 0.32554356
0.20000000 0.20000000 0.60000000    0.23583073    0.47026294    433.162216   -3.99816689   -3.43226305   -2.31059958    2.13015960   -0.20352780    0.57810999 0.48220657 0.14028901 0.37750442
0.25000000 0.25000000 0.50000000    0.26355197    0.48385713    407.494415   -4.14109231   -3.24128134   -2.33070173    1.68448331   -0.12931569    0.72913280 0.41031490 0.20076707 0.38891803
0.30000000 0.30000000 0.40000000    0.28787586    0.47838555    398.303793   -4.23872893   -3.06161692   -2.38135022    1.35014299   -0.03190244    0.87341741 0.36058163 0.27169834 0.36772004
0.35000000 0.35000000 0.30000000    0.31114467    0.45769206    396.383149   -4.30760981   -2.89123120   -2.47662712    1.08874583    0.07923489    1.01275479 0.32548149 0.35595849 0.31856002
0.40000000 0.40000000 0.20000000    0.33511967    0.42433748    397.726219   -4.35740424   -2.72826451   -2.64941961    0.87768525    0.19944020    1.14918458 0.30018346 0.45722163 0.24259491
0.45000000 0.45000000 0.10000000    0.36134643    0.38006404    400.348943   -4.39402775   -2.57095134   -3.00742737    0.70287996    0.32659140    1.28491904 0.28168767 0.58029108 0.13802125
0.50000000 0.50000000 0.00000000    0.39079221    0.32603774    403.145695   -4.42138735   -2.41737253    0.00000000    0.55492095    0.46035080    1.42291825 0.26806753 0.73193247 0.00000000

LLE results for multinary system

    x´(1)    x´(2)    x´(3)   x´´(1)   x´´(2)   x´´(3)  ln(g(x´(1)))  ln(g(x´(2)))  ln(g(x´(3))) ln(g(x´´(1))) ln(g(x´´(2))) ln(g(x´´(3)))     y(1)     y(2)     y(3)              T          ptot
 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000                                                                                                                     323.150000      0.000000
 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000                                                                                                                     323.150000      0.000000
 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000                                                                                                                     323.150000      0.000000
 0.891480 0.076257 0.032263 0.029035 0.408412 0.562553      0.066250      1.682624      3.067761      3.491315      0.004823      0.209496 0.313604 0.405227 0.281169     323.150000    377.164133
 0.882457 0.084031 0.033511 0.035085 0.431947 0.532968      0.073856      1.625795      2.999099      3.299431     -0.010928      0.232839 0.310510 0.418805 0.270685     323.150000    379.934216
 0.876306 0.089361 0.034332 0.039467 0.446604 0.513929      0.079167      1.588916      2.954405      3.180000     -0.019689      0.248734 0.308610 0.427356 0.264034     323.150000    381.607990
 0.871869 0.093224 0.034907 0.042753 0.456560 0.500687      0.083059      1.563171      2.923134      3.098830     -0.025147      0.260201 0.307338 0.433228 0.259434     323.150000    382.732161
 0.867941 0.096559 0.035500 0.045172 0.463704 0.491124      0.086560      1.540831      2.896059      3.041669     -0.028652      0.268544 0.306144 0.437697 0.256160     323.150000    383.406376
 0.865257 0.098898 0.035845 0.047213 0.469140 0.483647      0.088967      1.525926      2.877908      2.996761     -0.031312      0.275367 0.305448 0.440962 0.253590     323.150000    384.005828
 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000                                                                                                                     323.150000      0.000000
 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000                                                                                                                     323.150000      0.000000
 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000                                                                                                                     323.150000      0.000000
 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000                                                                                                                     323.150000      0.000000
 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000                                                                                                                     323.150000      0.000000
 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000                                                                                                                     323.150000      0.000000
 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000                                                                                                                     323.150000      0.000000
 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000                                                                                                                     323.150000      0.000000
 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000                                                                                                                     323.150000      0.000000
 0.883842 0.082837 0.033321 0.034117 0.428511 0.537372      0.072674      1.634301      3.009389      3.327705     -0.008735      0.229239 0.310937 0.416831 0.272233     323.150000    379.527654
 0.921458 0.050916 0.027626 0.012776 0.310715 0.676508      0.042644      1.901219      3.329786      4.321485      0.092877      0.131927 0.327640 0.348488 0.323872     323.150000    364.372933
 0.953647 0.024886 0.021467 0.002871 0.166690 0.830439      0.020638      2.217693      3.704919      5.826672      0.316225      0.049788 0.356033 0.250887 0.393080     323.150000    339.469887
 0.987350 0.000000 0.012650 0.000163 0.000000 0.999837      0.003272      2.781853      4.370218      8.709337      0.918634      0.000000 0.445663 0.000000 0.554337     323.150000    275.744064
 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000                                                                                                                     323.150000      0.000000
 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000                                                                                                                     323.150000      0.000000
 0.964749 0.016339 0.018913 0.001321 0.109551 0.889128      0.014081      2.362188      3.875224      6.608202      0.459689      0.025135 0.373200 0.198515 0.428286     323.150000    325.460587
```

If the phase space considered in an LLE search has three or more dimension, i.e. if the `LLE` option is used within a `ternary` computation or with a `multinary` computation where $n > 2$, then it is possible that several different phase separations take place in the phase space spanned by the compound concentrations. Each of the pairs of LLE tie points found by solving equation 2.3.7-12 correspond to a minimum in the total Gibbs free energy of the system[46]. Dependent on the starting concentration it is possible that the solution of eq. 2.3.7-12 leads to a local minimum for systems with more than two components and more than two phases. Now while all solutions of eq. 2.3.7-12 correspond to actual LLE phase separations, these may not be the most stable phases that are possible in the system. The overall most stable phase separation corresponds to the global minimum of the system Gibbs free energy. It is possible to assess the LLE points found by eq. 2.3.7-12 using the method of tangent plane analysis, which claims that a phase specified at a given temperature, pressure and starting composition $\underline{x}^s$ is unstable if the Gibbs free energy of mixing falls below a hyperplane tangent to the phase space hypersurface at the start composition vector[47]. For a given start concentration $\underline{x}^s$ and the LLE tie points $\underline{x}^l$ found via eq. 2.3.7-12, the tangent plane criterion *tm* can be defined as

$$tm = \sum_{i=1}^{n} x_i^s \left[\ln\left(x_i^s \gamma_i^s\right) - \ln\left(x_i^l \gamma_i^l\right) \right] \tag{2.3.7-13}$$

This number is printed to COSMO*therm* output file for all `ternary` and `multinary` computation where $n > 2$. Actually *tm* is a measure of the steepness of the tangent between the LLE point (the Gibbs free energy minimum) and the starting point. It may be useful in two situations: First, if *tm* is negative, the stationary LLE point that was found is not the global minimum of the free energy. Second, if COSMO*therm* detects a discontinuity during the course of a LLE search, when stepping from one grid starting point to the next, i.e. if we have the case that the LLE tie points optimized belong to two different phase separations in the system – which is noted by a warning message in the COSMO*therm* output and table files. If this is the case, the *tm* values may help to assess the nature of the LLE points found: a large *tm* value implies a steep descent to the Gibbs free energy minimum and thus more likely the global minimum. Now if, in combination with the COSMO*therm* warning message one LLE point yields a large value of *tm* while the value for the next LLE point in the grid is considerably smaller, this suggests that the LLE point with the smaller *tm* might be a local Gibbs minimum and thus not a stable point of phase separation. In both situations noted above it is strongly suggested to rerun the LLE searches using different starting concentrations (e.g. use a finer grid of starting concentrations in the `ternary` or `multinary` computation, or try starting a new LLE search from inside the LLE points that have been found in the initial run). This way ultimately, the global Gibbs minima (the "stable" phase separations) may be found.

---

[46] Michelsen, M.L., and Mollerup, J.M., *Thermodynamic Models: Fundamentals and Computational Aspects*, 2nd ed., Tie-Line Publications, Denmark (2007).

[47] Michelsen, M.L., *Fluid Phase Equilib.*, **9**, 1 (1982). Michelsen, M.L., *Fluid Phase Equilib.*, **9**, 21 (1982). Baker, L.E., Pierce, A.C., Luks, K.D., *Soc. Pet. Eng. J.*, 731 (1982).

Optimization of Liquid-Liquid Equilibrium Critical Temperatures

The liquid-liquid phase separation of partially miscible liquids is an example of a phase transition that has a critical point. For any given system, the liquid-liquid phase equilibrium occurs over a certain temperature range only. In most cases the mutual solubility of the liquid phases rises with temperature until complete miscibility is reached. In addition some systems exist, where the mutual solubibility decreases with temperature. Thus a temperature range exists, which is bounded above by the upper critical solution temperature (**UCST**) and below by the lower critical solution temperature (**LCST**) . Any partially miscible liquid system will exhibit either one or both critical points. In reality however, the critical temperatures may be not be accessible by measurement if the UCST is above the bubble-point temperature of the mixture, or if the LCST is below the mixture's melting point. Nevertheless it is possible to thermodynamically calculate the UCST and LCST at such conditions. The predicted UCST and LCST values correspond to a virtual superheated or subcooled liquid in this case. If complete miscibility is present at both the upper and lower temperature boundary of an LLE, the liquid-liquid phase diagrams of partially miscible liquids show a "closed-loop" structure, which is bound by the UCST and LCST critical temperatures. Figure 2.3.7.3-1 below, shows an example of a binary system with a predicted closed-loop LLE pattern. Experimentally, the upper critical temperature lies within the liquid region of the mixture. The lower one however, does not. Hence the experimental data only show an UCST closing of the LLE curve but no LCST. As is also obvious from Figure 2.3.7.3-1, the closing of the miscibility gap is quite rapid, i.e. the temperature range of the actual closing is quite small. Thus a range of LLE calculations over a given temperature grid needs to have a quite small temperature stepsize to be able to properly describe the closing area. The exact determination of a critical temperature thus may become quite laborious and time-consuming.

To make this task somewhat easier, COSMO*therm* offers automatic procedures to compute the critical closings of binary miscibility gaps in terms of UCST, LCST, and closed-loop LLE curves. Given in the context of a binary `LLE` calculation, option `OPT_UCST` will compute an upward grid of temperatures. Starting from the given input temperature, the optimization routine will sincrease the temperature in finite ΔT_{step} temperature steps and compute a LLE in each step. If the upper critical closing of the miscibility gap is reached, the steps will be refined iteratively and thus the UCST is optimized up to a given accuracy (in terms of a temperature threshold ΔT_{thresh}). Thus a LLE(T)-curve between the starting temperature and the optimized UCST is computed and written to the COSMO*therm* table file. Option `OPT_LCST` will compute a downward grid of temperatures. Starting from the given input temperature, the optimization routine will decrease the temperature in finite ΔT_{step} temperature steps and compute a LLE in each step. If the lower critical closing of the miscibility gap is reached, the steps will be refined iteratively and thus the LCST is optimized to the accuracy threshold ΔT_{thresh}. Thus a LLE(T)-curve between the starting temperature and the optimized LCST is computed and written to the COSMO*therm* table file. Option `OPT_LOOP` will compute a grid of temperatures in both directions. Starting from the given input temperature, the optimization routine will first increase the temperature by ΔT_{step} to find the UCST, which is iteratively refined up to ΔT_{thresh}. Subsequently the algorithm will decrease the temperature in finite ΔT_{step} temperature steps to find the LCST, which is also iteratively refined. Thus a complete closed-loop LLE(T)-curve between the optimized LCST and the optimized UCST is computed and written to COSMO*therm* table file. The default stepsize for the temperature grid calculation is ΔT_{step} = `10` K. It can be changed with option `LLE_TSTEP=`ΔT_{step}. The default accuracy threshold for the critical temperature optimization is ΔT_{thresh} = `0.1` K. It can be changed with option `LLE_TTHRESH=`ΔT_{thresh}.

Suboptions to `binary LLE` computations (optimization of LLE critical temperatures):

`OPT_UCST`	Optional for `binary LLE` calculations: optimize the upper critical solution temperature (UCST) of a LLE.
`OPT_LCST`	Optional for `binary LLE` calculations: optimize the lower critical solution temperature (LCST) of a LLE.
`OPT_LOOP`	Optional for `binary LLE` calculations: optimize the upper and lower critical solution temperature (UCST and LCST) of a LLE system with a closed-loop structure.
`LLE_TSTEP=`ΔT_{step}	Optional for `binary LLE` calculations with `OPT_UCST`, `OPT_LCST`, or `OPT_LOOP` optimization: Change the temperature stepsize for the automatic LLE critical point optimization. Argument ΔT_{step} is expected to be a positive real number temperature increment in [K]. Default is $\Delta T_{step} = 10$ K.
`LLE_TTHRESH=`ΔT_{thresh}	Optional for `binary LLE` calculations with `OPT_UCST`, `OPT_LCST`, or `OPT_LOOP` optimization: Change the temperature threshold for the automatic LLE critical point optimization. Argument ΔT_{thresh} is expected to be a positive real number temperature in [K]. Default is $\Delta T_{thresh} = 0.1$ K.

Renormalization of Liquid-Liquid Equilibrium Computations

As noted in the previous section, the liquid-liquid phase separation of partially miscible liquids is an example of a phase transition that has critical points, which may be an upper or lower critical solution temperature (**LCST** or **UCST**). Figure 2.3.7.3-1 shows an example of a binary system with a closed-loop LLE pattern, where experimentally, the upper critical temperature lies within the liquid region of the mixture, and the lower one does not. The plot also shows the COSMO*therm* prediction for this binary system. What is strikingly obvious is that the prediction, although qualitatively correct in showing a closed loop structure, is way off when it comes to the quantitive location of the actual upper and lower critical solution temperatures. For the UCST the predicted critical temperature is almost 60 K higher than the actual experimental value. The direct COSMO*therm* prediction strongly overestimates the UCST, while the LCST is underestimated, although typically not as badly as the UCST is overestimated. This kind of over- and underestimation of the UCST and LCST is not a random prediction error of COSMO*therm*. The deviations are systematic and there are thermodynamic reasons for the wrong predictions: pure as well as mixture systems with exhibit striking thermodynamic anomalies near a critical point. The reason for these anomalies, which are actual macroscopic measurable phenomena, are thermodynamical fluctuations (of density, concentration, compressibility, …), which become increasingly long-lived if a critical region is approached and are most pronounced at the critical point[48,49,50]. A well known example is the clouding of a gas when it becomes supercritical, which is due to liquid-like fluctuations within the vapor phase that are strong enough to scatter light[49]. In the case of liquid-liquid phase separations thermodynamic fluctuations in the composition space around the critical point lead to an early "apparent" closure of the measured LLE curve as compared to the actual liquid-liquid equilibrium compositions *x´(i)* and *x´´(i)* as computed by theoretico-computational prediction methods. The LLE equilibrium compositions can not be observed any more if the temperature approaches a critical point because the thermodynamic fluctuations in the liquid blur the phase separation of the mixture, which as a consequence appears to be miscible at temperatures below the UCST, and above the LCST, respectively. Hence the temperature range where a miscibility gap can be observed by measurements is always (and often considerably) smaller than the theoretical temperature range of the miscibility gap as defined by the thermodynamic equilibrium, see Figure 2.3.7.3-1.

---

[48] Klein, M.J.; Tisza, L. *Phys. Rev.* **76**, 1861 (1949).

[49] Greer, S.C.; Moldover, M.R. *Ann. Rev. Phys. Chem.* **32**, 233 (1981).

[50] Sengers, J.V.; Levelt-Sengers, J.M.H. *Ann. Rev. Phys. Chem.* **37**, 189 (1986).

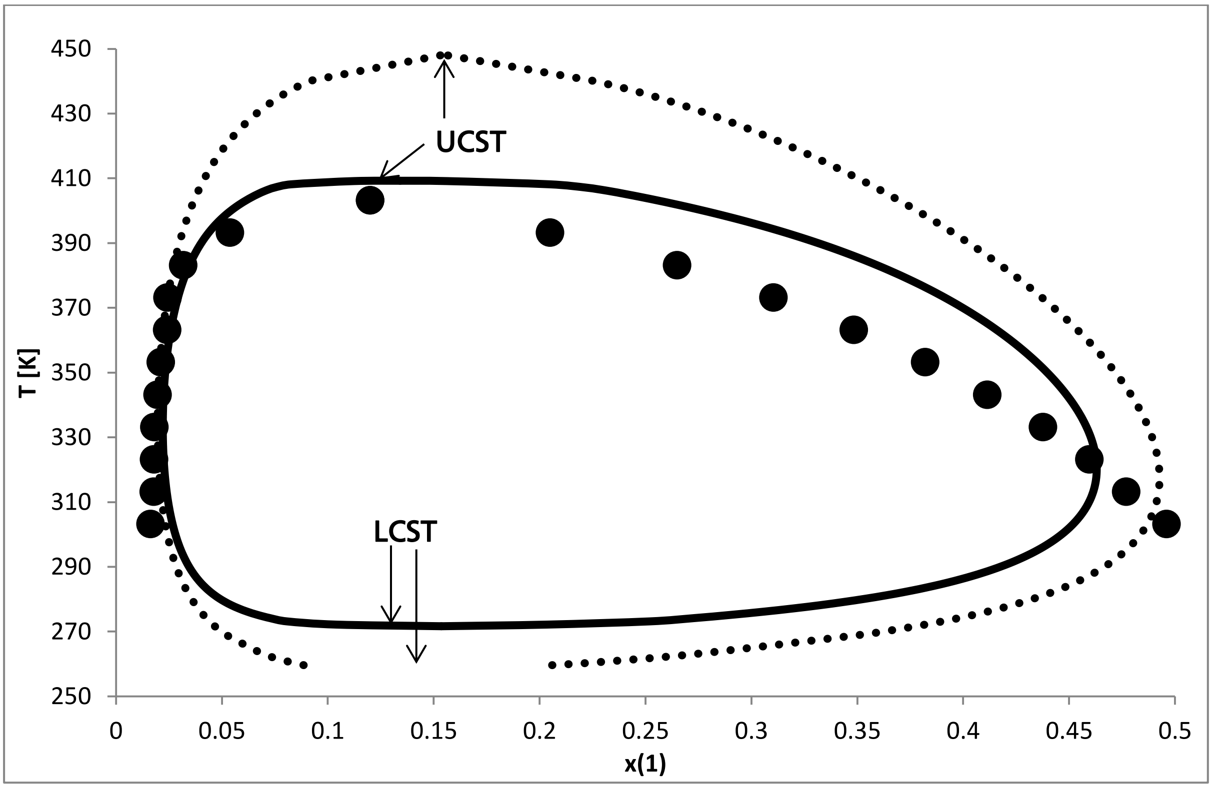


Figure 2.3.7.3-1: Closed loop liquid-liquid equilibrium (LLE) diagram of binary system 1-butanol (1) – water (2). Circles are experimental LLE points, dashed lines are COSMO*therm* predictions (BP-TZVPD-FINE level, C30-1701 parameterization), and solid lines are COSMO*therm* predictions after renormalization. The upper and lower critical solution temperatures of the experimental and predicted LLE are marked.

This is well-established knowledge in chemical thermodynamics and holds for activity coefficient models like Wilson's, van Laar, NRTL as well as for lattice-fluid based G^E-models like UNIQUAC and UNIFAC[51]. The common problem of these models and COSMO-RS is that they are based on a "mean field" assumption, namely that a liquid mixture has the same homogeneous mole fraction x_i on all length scales of a phase. A quantitative descriptor of this behavior that is widely used in statistical physics is the critical exponent β. The width *w* of a LLE(T) miscibility gap can be expressed in terms of β:

$$w(T) \approx \left(\frac{T-T_c}{T_c}\right)^{\beta} \tag{2.3.7-14}$$

The value of the critical exponent β (the order parameter of the system that vanishes at temperatures $T > T_c$) is a quantitative measure of the closing behavior of the LLE at the critical temperature T_c. The critical exponent β is almost universal. It only depends on a small number of basic properties of a given system, namely its dimensionality and the range of the interactions in the system[52]. Real liquids, real gases as well as liquid-gas mixtures are known to possess a critical exponent in the order of β = 0.33[53]. The activity

---

[51] De Pablo, J.J.; Prausnitz, J.M. *AIChE J.* **34**, 1595 (1988).

[52] Griffiths, R.B. *Phys. Rev. Lett.* **24**, 1479 (1970).

[53] Cardy, J. *Scaling and Renormalization in Statistical Physics*, Cambridge University Press (1996).

coefficient and G^E-models described above, however have a critical exponent of $\beta = 0.5$ at the closing of the LLE(T) temperature curve[51].

Thus in reality the closing of miscibility gaps is much faster than in mean-field γ-models, G^E-models, or COSMO-RS. This leads to poor predictions of the mutual solubilities in liquids, and also to an often poor qualitative description of the shape of the LLE(T) by these models. Predicted miscibility gaps are too wide, i.e. solubilities are too low, and as a consequence UCSTs are predicted too high and LCSTs too low. As noted above, the physical reasons for the problems of the mean field models at the critical point are thermodynamic fluctuations of composition and density. While the total free energy of the mixture under the model assumption of homogeneous phases with fixed composition still leads to phase separation in the equilibrium, even though the free energy barrier between the two phases is quite small near the critical point, it is the fluctuations that lead to an apparent miscibility in the real system – simply because the free energy barrier between the two phases is so small that it can be overcome by the fluctuations.

The practical consequences of these thermodynamic fluctuations on the actual measurement and prediction of phase diagrams with critical points however, are rarely acknowledged in physical chemistry and chemical engineering literature and textbooks. In standard γ-models, G^E-models, and in COSMO-RS the fluctuations are neglected. If the apparent upper or lower critical temperature is known, the LLE phase diagram can be scaled accordingly[50,51]. Unfortunately in *a priori* prediction methods like COSMO-RS and COSMO*therm*, the apparent critical temperatures are not known.

Some finding of solid-state physics however, offer a pathway towards a fundamental theoretical treatment of thermodynamic fluctuations within COSMO-RS theory: Critical phenomena of liquids and gases belong to same universality class as the magnetization of ferromagnetic and anti-ferromagnetic solids. Thus the critical behavior of lattice fluid models (a class that COSMO-RS and the γ- and G^E-models described above belong to) can be described in a similar way that solid-state physics models describe the critical behavior of the magnetization process[54]. In solid-state physics it was realized quite early that the spontaneous magnetization of bulk ferromagnetic solids has a critical exponent significantly lower than 0.5. This phenomenon can only be described correctly by statistical mechanics if spin fluctuations are taken into account[55]. The resulting Ising model can be solved analytically for special cases[55], or numerically to very good accuracy. For the spontaneous magnetization of a cubic lattice a critical exponent of $\beta = 0.325$ has been found, which is in well agreement with experimental findings[56]. As has been pointed out by Goldstein and Walker[54], it is possible to apply Ising's spin model to LLE phase diagrams if they are symmetric with respect to the concentration. In general however, LLE(T) curves are not symmetric with respect to the mole fraction concentrations of the miscibility gap $\underline{x}'$ and $\underline{x}''$. To be able to use the Ising model in the renormalization of COSMO-RS predicted LLE, hence a symmetrization procedure had to be introduced, which transforms the given COSMO-RS predicted LLE concentrations in mole fractions $\underline{x}'$ and $\underline{x}''$ into a coordinate system where the LLE is symmetric[57]. For a LLE computed in terms of miscibility gap mole fraction vectors $\underline{x}'$ and $\underline{x}''$ with upper and lower composition bounds $\underline{x}_{max}$ and $\underline{x}_{min}$, the transformation to a locally symmetric coordinate system z is:

---

[54] Goldstein, R.E.; Walker, J.S. *J. Chem. Phys.* **78**, 1492 (1983).

[55] Ising, E. *Z. Phys.* **31**, 253 (1925).

[56] Ito, N.; Hukushima, K.; Ogawa, K.; Ozeki, Y. *J. Phys. Soc. Jpn.* **69**, 1931(2000).

[57] Klamt, A.; Eckert, F.; Kaven, F.; *in preparation*.

$$z(\tilde{x}_i) = \frac{\tilde{x}_i}{\tilde{x}_i + f_x(1-\tilde{x}_i)} \text{ with } \tilde{x}_1 = \frac{\|\underline{x}' - \underline{x}_{min}\|}{\|\underline{x}_{max} - \underline{x}_{min}\|}, \ \tilde{x}_2 = \frac{\|\underline{x}'' - \underline{x}_{min}\|}{\|\underline{x}_{max} - \underline{x}_{min}\|}, \text{ and } f_x = \sqrt{\frac{\tilde{x}_1 \tilde{x}_2}{(1-\tilde{x}_1)(1-\tilde{x}_2)}} \qquad (2.3.7\text{-}15)$$

The resulting **LLE renormalization** is based upon the LLE miscibility gap width w :

$$w_R = max\left[0; w_z - e^{\left(6.16 - 13.095 w_z - e^{(145.26 - 260 w_z)}\right)}\right] \qquad (2.3.7\text{-}16)$$

This description has the advantage that it depends only on the width of the symmetrized miscibility gap $w_z = 1 - 2z(\tilde{x}_1)$. I.e. no data of the actual critical point has to be known in advance. Not even the temperature is included. Moreover, the parameters of the renormalization are fixed, as they result from a simple exponential fit of the symmetrized miscibility gap width of COSMO-RS (with critical exponent β = 0.5) to the one of the cubic lattice Ising model (with critical exponent β = 0.325). This implies that eq. 2.3.7-16 can be applied not only to COSMO*therm* LLE predictions, but to any LLE(T) prediction computed from a "mean-field" activity coefficient or G^E-model that has a critical exponent of β = 0.5. Thus the Ising model derived renormalization model decribed by equations 2.3.7-15 and 2.3.7-16 is completely general and does not include any adjustable parameters[57].

In COSMO*therm* the command **`RENORM`** as a suboption of the `binary`, `ternary`, or `multinary` options `LLE` and `LLE_NEW`, toggles the renormalization of the LLE phase diagram. If the `RENORM` option is activated, COSMO*therm* will produce an additional line (in case of `binary LLE` or `LLE_NEW` computations) or an additional table (in case of `ternary` or `multinary LLE` computations) in the output table file, which holds the renormalized LLE points $\underline{x}'_R$ and $\underline{x}''_R$. If the `$IL` option was used in `binary` or `ternary` computation, only the LLE points converted to the laboratory (IL-BINARY) reference frame will be renormalized (for more information on the `$IL` option and the reference frame conversion, please confer sections 2.3.7.5 and 4.9 of this manual). The renormalized LLE for the example system water – butoxyethanol is also given in Figure 2.3.7.3-1 above. Although not a perfect quantitative match with the experiment it can be seen that the shape and temperature range of the renormalized LLE loop is considerably closer to the measured one. Also the renormalized critical temperatures LCST and UCST are closer to the observed ones.

Suboptions to `binary`, `ternary`, or `multinary LLE` computations:

`RENORM`	Optional for `binary`, `ternary`, or `multinary LLE` calculations: compute the renormalized LLE points accounting for thermodynamic fluctuations in the liquid mixture.

2.3.7.4 Solid-Liquid Equilibrium Computation for Binary Mixtures

For the **`binary`** option, COSMO*therm* also offers the possibility to automatically search for points of solid-liquid-equilibrium (SLE) via the keyword **`SLE`**. If this command is given in the same line as the `binary` command, COSMO*therm* will search the computed mixtures for possible concentrations of solidification and if found, writes them to the COSMO*therm* output file. If this option is used, COSMO*therm* will compute the thermodynamic properties of the binary mixture at 325 additional mixture concentrations that are distributed on an even spaced grid. The thermodynamic properties of the additional mixtures are written to the COSMO*therm* output file and, in tabulated form to the COSMO*therm* table file. The SLE properties are calculated from the solid-liquid phase equilibrium condition eq. 2.3.7-17:

$$\begin{gathered} G_i^{Solid} = G_i^{Solution} \\ \mu_i^{pure} + \Delta G_{fus}(T) = \mu_i^{Solution} + RT\ln(x_i) \end{gathered} \tag{2.3.7-17}$$

Please note that the SLE search using eq. 2.3.7-17 assumes that there is a simple eutectic point in the binary mixture. Complicated systems with several phase transitions in the solid state can not be predicted by the `SLE` option. The SLE search algorithm will solve the solid-liquid equilibrium condition of eq. 2.3.7-17 on a grid of 325 mixture concentrations points of the binary system (it is the same grid that is used for the `binary lle_new` option described in the previous section). If the equilibrium condition 2.3.7-17 was met on the concentration grid, then COSMO*therm* will perform an additional iterative refinement of the SLE points. By default the iterative refinement will compute the SLE points up to an accuracy threshold of x_i = 10^{-5} mole fractions within a maximum number of 150 iterations. These defaults can be changed by options `SLE-iter-thresh=value` for the accuracy threshold value, and `maxiter-SLE=value` for the maximum number of iterations. Please note that in addition to the solution of eq. 2.3.7-17, the binary grid is checked for the presence of a solid-liquid-liquid thermodynamic equilibrium (SLLE), which means that the virtual system of supercooled liquids of the given compounds shows a miscibility gap at the given conditions. If such a virtual LLE miscibility gap is present in a "SLLE" system, then the equilibrium condition 2.3.7-17 will have several solutions. In this case the SLE search algorithm will pick the correct solution (where the SLE point is outside the virtual miscibility gap) and start the iterative refinement of the SLE search from this point. Both LLE and the resulting SLLE points will be written to the of COSMO*therm* output and table file.

As noted above, the SLE option assumes that phase diagram of the binary system shows a simple eutectic point, i.e. a temperature where the solubility curves of the two binary species meet. Below the eutectic point all species involved are solid. Such a simple eutectic point can be determined by an automatic search procedure in COSMO*therm*. The search procedure is activated with `SLE` suboption `OPT_EUTECTIC[=direction]`, where argument `direction`, which can be `down`, or `up`, simply determines the search direction in the temperature. If `down` or no argument is given, the search for the eutectic point temperature will be done along decreasing temperature, for argument `up`, the search for the eutectic point temperature will done be along increasing temperature. The search for the eutectic point will start at the given input temperature. The search then will proceed with a given fixed temperature stepsize, which will be refined iteratively if the temperature is close to the eutectic point. The default initial stepsize for the temperature grid calculation is `ΔTstep = 10` K. It can be changed with option `LLE_TSTEP=ΔTstep`. The default accuracy threshold for the critical temperature optimization is `ΔTthresh = 0.1` K. It can be changed with option `LLE_TTHRESH=ΔTthresh`.

Because COSMO*therm* can only calculate compound in a liquid, for the solid-liquid equilibrium of solid compound with a solvent, the Gibbs free energy of fusion of the compound, ΔG_{fus} has also be taken into account. ΔG_{fus} can be given in the compound section of the COSMO*therm* input file via option `DGfus=value` (see section 2.2.1). A temperature dependent free energy of fusion can be calculated from experimental compound data such as enthalpy or entropy of fusion (ΔH_{fus} or ΔS_{fus} , respectively) and melting temperature T_{melt}:

$$\begin{aligned}\Delta G_{fus}(T) &= \Delta H_{fus}\left(1-\frac{T}{T_{melt}}\right) - \Delta Cp_{fus}\left(T_{melt}-T\right) + \Delta Cp_{fus} T \ln\frac{T_{melt}}{T} \\ \Delta G_{fus}(T) &= \Delta S_{fus}\left(T_{melt}-T\right) - \Delta Cp_{fus}\left(T_{melt}-T\right) + \Delta Cp_{fus} T \ln\frac{T_{melt}}{T}\end{aligned} \qquad (2.3.7\text{-}18)$$

Thus, the usage of compounds melting temperatures and enthalpy or entropy and optionally heat capacity of fusion allows the automatic calculation of the compounds solid-liquid equilibrium at different temperatures with the `binary SLE` option. ΔH_{fus} or ΔS_{fus} can be given in the compound section of the COSMO*therm* input file via option `DHfus=value` or `DSfus=value`, respectively (see section 2.2.1). Optionally the heat capacity of fusion ΔCp_{fus} can also be used to decribe the temperature dependency of the Gibbs free energy of fusion. ΔCp_{fus} can be given in the compound section of the COSMO*therm* input file with option `Dcpfus=value` (see section 2.2.1). If no `Dcpfus` value is given it is assumed zero. A compounds melting temperature T_{melt} can be given in the compound section of the COSMO*therm* input file via option `Tmelt=temp` (see section 2.2.1). The ΔCp_{fus} estimate described in the solubility section 2.3.4, (keyword `Dcpfus_estimate`), may also be used in binary SLE computations. However, following the considerations taken in section 2.3.4, the ΔCp_{fus} estimate should only be used the prerequisites described there are met. As an alternative to the definition of the free energy of fusion described above the `binary SLE` option also allows the classification of one or both binary phases as liquid. This can be done with the `liquid[=l]`keyword given in the compound line of the `binary SLE` option, where argument `l` is the identifier of the binary phase (i.e. `l` can be 1 or 2, the first or the second phase given in the binary option). If no argument is given both phases are considered to be liquid. In this case only the LLE's can be found in a possible SLLE equilibrium. For a phase thus designated as liquid ΔG_{fus} is defined as zero at any temperature. This slightly artificial definition - in reality ΔG_{fus} always will be zero at some (low) temperature - serves the purpose of a simple shortcut for solvents, e.g. if only one side of the SLE is looked at and the other side is considered a liquid solvent with unknown ΔG_{fus}. In addition to the SLE computation of two neutral compounds, the `binary SLE` option allows for the definition of a multicomponent phase (i.e. one of the phase is defined as a ionic liquid, a salt or a cocrystal). This requires the additional input of two types of data: the definiton of the multicomponent phase, and the input of the multicomponent phase free energy of fusion. The definition of the multicomponent phase is identical to the input of a ionic liquid phase in a binary VLE phase diagram, which is described in section 2.3.7.5. The input of the multicomponent phase free energy of fusion is identical to the input of a salt free energy of fusion described in section 2.3.4.

Suboptions to `binary SLE` computations:

`Dcpfus_estimate`	Optional for the input of a temperature dependent compound free energy of fusion (via input of enthalpy or entropy of fusion ΔH_{fus} or ΔS_{fus}, and melting point T_{melt}): toggle the approximation of the heat capacity of fusion as $\Delta Cp_{fus} = \Delta S_{fus} = \Delta H_{fus}/T_{melt}$. The value of $\Delta G_{fus}(T)$ thus obtained wil be used to compute the SLE of the binary system. If the `Dcpfus_estimate` keyword is given in binary mixture SLE computation input line, the approximation to ΔCp_{fus} is valid for all compounds and the given mixture line.
`SLE-iter-thresh=value`	Optional for a `SLE` computation: give the accuracy threshold for the iterative refinement of the SLE point calculation. Argument `value` is expected to be a nonzero positive real number.Default is `SLE-iter-thresh=1.0E-5`.
`maxiter-SLE=value`	Optional for a SLE computation: give the maximum number of iterations for the iterative refinement of the SLE point calculation. Argument `value` is expected to be a nonzero positive integer number.Default is `maxiter-SLE=150`.
`liquid[=l]`	Optional for a `SLE` computation: Define one or both binary phase(s) as liquid phases with ΔG_{fus}(T)=0. If no argument is given, both phases are defined as liquid. The value of argument `l` is defined as the binary phase number, i.e. it can be `1` or `2`, where `1` defines the first phase `i` given in the `binary={i j}` option as liquid, whereas `2` defines the second phase `j` given in the `binary={i j}` option as liquid.
`OPT_EUTECTIC[=direction]`	Optional for binary `SLE` calculations: optimize the eutectic point temperature of a SLE system. The optional argument `direction` either can be `down`, which triggers a "downward" search with decreasing temperature, or `up`, which triggers an "upward" search with increasing temperature
`LLE_TSTEP=`ΔT_{step}	Optional for binary `SLE` calculations with `OPT_EUTECTIC` optimization: Change the temperature stepsize for the automatic SLE eutectic point optimization. Argument ΔT_{step} is expected to be a positive real number temperature increment in [K]. Default is $\Delta T_{step} =$ `10` K.
`LLE_TTHRESH=`ΔT_{thresh}	Optional for binary `SLE` calculations with `OPT_EUTECTIC` optimization: Change the temperature threshold for the automatic SLE eutectic point optimization. Argument ΔT_{thresh} is expected to be a positive real number temperature in [K]. Default is $\Delta T_{thresh} =$ `0.1` K.

2.3.7.5 Treatment of Ionic Liquids, Salts, and Complexes in Binary and Ternary Mixture Computations

COSMO*therm* also offers the possibility to compute phase diagrams of **`binary`** or **`ternary`** mixtures of a Ionic Liquid (IL) phase with additional solvent phases. Because in COSMO-RS theory any Ionic Liquid or dissoluted salt phase has to be treated by means of the individual ions forming the IL or salt, a binary phase diagram computation has to be conducted in the form of a "pseudo-binary" multicomponent phase diagram with the boundary condition of the anion and cation concentrations forming the IL or salt have a fixed ratio according to the IL / salt stoichiometry. This means that within COSMO*therm*, the IL / salt is treated by means of the individual ions, but on output, the results of the individual ion's properties are combined to form a single IL or salt phase. Please note that is also possible to include neutral components into the definition of the IL or salt phase. Thus molecular complexes and related multicomponent associates, where two or more distinct molecular components are assumed to be tied together (e.g. salts like gypsum $CaSO_4 \cdot 2H_2O$ where the ionic components are associated with one or several solvent molecules), can be treated with the **`binary`** and **`ternary`** IL option as well. In such a case - i.e. if a salt like gypsum is dissoluted in solvent water, which also is a component of its salt stoichiometry – COSMO*therm* will solve the thermodynamic equilibrium condition for this system and notify this in the output table of the solubility.

In practice, a binary VLE, LLE or SLE computation containing an IL or salt phase is toggled by the command **`binary={i $IL}`** command, where `i` is the number of the "solvent" phase, i.e. a neutral solvent compound (with number i as given in the sequence of compounds in the compound input section) and **`$IL`** denotes the "IL" or "salt" phase as defined by the `IL` and `IL_n` input options, below. Alternatively, the **`nbinary={name`$_i$ `$IL}`** command can be used, where `name`$_i$ is the name of the "solvent" phase compound as given in the compound input section and **`$IL`** denotes the "IL" or "salt" phase. If the **`$IL`** keyword is given in the `binary` or `nbinary` input, denoting a IL or salt phase to be present in the pseudo-binary phase equilibrium, it is necessary to define the IL or salt phase via the individual ions that form the specific IL or salt. The IL or salt is pieced together from its anion(s) and cation(s) and thus its composition has to be defined in the input. This is possible with the **`IL={i j …}`** command, where `i`, `j`,… are the numbers of the ion compounds (numbers as given in the sequence of compounds in the compound input section) or the **`nIL={name`$_i$ `name`$_j$ `…}`** command, where `name`$_i$, `name`$_j$, … are the names of the ion compounds as given in the compound input section. In addition the IL's or salts stoichiometry has to be defined for the individual anion and cation compounds: The input of the IL stoichiometry factors ν_i, ν_j is possible via the **`IL_n={`ν_i ν_j `…}`** command, where ν_i, ν_j,… are the stoichiometry numbers of the IL defined by the ions as given by the `IL={i j …}` or `nIL={name`$_i$ `name`$_j$ `…}` command. For example the IL $[bmim]^+[BF_4]^-$ would be defined by the combination of the option `nIL={1-butyl-3-methyl-imidazolium0 bf4}`, defining the IL composition from bmim-cation (`1-butyl-3-methyl-imidazolium0.cosmo`) and tetrafluoroborate-anion (`bf4.cosmo`), and the option `IL_n={1 1}`, defining the stoichiometry of the IL or salt (a simple equimolar two-component IL with one bmim-cation and one BF_4-anion in this case). If these options are given as additional input to an automatic binary VLE computation ("`binary={i $IL}`" or "`nbinary={name`$_i$ `$IL}`") input line, COSMO*therm* will compute the pseudo-binary phase diagram of the IL phase of the given composition and stoichiometry with the given "solvent" phase via eqs. 2.3.7-1 and 2.3.7-2, and write the result to the COSMO*therm* output and table file. Note, that the computed properties of the IL or salt compounds will be written output file by means of their individual ion properties, while the table file will contain the mixture property table in its usual "binary" form, where the properties of the ions are summed up to a single IL or salt phase property. It is possible to define up to two separate IL phases in one `binary` computation. In the COSMO*therm* `binary` input up to two `$IL` descriptors denoting "IL" phases may be given. E.g. input of the `binary={$IL $IL}` command describes a system with two IL phases. If more than one IL phase is given via

`$IL` keys in the `binary={i j}`, or `nbinary={name_i name_j}` options, the IL phases have to be defined by subsequent groups of `IL={k l …}` or `nIL={name_k name_l …}` and `IL_n={ν_k ν_l …}` input keys. The first `IL` / `IL_n` group found in the ternary mixture input line is assigned to the first `$IL` key in the `binary={…}` input, the second `IL` / `IL_n` group found in the input line is assigned to the second `$IL` key in the `binary={…}` input.

The vapor pressure of an IL phase is computed from the partial vapor pressure contributions of each of the components of the IL phase in terms of their activity in the overall mixture. By default a `$IL` phase is assumed to show "ideal" mixing behavior. This means that the overall activity of a pure IL phase is one and the vapor pressure of the IL phase is not computed from the activities of the IL phase components but from their relative mixture ratio in the IL phase only. Alternatively, it is possible to define the IL phase in a way that it shows "real" mixing behavior via input option `ILphase=REAL`. If the `ILphase=REAL` option is activated the vapor pressure of the IL phase is computed from the contribution of the IL phase components with respect to their activities based on the concentrations derived from the ratio of the components stoichiometry, as well as to the overall concentrations of all compounds in the given binary phase. This means that now the IL components behave like a "real" stoichiometric mixture of the IL phase's components. A composite IL phase with nonzero vapor pressure (e.g. a liquid salt-solvent associate) thus defined as "real" phase will have the same partial vapor pressure in the `binary $IL` system as in a `multinary` (see section 2.3.7.7) system defined with the same relative and absolute concentrations of the components.

In addition to the regular VLE phase diagram computation it is also possible to compute liquid-liquid phase equilibrium (LLE), i.e. phase separation / miscibility gaps for pseudo-binary solutions with a Ionic Liquid or salt phase. An IL-LLE computation is toggled with the keywords **`lle`** or **`lle_new`**, the same way it is done for binary mixtures of non-ionic compounds (see section 2.3.7.3 "Liquid-Liquid Equilibrium Computation for Binary Mixtures"). Please note however, that for IL-LLE's some care has to be taken in the interpretation of the COSMO*therm* results for the computed miscibility gap mole fractions. To compare the computed IL or salt mole fraction with experimental data, depending on the reference state of the miscibility gap measurement, it may be necessary to convert the computed mole fractions along the guidelines given in section 4.9 "Ionic Liquids" of this manual: The "pseudo-binary" definition of the mole fraction, where x_{IL} is defined as the sum of the ions mole fractions $x_{IL} = \sum x_i^{ion}$ is different from the "laboratory-binary" definition, where the Ionic Liquid is considered to be one single compound. This means that for a A^-C^+-IL 1 mol of the IL A^-C^+ in the "laboratory-binary" framework corresponds to 1 mol A^- and 1 mol C^+ in the pseudo-binary" framework. The conversion between the "pseudo-binary" framework as used by COSMO*therm* and the "laboratory-binary" framework is explained in section 4.9. If an LLE is computed with the `LLE` or `LLE_NEW` option and one phase is defined as Ionic Liquid, COSMO*therm* will do this conversion automatically and print the mole fraction of the LLE as additional lines to the output- and table-file:

```
LLE point found at x`(1) = 0.20067648  x`(2) = 0.79932352 and x``(1) = 0.99707420  x``(2) = 0.00292580  (T =  298.15 K)
LLE      IL-BINARY  x`(1) = 0.33427236  x`(2) = 0.66572764 and x``(1) = 0.99853496  x``(2) = 0.00146504  (T =  298.15 K)
Spinodal LLE point x`(1) = 0.51250000  x`(2) = 0.48750000 and x``(1) = 0.97250000  x``(2) = 0.02750000  (T =  298.15 K)
Spinodal IL-BINARY x`(1) = 0.67768595  x`(2) = 0.32231405 and x``(1) = 0.01394170  x``(2) = 0.98605830  (T =  298.15 K)
```

The first and third lines are describing the binodal and spinodal LLE points in the "pseudo-binary" framework, in analogy to the LLE computations of neutral compounds. The second and fourth lines denoted by the phrase "`IL-BINARY`" are the additional output lines that are given only for IL-LLE computations. They contain the binodal and spinodal LLE points converted to the "laboratory-binary" framework, respectively. Please note that this conversion between "pseudo-binary" and "laboratory-binary" framework is done for the values of the binodal and spinodal mole fractions of the miscibility gap only. The body of the binary phase diagram table uses the "pseudo-binary" definition throughout.

It is also possible to compute solid-liquid phase equilibria (SLE) for pseudo-binary solutions with an Ionic Liquid or salt phase. An IL-SLE computation is toggled with the keyword **`SLE`**, the same way it is done for binary mixtures of non-ionic compounds (see section 2.3.7.4 "Solid-Liquid Equilibrium Computation for Binary Mixtures"). To do an IL-SLE computation it is necessary to provide experimental data that can be used to compute the free energy of fusion of the IL / salt phase via eq. 2.3.7-18 and which subsequently can be used in eq. 2.3.7-12 to compute the SLE. The input of the required experimental data is equivalent to the input of the salt free energy of fusion used for the computation of solid salt solubilities (see section 2.3.4 "Automatic Solubility Calculation" of this manual): the IL / salt heat of fusion data input has to be done in the same mixture line where the `SLE`, `salt` and `salt_n` commands are located. The net Gibbs free energy of fusion of the salt can be given by options `DGfus_salt` or `DGfus_salt_SI`. The net enthalpy of fusion of the salt as used in eq. 2.3.7-18 can be given by options `DHfus_salt` or `DHfus_salt_SI`. The net entropy of fusion of the salt as used in eq. 2.3.7-18 can be given by options `DSfus_salt` or `DSfus_SALT_SI`. The net heat capacity of fusion of the salt as used in equation 2.3.7-18 can be given by options `Dcpfus_salt` or `Dcpfus_SALT_SI`. The net melting temperature of the salt as used in equation 2.3.7-18 can be given by options `Tmelt_salt`, `Tmelt_salt_C` or `Tmelt_salt_K`. (see below for details). The ΔCp_{fus} estimate described in the solubility section 2.3.4, (keyword `Dcpfus_estimate`), may also be used in binary salt SLE computations. However, following the considerations taken in section 2.3.4, the ΔCp_{fus} estimate should be used with great caution if salts or Ionic Liquids are involved. All of these input options expect "net" salt data values that are defined for the salt as defined in composition and stoichiometry by the `IL` and `IL_n` commands. Thus no individual ion heat of fusion data can be used for SLE computations. Please note that similar to considerations on salt solubility or Ionic Liquid activity coefficient calculations (see sections 2.3.4 and 4.9 of this manual, respectively), for IL-SLE computations some care has to be taken in the interpretation of the COSMO*therm* results for the computed SLE mole fraction: To compare the computed salt mole fraction with experimental data, depending on the reference state of the SLE measurement, it may be necessary to convert the computed SLE mole fraction value along the guidelines given in section 4.9 "Ionic Liquids" of this manual.

What has been noted above for `binary` IL computations also holds for **`ternary`** IL computations - with a few important differences, however: the `ternary` option does not allow solid-liquid equilibrium computations (`SLE` option), nor is it possible to search for spinodal liquid-liquid equilibrium points (`lle_new` option). The regular (binodal) liquid-liquid equilibrium search (`LLE` option, see section 2.3.7.3) however, is possible in combination with the `ternary` option for Ionic Liquids. For the `ternary` option it is possible to define up to three separate IL phases in one `ternary` computation. In the COSMO*therm* `ternary` input up to three `$IL` descriptors denoting "IL" phases may be given. E.g. input of the `ternary={$IL i $IL}` command describes a system with two IL phases and one neutral solvent phase, where `i` is the number of the "solvent" compound, i.e. a neutral solvent compound (with number i as given in the sequence of compounds in the compound input section). If more than one IL phase is given via `$IL` keys in the `ternary={i j k}`, or `nternary={name`$_i$ `name`$_j$ `name`$_k$`}` option, the IL phases have to be defined by subsequent groups of `IL={i j …}` or `nIL={name`$_i$ `name`$_j$ `…}` and `IL_n={`ν_i ν_j `…}` input keys. The first `IL` / `IL_n` group found in the ternary mixture input line is assigned to the first `$IL` key in the `ternary={…}` input, the second `IL` / `IL_n` group found in the input line is assigned to the second `$IL` key in the `ternary={…}` input, and so on.

If the liquid-liquid equilibrium search (`LLE` option, see section 2.3.7.3) is combined with the `ternary` option for Ionic Liquids, three table blocks will be printed to the COSMO*therm* table file. First the regular VLE output table with the mole fraction concentrations of the grid points (and all computed themrdynamic properties such as H^E, G^E, p^{tot} at the grid points) is printed, followed by a table with the results of the LLE searches, each started at the concentration grid points given in the upper table. The LLE results table holds the two tie point concentrations *x´(i)* and *x´´(i)* for the three phases *i* = 1, 2, 3. If no LLE was found the values of *x´(i)* and *x´´(i)* are set to zero. The following columns of the LLE results table hold the logarithm of the activity coefficients at the tie points (blank if no LLE was found) and the temperature and total pressure at the tie points. Note that the second table holds the LLE ouput in the points in the "pseudo-ternary" framework, analogue to the ternary LLE computations of neutral compounds, meaning that the tie point concentrations *x´(i)* and *x´´(i)* of an IL phase are the sum of the concentrations of the individual ions the IL phase consists of. Finally, a table block with the LLE results (tie point concentrations *x´(i)* and *x´´(i)* and activity coefficients) as converted to the "laboratory-ternary" framework is printed to .the COSMO*therm* table file (see section 4.9 for details on this conversion). A typical table output of a `ternary` `LLE` computation with an IL phase is shown below:

```
Results for ternary mixture of benzene_c0 (1) + hexane_c0 (2) + [C4mim][PF6] (3) at T =  298.15 K - Job 1

         x1          x2          x3          H^E          G^E         ptot  mu1+RTln(x1)  mu2+RTln(x2)  mu3+RTln(x3)    ln(gamma1)    ln(gamma2)    ln(gamma3)          y1          y2          y3
1.000000000 0.000000000 0.000000000   0.00000000   0.00000000 102.16898460   -3.81460925    0.00000000    0.00000000    0.00000000    1.04653169    2.79024157  1.00000000  0.00000000  0.00000000
0.750000000 0.125000000 0.125000000   0.33867408   0.25858434 225.85047178   -3.96440692   -7.10021110   -0.57912323    0.03484722    1.16748472    2.11502867  0.35131225  0.64868775  0.00000000
0.500000000 0.250000000 0.250000000   0.61168176   0.47925409 392.16695570   -4.15879224   -6.61017243   -0.40857815    0.11222056    1.30144556    1.70973469  0.14573198  0.85426802  0.00000000
0.250000000 0.375000000 0.375000000   0.84432998   0.67283480 608.44583106   -4.51181330   -6.28811740   -0.32275254    0.20952385    1.43955856    1.44912966  0.05176452  0.94823548  0.00000000
0.000000000 0.500000000 0.500000000   1.04847291   0.84574097 883.42330584    0.00000000   -6.03569512   -0.25427357    0.31650033    1.57792550    1.27702931  0.00000000  1.00000000  0.00000000

LLE results for multinary system - Job 1

    x`(1)     x`(2)     x`(3)    x``(1)    x``(2)    x``(3)  ln(g(x`(1)))  ln(g(x`(2)))  ln(g(x`(3))) ln(g(x``(1))) ln(g(x``(2))) ln(g(x``(3)))      y(1)      y(2)      y(3)           T          ptot
0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000                                                                                                                  298.150000      0.000000
0.3296383 0.0072257 0.6631359 0.8120831 0.1423945 0.0455224      0.916038      3.830002      0.157436      0.014427      0.849048      2.836205 0.4095036 0.5904964 0.0000000  298.150000    205.554713
0.2379508 0.0077927 0.7542565 0.6223949 0.3631266 0.0144785      1.094392      4.218238      0.080108      0.132884      0.376675      4.033178 0.2734147 0.7265853 0.0000000  298.150000    265.628087
0.1395657 0.0069690 0.8534653 0.3356997 0.6605997 0.0037006      1.292971      4.639976      0.027871      0.415293      0.088297      5.468692 0.1648833 0.8351167 0.0000000  298.150000    315.103100
0.0000000 0.0053971 0.9946029 0.0000000 0.9991867 0.0008133      1.574364      5.221049      0.000268      0.836858     -0.000028      7.109298 0.0000000 1.0000000 0.0000000  298.150000    364.375670

LLE results for multinary system converted to laboratory (Lx) frame (IL = single compound) - Job 1

   Lx`(1)    Lx`(2)    Lx`(3)   Lx``(1)   Lx``(2)   Lx``(3) ln(g(Lx`(1))) ln(g(Lx`(2))) ln(g(Lx`(3))) ln(g(Lx``(1))) ln(g(Lx``(2))) ln(g(Lx``(3)))     Ly(1)     Ly(2)     Ly(3)           T          ptot
0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000                                                                                                                     298.150000      0.000000
0.4931516 0.0108100 0.4960384 0.8309976 0.1457110 0.0232913      0.513217      3.427182     -0.245384      -0.008597       0.826024       2.813181 0.4095036 0.5904964 0.0000000  298.150000    205.554713
0.3820221 0.0125109 0.6054670 0.6269334 0.3657745 0.0072921      0.620977      3.744823     -0.393307       0.125618       0.369410       4.025912 0.2734147 0.7265853 0.0000000  298.150000    265.628087
0.2434566 0.0121566 0.7443868 0.3363220 0.6618243 0.0018537      0.736567      4.083573     -0.528532       0.413441       0.086445       5.466840 0.1648833 0.8351167 0.0000000  298.150000    315.103100
0.0000000 0.0107363 0.9892637 0.0000000 0.9995932 0.0004068      0.886599      4.533284     -0.687496       0.836451      -0.000435       7.108891 0.0000000 1.0000000 0.0000000  298.150000    364.375670
```

Ionic Liquid / salt binary and ternary phase diagram computation options:

`binary={i $IL}`
or
`nbinary={name_i $IL}`
	Optional for `binary` computations: Toggle Ionic Liquid / salt pseudo-binary phase diagram computation. The input of the Ionic Liquid pseudo-binary mixture is possible either via the `binary={i $IL}` command, where `i` is the number of the "solvent" phase, i.e. a neutral solvent compound (with number `i` as given in the sequence of compounds in the compound input section, arguments `i`.is expected to be a positive integer number) and the term `$IL` denotes the IL / salt phase, - or alternatively, by the `nbinary={name_i $IL}` command, where `name_i` is the name of the "solvent" phase compound as given in the compound input section and the term `$IL` denotes the IL / salt phase as defined by the `IL` and `IL_n` options given below.

`ternary={i j $IL}`
or
`ternary={name_i name_j $IL}`
	Optional for `ternary` computations: Toggle Ionic Liquid pseudo-ternary phase diagram computation. The input of the Ionic Liquid pseudo-ternary mixture is possible either via the `ternary={i j $IL}` command, where `i,j` is the number of the "solvent" phases, i.e. neutral solvent compound (with numbers `i,j` as given in the sequence of compounds in the compound input section, arguments `i`.is expected to be a positive integer number) and the term `$IL` denotes the IL / salt phase, - or alternatively, by the `nternary={name_i name_j $IL}` command, where `name_i` and `name_j` are the names of the "solvent" phase compounds as given in the compound input section and the term `$IL` denotes the IL / salt phases as defined by the `IL` and `IL_n` options given below. All three neutral solvent phases {`i j k`} or {`name_i name_j name_k`} may be replaced by a Ionc Liquid phase definition denoted by the `$IL` command.

Suboptions of the `binary={i $IL}` or `nbinary={name`$_i$ `$IL}`, and the `ternary={i j $IL}` or `nternary={name`$_i$ `name`$_j$ `$IL}` options:

`IL={i j …}`
or
`nIL={name`$_i$ `name`$_j$ `…}`
	Required for `binary` or `ternary` `$IL` computations: Define a Ionic Liquid / salt from individual anion and cation compounds. The input of the Ionic Liquid composition is possible either via their compound number (`IL={i j …}` command, where `i`, `j`,… are the numbers of the ion compounds numbers as given in the sequence of compounds in the compound input section, the arguments `i`, `j`,…are expected to be positive integer numbers) or via their compound number (`nIL={name`$_i$ `name`$_j$ `…}` command, where `name`$_i$, `name`$_j$,… are the names of the ion compounds as given in the compound input section).

`IL_n={`ν_i ν_j `…}`	Required for `binary` or `ternary` `$IL` computations: Define the Ionic Liquid / salt stoichiometry from individual anion and cation compounds. The input of the Ionic Liquid stoichiometry factors ν_i, ν_j is possible via the `IL_n={`ν_i ν_j `…}` command, where ν_i, ν_j,… are the stoichiometry numbers of the Ionic Liquid defined by the ions as given by the `IL={i j …}` or `nIL={name`$_i$ `name`$_j$ `…}` command. The stoichiometry factors ν_i, ν_j,… are expected to be positive integer numbers.
`ILphase=REAL`	Optional for binary or ternary $IL computations: Define the Ionic Liquid / salt / complex phase as "real" mixture with respect to activity and vapor pressure of the phase. The argument of the `ILphase` option is expected to be upcase.

Suboptions of the `SLE` and `binary={i $IL}` or `nbinary={name`$_i$` $IL}` option are:

`DGfus_salt=value`
or
`DGfus_salt_SI=value`
	Optional: Give the free enthalpy of fusion ΔG_{fus} for an IL / salt. For the `DGfus_salt=value` option, ΔG_{fus} is expected in [kcal/mol], for the `DGfus_salt_SI` option, ΔG_{fus} is expected to be in [kJ/mol]. Argument `value` is expected to be a real number.

`DHfus_salt=value`
or
`DHfus_salt_SI=value`
	Optional: Give the enthalpy of fusion ΔH_{fus} for an IL / salt. For the `DHfus_salt=value` option, ΔH_{fus} is expected in [kcal/mol], for the `DHfus_salt_SI` option, ΔH_{fus} is expected to be in [kJ/mol]. Argument `value` is expected to be a real number.

`DSfus_salt=value`
or
`DSfus_salt_SI=value`
	Optional: Give the entropy of fusion ΔS_{fus} for an IL / salt. For the `DSfus_salt=value` option, ΔS_{fus} is expected in [kcal/mol], for the `DSfus_salt_SI` option, ΔS_{fus} is expected to be in [kJ/mol]. Argument `value` is expected to be a real number.

`Dcpfus_salt=value`
or
`Dcpfus_salt_SI=value`
	Optional: Give the heat capacity of fusion ΔCp_{fus} for an IL / salt. For the `Dcpfus_salt=value` option, ΔCp_{fus} is expected in [kcal/mol], for the `Dcpfus_salt_SI` option, ΔCp_{fus} is expected to be in [kJ/mol]. Argument `value` is expected to be a real number.

`Tmelt_salt=temp`
or
`Tmelt_salt_C=temp`
or
`Tmelt_salt_K=temp`
	Optional: Give the melting temperature T_{melt} for an IL / salt. For the `Tmelt_salt=temp` and `Tmelt_salt_C=temp` options, T_{melt} is expected in [°C], for the `Tmelt_salt_K=temp` option, T_{melt} is expected in [K]. Argument `temp` is expected to be a real number.

`Dcpfus_estimate`	Optional for the input of a temperature dependent compound free energy of fusion (via input of enthalpy or entropy of fusion ΔH_{fus} or ΔS_{fus}, and melting point T_{melt}): toggle the approximation of the heat capacity of fusion as $\Delta Cp_{fus} = \Delta S_{fus} = \Delta H_{fus}/T_{melt}$. The value of $\Delta G_{fus}(T)$ thus obtained wil be used to compute the SLE of the given Ionic Liquid / salt system. If the `Dcpfus_estimate` keyword is given in binary mixture SLE computation input line, the approximation to ΔCp_{fus} is valid for all compounds including the given Ionic Liquid / salt in the given mixture line.

2.3.7.6 Treatment of Composite Phases in (Pseudo)-Binary/Ternary Mixture Computations

COSMO*therm* offers the possibility to compute phase diagrams of **`binary`** or **`ternary`** mixtures where each of two solvent phases can be defined as a composite with a given mixture concentration. This assumes that the "**composite**" or "**mixture**" phases consist of a mixture of compounds that are present at a given and fixed mixture ratio. Implicitly this approach also assumes that the physical and thermodynamic properties of the given mixture phases are additive with respect to the contributions of the mixtures components. Using mixture phases thus defined, COSMO*therm* can perform a "pseudo-binary" or "pseudo-ternary" multicomponent phase diagram computation. This means that within COSMO*therm*, the mixture phase is treated by means of its individual components at the given mixture ratio, but on output, the results of the individual compound properties are combined to form a single mixture phase. It is possible to include neutral components as well as ions into such a composite phase if it is assured that the resulting mixture is charge neutral.

A composite / mixture phase in a `binary` VLE, LLE or SLE calculation, or a `ternary` VLE, or LLE calculation is denoted by the placeholder keyword **`$MIX`**. Each or any of the `binary` or `ternary` phases can be defined as mixture phases by the `$MIX` placeholder. If one or two `$MIX` identifiers are given within the `binary={i $MIX}` or `nbinary={name`$_i$` $MIX}` options, COSMO*therm* expects to find the mixture concentrations of the composite (`$MIX`) phase to be in the same line as the `binary $MIX` input. The mixture concentrations can be given via keywords `xm={`x_1 x_2 x_3 …`}` (in mole fraction concentrations), or `cm={`c_1 c_2 c_3 …`}` (in mass fraction concentrations). If two `$MIX` phases are requested, COSMO*therm* expects two entries of the mixture phase concentration input `xm` or `cm`, which will be assigned to the `$MIX` phases simply by the sequence at which they are given in the mixture input line. The treatment of `ternary` phases diagrams is equivalent: if one, two, or three `$MIX` identifiers are given within the `ternary={i j $MIX}` or `nternary={name`$_i$` name`$_j$` $MIX}` options, COSMO*therm* expects to find the mixture concentrations of the composite (`$MIX`) phases to be in the same line as the `ternary $MIX` input, in the order in which they are given in the `ternary={}` input option. Please note that the mixtures of the composite phases are not allowed to overlap, neither with another pure compound phase, nor with another composite / mixture phase.

The vapor pressure of a mixture phase is computed from the partial vapor pressure contributions of each of the components of the IL phase in terms of their activity in the overall mixture. The mixture phase by default is assumed to show "ideal" mixing behavior. This means that the overall activity of a pure `$MIX` phase is one and the vapor pressure of the mixture phase is not computed from the activities of the mixture components but from their relative mixture ratio in the mixture phase only. Alternatively, it is possible to define the `$MIX` phase in a way that it shows "real" mixing behavior via input option `MIXphase=REAL`. If the `MIXphase=REAL` option is activated the vapor pressure of the mixture phase is computed from the contribution of the mixture phase components with respect to their activities based on the concentrations derived from the ratio of the components stoichiometry, as well as to the overall concentrations of all compounds in the given binary or ternary phase. Hence a mixture phase thus defined as "real" phase will show the same partial vapor pressure in the `binary` or `ternary $MIX` system as in a `multinary` (see section 2.3.7.7) system defined with the same relative and absolute concentrations of the components.

It is also possible to compute solid-liquid phase equilibria (SLE) for pseudo-binary solutions with composite phases. A `binary $MIX SLE` computation is toggled with the keyword **`SLE`**, the same way it is done for binary mixtures of pure compounds (see section 2.3.7.4 "Solid-Liquid Equilibrium Computation for Binary Mixtures"). To perform a `$MIX SLE` computation it is necessary to provide experimental data that can be used to compute the free energy of fusion of the composite / mixture phase via eq. 2.3.7-18 and which subsequently can be used in eq. 2.3.7-12 to compute the SLE. The input of the required experimental data is equivalent to the input of the salt free energy of fusion used for the computation of Ionic Liquid / salt SLE (see section 2.3.7.4, above): the composite / mixture phase heat of fusion data input has to be done in the same mixture line where the `SLE` and `$MIX` commands are located. The net Gibbs free energy of fusion of the mixture can be given by options `DGfus_mix` or `DGfus_mix_SI`. The net enthalpy of fusion of the mixture as used in eq. 2.3.7-18 can be given by options `DHfus_mix` or `DHfus_mix_SI`. The net entropy of fusion of the mixture as used in eq. 2.3.7-18 can be given by options `DSfus_mix` or `DSfus_mix_SI`. The net heat capacity of fusion of the mixture as used in equation 2.3.7-18 can be given by options `Dcpfus_mix` or `Dcpfus_mix_SI`. The net melting temperature of the mixture as used in equation 2.3.7-18 can be given by options `Tmelt_mix`, `Tmelt_mix_C` or `Tmelt_mix_K` (see below for details). The ΔCp_{fus} estimate described in the solubility section 2.3.4, (keyword `Dcpfus_estimate`), may also be used in binary mixture SLE computations. However, following the considerations taken in section 2.3.4, the ΔCp_{fus} estimate should be used with great caution if ions are involved in the mixture. All of these input options expect "net" mixture data values that are defined for the whole of the mixture as defined in composition by the `xm` or `cm` commands. Thus no individual heat of fusion data of the mixture components can be used for `SLE $MIX` computations.

An example for a SLE computation with a composite phase is given below. The pseudo-binary system consists of one single compound phase of aspirin (1) and a composite `$MIX` phase (2) which is an equimolar mixture of chloroform and acetone. I.e. the example input computes the SLE solubility of aspirin in a mixed solvent consisting of chloroform and acetone. Please note that the heat of fusion data input for the single / pure phase aspirin (1) is given in the compound input section or vap-file, as usual, while the input of the heat of fusion data of the composite phase (2) is given in the temperature / mixture line input:

```
f=aspirin_c0 tmelt_k=408.1500 DHfus=2.755
f=chcl3_c0
f=propanone_c0
tc=25 sle binary={1 $MIX} xm={0 0.5 0.5} tmelt_mix_K=180 dhfus_mix_si=5.7
```

Options for pseudo-binary phase diagram computations with composite / mixture phases:

`binary={i $MIX}`
or
`nbinary={name`$_i$` $MIX}`
	Optional for `binary` computations: Toggle composite / mixture pseudo-binary phase diagram computation. The input of the mixture phase is possible either via the `binary={i $MIX}` command, where `i` is the number of the "solvent" phase, i.e. a neutral solvent compound (with number `i` as given in the sequence of compounds in the compound input section - argument `i`.is expected to be a positive integer number) and the term `$MIX` denotes the composite / mixture phase, or alternatively, by the `nbinary={name`$_i$` $MIX}` command, where `name`$_i$ is the name of the "solvent" phase compound as given in the compound input section and the term `$MIX` denotes the composite / mixture phase as defined by the `xm` or `cm` options given below. Note that any – or both – of the binary phases can be defined as `$MIX` phases.

Options for pseudo-ternary phase diagram computations with composite / mixture phases:

`ternary={i $MIX $MIX}`
or
`nternary={name`$_i$` $MIX $MIX}`
	Optional for `ternary` computations: Toggle composite / mixture pseudo-binary phase diagram computation. The input of the mixture phase is possible either via the `ternary={i $MIX $MIX}` command, where `i` is the number of a "solvent" phase, i.e. a neutral solvent compounds (with number `i` as given in the sequence of compounds in the compound input section - argument `i` is expected to be a positive integer number) and the terms `$MIX` denote the composite / mixture phases, or alternatively, by the `nternary={name`$_i$` $MIX $MIX}` command, where `name`$_i$ is the name of a "solvent" compound as given in the compound input section, and the terms `$MIX` denote the composite / mixture phase as defined by the `xm` or `cm` options given below. Note that any – or all – of the ternary phases can be defined as `$MIX` phases.

Suboptions of the `binary={i $MIX}` or `nbinary={name_i $MIX}` options:

`xm={x1 x2 …}`
or
`cm={c1 c2 …}`
	Required for `binary` or `ternary $MIX` computations: Give finite mixture concentration for the `$MIX` composite phase defined in the binary computation input. The input of the concentrations is possible either in mole fractions (`xm={}`) or mass fractions (`cs={}`) of the compounds of the mixture (as real numbers x_i and c_i). The arguments are expected as real numbers between zero and one in the same sequence of compounds as given in the second input area. If the values do not add up to one, COSMOtherm will normalize them. If less mole fractions / concentrations than compounds are given, the missing ones will be assumed zero.

`MIXphase=REAL`	Optional for `binary` or `ternary $MIX` computations: Define the mixture phase as "real" mixture with respect to activity and vapor pressure of the phase. The argument of the `MIXphase` option is expected to be upcase.

Suboptions of the `SLE` and `binary={i $MIX}` or `nbinary={name`$_i$` $MIX}` options:

`DGfus_mix=value`
or
`DGfus_mix_SI=value`
	Optional: Give the free enthalpy of fusion ΔG_{fus} for mixture / composite. For the `DGfus_mix=value` option, ΔG_{fus} is expected in [kcal/mol], for the `DGfus_mix_SI` option, ΔG_{fus} is expected to be in [kJ/mol]. Argument `value` is expected to be a real number.

`DHfus_mix=value`
or
`DHfus_mix_SI=value`
	Optional: Give the enthalpy of fusion ΔH_{fus} for mixture / composite. For the `DHfus_mix=value` option, ΔH_{fus} is expected in [kcal/mol], for the `DHfus_mix_SI` option, ΔH_{fus} is expected to be in [kJ/mol]. Argument `value` is expected to be a real number.

`DSfus_mix=value`
or
`DSfus_mix_SI=value`
	Optional: Give the entropy of fusion ΔS_{fus} for mixture / composite. For the `DSfus_mix=value` option, ΔS_{fus} is expected in [kcal/mol], for the `DSfus_mix_SI` option, ΔS_{fus} is expected to be in [kJ/mol]. Argument `value` is expected to be a real number.

`Dcpfus_mix=value`
or
`Dcpfus_mix_SI=value`
	Optional: Give the heat capacity of fusion ΔCp_{fus} for a mixture / composite. For the `Dcpfus_mix=value` option, ΔCp_{fus} is expected in [kcal/mol], for the `Dcpfus_mix_SI` option, ΔCp_{fus} is expected to be in [kJ/mol]. Argument `value` is expected to be a real number.

`Tmelt_mix=temp`
or
`Tmelt_mix_C=temp`
or
`Tmelt_mix_K=temp`
	Optional: Give the melting temperature T_{melt} for a mixture / composite. For the `Tmelt_mix=temp` and `Tmelt_mix_C=temp` options, T_{melt} is expected in [°C], for the `Tmelt_mix_K=temp` option, T_{melt} is expected in [K]. Argument `temp` is expected to be a real number.

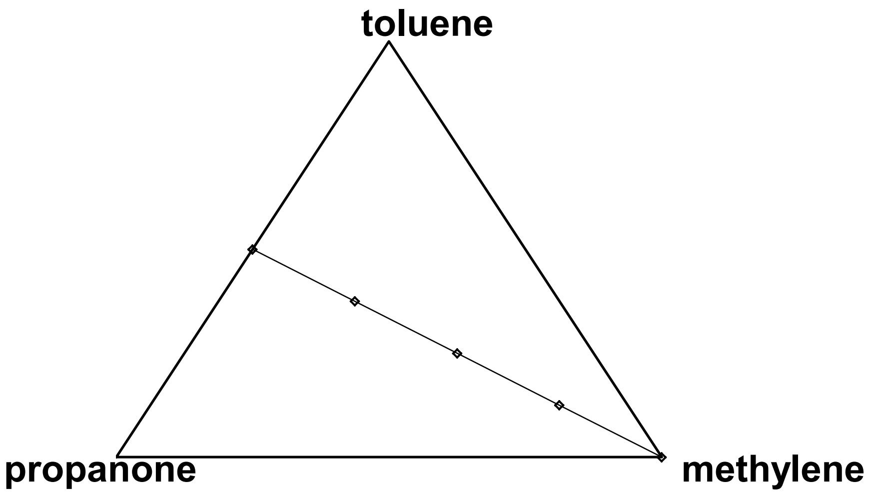
`Dcpfus_estimate`	Optional for the input of a temperature dependent compound free energy of fusion (via input of enthalpy or entropy of fusion ΔH_{fus} or ΔS_{fus}, and melting point T_{melt}): toggle the approximation of the heat capacity of fusion as $\Delta Cp_{fus} = \Delta S_{fus} = \Delta H_{fus}/T_{melt}$. The value of $\Delta G_{fus}(T)$ thus obtained wil be used to compute the SLE of the given Ionic Liquid / salt system. If the `Dcpfus_estimate` keyword is given in binary mixture SLE computation input line, the approximation to ΔCp_{fus} is valid for all compounds including the given Ionic Liquid / salt in the given mixture line.

2.3.7.7 N-dimensional Phase Diagrams: "Multinary" Mixture Computations

The **`multinary`** option is a generalization of the `binary/ternary` concept to *n* dimensions. It allows for the automatic computation of the thermodynamic properties of *n*-dimensional multi-component mixtures similar to the way done in the `binary/ternary` options described in section 2.3.7. The `multinary` option performs an automatic calculation of a slice of the *n*-dimensional phase space. The section of the phase space to be computed is defined in terms of of a custom concentration grid. This requires the input of start- and end-concentration vectors of mole, mass, or surface fraction concentrations plus the number of points to be calculated between these vectors. The multinary calculation moves along these coordinates using equidistant concentration steps. The start- and end vectors are defined as the first and last point in phase space[58]. Example 2.3.7-1 shows the input file for the `multinary` computation of a section of a three-component system. The section computed is also shown graphically in the ternary diagram of this system. The vapor pressures of the pure compounds which are used in the computation of the thermodynamic properties are handled as described above for the `binary/ternary` options, i.e. they can be approximated by COSMO*therm* or given in the molecular section of the COSMO*therm* input file. The maximum number of compounds that can be handled by a multinary computation is 235.

`Example 2.3.7-1:`

```
ctd = BP_TZVP_C30_1701.ctd                                # Global Commands 1
cdir=/COSMOTHERM/CTDATA-FILES                             # Global Commands 2
!! acetone - methylene chloride - toluene MULTINARY !!    # Comment Line
f=propanone.cosmo ef=propanone.energy                     # Compound Input 1
f=ch2cl2.cosmo    ef=ch2cl2.energy                        # Compound Input 2
f=toluene.cosmo   ef=toluene.energy                       # Compound Input 3
tc=0 multinary xstart={0 1 0} xend={0.5 0 0.5} xstep=5    # MULTINARY Computation
```



---

[58] Note, that the automatic computation of thermodynamic properties as done in `binary/ternary` and `multinary` computations requires the additional calculation of the chemical potentials of the pure substance of all of the compounds involved. These additional calculations automatically are done by COSMO*therm*. They will not be printed into the tabulated results file, only into the long COSMO*therm* output file.

The additional input required for the `multinary` option (in the same line of the input file) is:

xstart={x_1 x_2 ..} or cstart={c_1 c_2 ..} or qstart={q_1 q_2 ..}	Required for `multinary` computations: Custom concentration grid start vector mole fraction (`xstart`), mass fraction (`cstart`), or surface fraction (`qstart`) concentrations of the compounds of a multinary mixture (as real numbers x_i, c_i, or q_i). The arguments are expected to be real numbers between zero and one in the same sequence of compounds as given in the second input area. If the values do not add up to one, COSMO*therm* will normalize them. If less mole fractions / concentrations than compounds are given, the missing ones will be assumed zero. If a negative number is given, the concentration for this compound will be inserted automatically using the normalization of the sum of mole fractions. Only one negative number is allowed per mixture input line.
xend={x_1 x_2 ..} or cend={c_1 c_2 ..} or qend={q_1 q_2 ..}	Required for `multinary` computations: Custom concentration grid end vector mole fraction (`xend`), mass fraction (`cend`), or surface fraction (`qend`) concentrations of the compounds of a multinary mixture (as real numbers x_i, c_i, or q_i). The attributes of the `xend`, `cend`, and `qend` commands are the same as described for the `xstart`, `cstart`, and `qstart` keywords, above. Please note that is possible to leave out this definition of a concentration grid end vector in the input. In such a case only one grid point, as defined by the `xstart`, `cstart`, or `qstart` keywords, above, will be computed,.
xstep=n_{points} or cstep=n_{points} or qstep=n_{points}	Optional for `multinary` computations: Number of points to be calculated in custom concentration grid of a multinary mixture computation. Argument n_{points} is expected to be an integer number between 2 and 235. Default is n_{points}=10. If `xstep` is used, evenly spaced mole fraction concentration grid points will be set between the given start- and end-concentration vectors. If `cstep` is used, evenly spaced mass fraction concentration grid points will be set between the given start- and end-concentration vectors. If `qstep` is used, evenly spaced surface fraction concentration grid points will be set between the given start- and end-concentration vectors. The `xstep`, `cstep`, or `qstep` options are independent concentration definition of the start- and end-concentration vectors given. Any possible combination of `xstep`, `cstep`, or `qstep` keywords with `xstart`, `cstart`, or `qstart` and `xend`, `cend`, or `qend` is allowed.

2.3.7.8 Phase Diagrams: Isobaric

The `binary/ternary/multinary` options by default compute phase diagrams at a fixed given temperature. It is also possible to compute phase diagrams at a given fixed pressure with variable temperatures via the `isobar` command. If this sub option of the `binary/ternary/multinary` options is used, for each given concentration, COSMO*therm* will compute the mixture properties at the temperature given in the input file plus at additional two temperature values above and below the given initial temperature. Thus, a starting guess for the temperature is necessary for any `isobar` computation. By default room temperature (298.15 K) is used as starting guess. Alternatively it is possible to override this default and read the first guess for the exact temperature from the input. This is done with the usual temperature input commands (`tk=temp` [K], `tc=temp` [°C] or `tf=temp` [°F]). The vapor pressures computed at the three temperature values are then used to interpolate the temperature value at the given pressure. In the fourth step, all the thermodynamic properties of the mixture are calculated at this „optimized“ interpolated temperature. Thus, compared to the isothermal case, the automatic computation of isobar phase diagram requires three additional COSMO*therm* calculations per mixture. If the temperature thus optimized is outside the range of the three computed temperatures, i.e. if the scheme was used to extrapolate rather than to interpolate the temperature, extrapolation errors might be introduced into the resulting temperature. However, such errors can be minimized using the `iterative[=thresh]` sub option of the isobar option. If the `iterative` option is given, the interpolated or extrapolated temperature is refined iteratively: The optimized temperature is used as new starting guess for the temperature. Again COSMO*therm* will compute the mixture properties at that temperature plus at additional two temperature values above and below that temperature and interpolate a new guess for the optimized temperature from the vapor pressures computed at these three temperatures. This procedure is repeated until the change in the guess of the optimized temperature is below a certain threshold `thresh` (default `thresh=0.2` [K]).

The syntax of the `isobar` option is:

`isobar=pressure`
or
`isobar_Pa=pressure`
or
`isobar_kPa=pressure`
	Optional for `binary`, `ternary` or `multinary` computations: Compute phase diagram at fixed pressure given in [mbar] (and in [Pa] for the `isobar_Pa` command or in [kPa] for the `isobar_kPa` command). The argument `pressure` is expected to be a positive real number.

Suboptions of the `isobar` option are:

`iterative[=thresh]`	Optional for `binary`, `ternary` or `multinary` computations with the `isobar` option: Iteratively refine the interpolated temperature to achieve a higher quality of the isobar phase diagram. The optional argument `thresh` is the accuracy of the temperature to be reached by the iterative process. The iterative procedure is repeated until the change in energy is below the threshold value `thresh`. If no argument is given a default accuracy of thresh=0.2 [K] is used. The optional argument `thresh` is expected to be a positive real number.

2.3.7.9 Treatment of Compressible Liquids and Nonideal Gas (Equation of State Methods)

COSMO*therm*'s underlying COSMO-RS theory (see section 1.2) is based on the assumption of incompressible liquid and ideal gas phase. This has the consequence that the vapor-liquid and liquid-liquid equilibrium calculation methods offered by COSMO*therm* (`binary`, `ternary`, and `multinary` VLE and LLE) are restricted to systems that are well below the critical point. Systems that at the given temperature, pressure, and mixture conditions are near or beyond the critical point are described only approximately, as under these circumstances a liquid phase has to be considered compressible and a vapor phase nonideal. To overcome this obstacle of critical or near-critical systems, COSMO*therm* offers the combination of the regular COSMO-RS based predictions with **Equation of State** (**EOS**) methodologies[59].

EOS are empiricial or semiempirical model equations, which relate the state variables temperature **T**, pressure **p**, and volume **V** (or equivalently molar density ρ) of pure compounds. COSMO*therm* currently offers the Soave-Redlich-Kwong (**SRK**), and the Peng-Robinson (**PR**) EOS model, both of which are cubic EOS that are widely used in chemical engineering applications.

The basic formulation of the SRK[60] EOS reads

$$p = \frac{RT}{V-b} - \frac{a(T)}{V(V+b)} \tag{2.3.7-19}$$

The SRK EOS holds two adjustable parameters *a*(T) and *b*, which can be related[60] to pure compound critical properties namely critical temperature T_C, critical pressure p_C, and critical volume V_C or critical density ρ_C:

$$a(T) = c_a \frac{(RT_C)^2}{p_C} \alpha(T) \text{ with } \alpha(T) = \left(1 + \kappa\left(1 - \sqrt{(T/T_C)}\right)\right)^2 \text{ and } \kappa = c_0 + c_1\omega + c_2\omega^2$$
$$b = c_b \frac{RT_C}{p_C} \tag{2.3.7-20}$$

Where parameter values are c_a = 0.42748, c_b = 0.08644, c_0 = 0.48, c_1 = 1.57, and c_2 = -0.176 as given by Soave[60]. In addition to the critical properties, equation 2.3.7-20 requires the knowledge of the compound acentric factor ω, which according to Pitzer *et al.*[61] is defined as:

$$\omega = -1 - \log_{10}\left[p_{vap}(T = 0.7T_C)/p_C\right] \tag{2.3.7-21}$$

The definition of the acentric factor does not involve any additional correlations. It is computed from the critical pressure p_C and a pure compound vapor pressure calculation at a temperature of $0.7{*}T_C$, which typically is near the compounds boiling point temperature.

The PR[62] EOS in it's original form is defined as

$$p = \frac{RT}{V-b} - \frac{a(T)}{V(V+b)+b(V-b)} \tag{2.3.7-22}$$

---

[59] Hahn, M.; Eckert, F.; Klamt A., *in preparation* (2016).

[60] Redlich, O.; Kwong, J.N.S., *Chem. Rev.* **44**, 233 (1949). Soave, G., *Chem. Eng. Sci.* **27**, 1197 (1971).

[61] Pitzer, K.S.; Lippmann, D.Z.; Curl, R.F.; Huggins, C.M.; Petersen, D.E., *J. Am. Chem. Soc.* **77**, 3433 (1955).

[62] Peng, B.E.; Robinson, D.B., *Ind. Eng. Chem. Fundamentals* **15**, 59 (1976).

The parameters *a*(T) and *b* of the PR EOS can also by computed by eq. 2.3.7-20. The parameter values for the PR EOS are c_a = 0.45724, c_b = 0.077796, c_0 = 0.375, c_1 = 1.454, and c_2 = -0.269[62]. Note, that the COSMO*therm* implementation of the SRK and PR EOS uses a modification of *a*(T) proposed by Boston and Mathias[63], as this is commonly believed to have a more realistic behavior at near critical and supercritical conditions[59,64]. The critical properties T_C, p_C, and V_C or ρ_C and the acentric factor ω required to evaluate the EOS either can be given in the compound input or the compound vap-file (see section 2.2.3 of this manual), or they can be estimated by COSMO*therm* (via the "pure compound critical properties" `PCCP` option, see section 2.3.13). If no experimental data is provided in the compound input or vap-file, COSMO*therm* will automatically compute the estimate via the `PCCP` option.

To be able to describe mixtures the EOS have to be combined with a specific **mixing rule** (**MR**). When EOS are extended to mixtures, the basic form of the EOS does not change. The EOS parameters however, will now be functions of the composition (*a*(T,x) and *b*(x) in case of SRK and PR EOS). COSMO*therm* offers the van der Waals "one fluid" (**VDW**) mixing rule, the Wong-Sandler (**WS**) mixing rule, and the modified Huron-Vidal (**MHV1**) mixing rule. The mixing rules can be combined with each of the EOS methods offered.

A straightforward and simple mixing rule is based on the assumption that for mixtures of roughly equal-sized spherical particles that mix ideally (i.e. no strong interactions present), the liquid mixture behaves like a random fluid[65]. Real mixtures in this prespective are considered to be hypothetical single-component fluids. In terms of this "one fluid" assumption in an EOS framework, a mixture can be described by composition dependent parameters that describe size and energy differences of the particles involved. The resulting mixing rule is called the "one fluid" van der Waals (**VDW**) MR[66]. It is available in COSMO*therm* in the form of eqs. 2.3.7-23:

$$a_m(T,x) = \sum_i \sum_j x_i x_j \sqrt{a_i(T) a_j(T)}\left(1 - k_{ij}\right) \quad \text{and} \quad b_m(x) = \sum_i \sum_j x_i x_j \frac{1}{2}\left(b_i + b_j\right)\left(1 - l_{ij}\right) \qquad (2.3.7\text{-}23)$$

Here subscript *m* indicates the mixture. I.e. a_m and b_m are adjustable EOS parameters *a* and *b* that additionally depend on the mixture concentration. The VDW MR introduces one additional binary interaction parameter k_{ij} which describes intermolecular interactions. The binary parameter k_{ij} is symmetric ($k_{ji} = k_{ij}$) and vanishes in the case of self-interactions ($k_{ii} = k_{jj} = 0$). The covolume parameter l_{ij}, which describes size differences, is set to zero[66], simplifying the mixing rule for parameter *b* to a simple linear interpolation expression. As the VDW MR only accounts for interaction energy differences in terms of a single binary parameter k_{ij}, which has to describe all possible types of interaction in the mixture, it should only be applied to systems without associating compounds and strong directional interactions such as hydrogen bonding.

---

[63] Boston, P.M.; Mathias P.M., *Proceedings of the 2nd International Conference onPhase Equilibria and Fluid Properties in the Chemical Process Industries*, West Berlin, March, 823 (1980).

[64] Neau, E.; *et al.*, *Fluid Phase Equilibria* **276**, 87 (2009).

[65] Brown, W.B., *Phil. Trans. R. Soc.* **250**, 175 (1957). Rowlinson, J.S.; Watson, I.D., *Fluid Phase Equilibria* **24**, 1565 (1969).

[66] Henderson D.; Leonard, P.J., *Proc. Nat. Acad. Sci.* **67**, 181 (1970).

The mixing rule of Wong and Sandler[67] (**WS** MR) is implemented with a modification proposed by Orbey and Sandler[68] as:

$$a_m(T,x) = b_m\left(\sum_i x_i \frac{a_i(T)}{b_i} + \frac{G^{Ex}}{C^*}\right) \qquad \text{and} \qquad b_m(x) = \frac{\sum_i \sum_j x_i x_j \left(b - \frac{a}{RT}\right)_{ij}}{1 + \frac{G^{Ex}}{RT\,C^*} - \sum_j x_j \frac{a_j(T)}{b_j RT}} \qquad (2.3.7\text{-}24)$$

In this framework the second order cross virial term $(b - a/RT)_{ij}$ is defined as:

$$\left(b - \frac{a}{RT}\right)_{ij} = \frac{1}{2}\left[\left(b_i - \frac{a_i}{RT}\right) + \left(b_j - \frac{a_j}{RT}\right)\right](1 - k_{ij}) := \frac{1}{2}(b_i + b_j)(1 - l_{ij}) - \frac{1}{RT}\sqrt{a_i(T)a_j(T)}\,(1 - k_{ij}) \qquad (2.3.7\text{-}25)$$

As noted above for the VDW MR, the covolume parameter l_{ij} is set to zero. Thus apart from the binary interaction parameter k_{ij}, the WS MR requires the input of G^{Ex}, the excess Gibbs free energy of the system at the given temperature, pressure and mixture conditions, and a dimensionless constant C^*, which is specific to the EOS that is used with the WS mixing rule. The EOS dependent constant C^* can be calculated analytically or numerically from the limiting reduced liquid phase volume ($\lim_{p\to\infty} u = 1$, where the reduced liquid volume is defined as $u = V/b$). In the COSMO*therm* implementation of the WS mixing rule a value of $C^* = \ln(1/2) = -0.6931$ is used for the SRK EOS[70,59], and $C^* = \ln(\sqrt{2} - 1)/\sqrt{2} = -0.6232$ for the PR EOS[69,59].

The mixing rule of Huron and Vidal[70] is implemented with the "first order" modification proposed by Michelsen[71], which is commonly denoted as "**MHV1**" MR.

$$\alpha_m(T,x) = \sum_i x_i \alpha_i + \frac{1}{C^*}\left[\frac{G^{Ex}}{RT} + \sum_i x_i \ln\left(\frac{b_m}{b_i}\right)\right] \qquad \text{and} \qquad b_m(x) = \sum_i x_i b_i \qquad (2.3.7\text{-}25)$$

In eq. 2.3.7-25 $a_m(T,x)$ is replaced by the dimensionless parameter $\alpha = a/(bRT)$. The MHV1 MR also uses G^{Ex}, the excess Gibbs free energy of the system, and an EOS dependent constant C^* that can be calculated analytically from thermodynamical consideration[71] or fitted to experimental data[72]. In the COSMO*therm* implementation of the MHV1 mixing rule a value of $C^* = -0.64663$ is used for the SRK EOS[72,59], and $C^* = -0.53087$ for the PR EOS[71,59].

As noted above, the WS and MHV1 mixing rules additionally require the Gibbs free energy of mixing G^{Ex} at the given temperature and mixture conditions. G^{Ex} is computed on the fly by COSMO*therm* via

$$G^{Ex}(T,x) = RT \sum_i x_i \ln(\gamma_i) \qquad (2.3.7\text{-}26)$$

Note, that in case of the MHV1 MR only the residual part of the compound activity coefficients γ_i^{res} is used, as this MR was derived with the assumption that the excess entropy of mixing S^{Ex} is zero. Thus the combinatorial contribution to γ_i and G^{Ex} - which is pure entropy – should not be used with the MHV1 MR[73].

---

[67] Wong, D. S. H.; Sandler, S. I., *AIChE J.* **38**, 671 (1992). Wong, D. S. H.; Orbey, H.; Sandler, S. I., *Ind Eng. Chem. Res.* **31**, 2033 (1992).

[68] Orbey, H.; Sandler, S. I., *Fluid Phase Equilibria* **111**, 53 (1995).

[69] Stryek, R.; Vera, J.H., *Can. J. Chem. Eng.* **64**, 323 (1986).

[70] Huron, M.-J.; Vidal, J., *Fluid Phase Equilibria* **3**, 255 (1979).

[71] Michelsen, M.L., *Fluid Phase Equilibria* **60**, 42 (1990). Michelsen, M.L., *Fluid Phase Equilibria* **60**, 213 (1990).

[72] Fischer, K.; Gmehling, J., *Fluid Phase Equilibria* **121**, 185 (1996).

[73] Kontogeorgis, G.M.; Coutsikos, P., *Ind. Eng. Chem. Res.* **51**, 4119 (2012).

All mixing rules considered here include - in addition to the Gibbs free energy of mixing G^{Ex} for WS and MHV1 MR – the input or adjustment of the binary interaction parameter k_{ij}. There are several possible choices for k_{ij}, which correspond to different choices with respect to the predictivity and extrapolative character of the given EOS and MR combination. The simplest choice of neglecting the binary interaction parameter (k_{ij} = 0) corresponds to the case of maximum "predictivity". The EOS/MR is not adjusted to any data and thus it typically will not reproduce a given phase diagram, even if the EOS/MR is evaluated at the exact same temperature, pressure and mixture conditions as the phase diagram. Thus in the case of k_{ij} = 0, the computed phase diagram will mostly reflect the character of the EOS/MR combination used. Other common choices of k_{ij} try to adjust the parameter value in a way that a given phase diagram is represented in an optimal way. Possible ways of adapting k_{ij} include: the variational adjustment of k_{ij} to the given phase diagram data to best represent the phase diagram data in a least-squares sense – this is the opposite extreme in that it forces the EOS/MR to optimally interpolate a given phase diagram over the whole range of concentration points; the variational adjustment of k_{ij} to one phase diagram data point at equimolar concentrations at a temperature of $T = 0.85\ T_C$ – this is somewhat a compromise between the inter- and extrapolative quality of the EOS/MR; compute k_{ij} from G^{Ex} data and the compound a_i parameters of the EOS - this is also a compromise between the inter- and extrapolative quality of the EOS/MR, the actual fit procedure is specific to the mixing rule used[59,73].

The EOS and MR methods described above are available for **`binary`** VLE, LLE, and SLE calculations. The usage of an EOS is requested by the keyword **`EOS[=eos]`**, where the optional argument `eos` is the equation of state method. Currently two EOS methods are available: `SRK` for the Soave-Redlich-Kwong EOS, and `PR` for the Peng-Robinson EOS. If no argument is given, the PR EOS will be used by default. If no other input option is given, the Wong-Sandler (WS) mixing rule will be used in EOS mixture calculations. The mixing rule can be changed by keyword `MR=mr`, where argument `mr` is a shortcut for the mixing rule to be used. Currently three mixing rules are available: `VDW` for the "one-fluid" van der Waals (VDW) MR, and `MHV1` for the modified Huron-Vidal (MHV1) MR, and `WS` for the Wong-Sandler (WS) MR. The binary interaction parameter of the mixing rule k_{ij} by default is fitted to G^{Ex} data and a_i parameters of the EOS. This can be changed with the keyword `KIJ=method`, where argument `method` is denotes one of the fitting methods explained above: argument `0` sets k_{ij} = 0 and requires the EOS/MR to work in "predictive" mode; with argument `fit` k_{ij} will be fitted to the phase diagram data as computed by COSMO*therm*; with argument `single` k_{ij} will be fitted to one data point computed by COSMO*therm* at equimolar concentrations at a temperature of $T = 0.85\ T_C$; with argument `calc` (= default) k_{ij} will be fitted to G^{Ex} data and a_i parameters of the EOS. Alternatively, the value of k_{ij} can be read from the input via keyword `KIJ={i j kij}`, where argument `kij` is the actual real number value of the interaction parameter to be used. The integer numbers `i` and `j` are the compound/phase indices. Please note that the convergence of solving the EOS/MR may be poor or even fail if the options `KIJ=fit` or `KIJ=single` are used at temperatures near or beyond the critical poin of the given system.

If no other input is given, the EOS calculation will be done at the same temperature and pressure conditions as the underlying "regular" VLE, LLE, or SLE calculation. In this case the EOS/MR is operated in an interpolative way. It is also possible to extrapolate the EOS/MR to different temperature or pressure conditions. To do this, the additional extrapolation temperature input `EOS_temp=T` (for *T* in [°C]), or it's unit variations `EOS_temp_C=T` (for *T* in [°C]), or `EOS_temp_K=T` (for *T* in [K]) can be used for *isothermal* phase diagram computations. An according functionality for *isobar* EOS/MR computations is not yet implemented . Note that if the EOS/MR model is used in an extrapolative way, it is recommended to use a temperature of T = 0.85 T_C (with T_C being the lowest pure compound critical temperature of the compounds involved) to compute the underlying "regular" VLE, LLE, or SLE calculation, because lower temperatures may impede the evaluation of the EOS/MR and possibly lead to divergence of the model equations. At higher temperatures, in particular temperatures above the critical points of the compounds involved, the COSMO*therm* predictions of the γ_i and G^{Ex} values used by the MR can not be considered reliable any more. Hence a temperature of T = 0.85 T_C is a reasonable[73] and widely used compromise between the two limiting extremes.

If activated in a `binary` VLE, LLE, or SLE calculation the EOS option will create an additional binary phase diagram table, which is similar to the regular phase diagram table, but involves additional properties. The table columns of the properties `x`, and `y` (mole fraction of the compound in the liquid and in the gas phase), as well as `H^E`, `G^E`, and `ptot` (excess enthalpy, excess Gibbs free energy, and total vapor pressure of the system) are fully equivalent to the table columns of the regular `binary` VLE LLE, or SLE calculations with the only differences that the data in these columns is evaluated from the EOS/MR model at the given EOS extrapolation temperature or pressure. The `mu[i]+RTln(x[i])` columns are not printed for the EOS model. Instead of the `ln(gamma[i])` columns in the regular binary, the EOS model prints out `mu_ex[i]`, which is the fugacity-derived chemical-potential of compound *i* in the liquid phase $\mu_i^{Ex} = f_i^L/\left(x_i f_i^{L,0}\right)$ using the mixture and pure compound liquid fugacity f_i^L of compound *i* as computed from the EOS. The property μ_i^{Ex}/RT is the compressible-liquid and nonideal-gas equivalent of the activity coefficient as computed by the EOS. Additional columns of the EOS table include `PhiL[i]`, and `PhiG[i]`, which are the fugacity coefficients of compound *i* in the liquid and gas phase, $\phi_i^L = f_i^L/(x_i p_{tot})$ and $\phi_i^G = f_i^G/(y_i p_{tot})$, respectively. Note that this definition of liquid and gaseous fugacity coefficients corresponds to the ϕ-ϕ (phi-phi) approach to phase diagrams, where the EOS is used to describe both the liquid and the gas phase.

The usage of EOS models in the framework of VLE, LLE, or SLE calculations is activated by:

`EOS[=eos]`	Required for `EOS` computations: Trigger the computation of an EOS model. The optional argument `eos` chooses the EOS model to be used. Possible EOS models: `SRK` requests the Soave-Redlich-Kwong (SRK) EOS; `PR` requests the Peng-Robinson (PR) EOS. If no argument is given, the PR EOS will be used.

Suboptions of the **`EOS`** option are:

`MR[=mr]`	Optional for `EOS` computations: Choose a mixing rule for the EOS model computation. The optional argument `mr` chooses a shortcut for the mixing rule to be used. Possible mixing rules: `VDW` ("one-fluid" van der Waals (VDW) mixing rule); `MHV1` (modified Huron-Vidal (MHV1) mixing rule, default); `WS` (Wong-Sandler (WS) mixing rule).

`KIJ[=method]`
or
`KIJ={i j` k_{ij}`}`
	Optional for `EOS` computations: Choose the calculation method of mixing rule binary interaction parameter k_{ij} or directly give k_{ij} value as input. Argument `0` sets $k_{ij} = 0$. With argument `fit` k_{ij} will be fitted to the phase diagram data as computed by COSMO*therm*. With argument `single` k_{ij} will be fitted to one data point computed by COSMO*therm* at equimolar concentrations at a temperature of T = 0.85 T_C. With argument `calc` k_{ij} will be fitted to G^{Ex} data and a_i parameters of the EOS. Alternatively, the value of k_{ij} can be read directly from the input in waived brackets, where integer arguments `i` and `j` are the compound/phase indices, and `k`$_{ij}$ is the actual real number value of the interaction parameter to be used.

`EOS_temp=T`
or
`EOS_temp_C=T`
or
`EOS_temp_K=T`
	Optional for *isothermal* `EOS` computations: Input of the extrapolation temperature, i.e. the temperature where the EOS model will be applied and evaluated. Option `EOS_TEMP=T` expects the argument `T` to be a temperature in [°C], `EOS_TEMP_C=T` expects the argument `T` to be a temperature in [°C], and `EOS_TEMP_K=T` expects the argument `T` to be a temperature in [K].

2.3.8 Similarity

COSMO*therm* allows the automatic calculation of the molecular σ-profile similarity $S_{i,j}$ (SIM) of two compounds *i* and *j*, using the `similarity={i j}` or `nsimilarity={name_i name_j}` options in the mixture input section of the COSMO*therm* input file (see section 5.1 on σ-profiles). For the `similarity={i j}` option $S_{i,j}$ is computed for the two compounds with the compound numbers `i` and `j` in the order of compounds given in the compound input section. For the `nsimilarity={name_i name_j}` option, $S_{i,j}$ is computed for the two compounds with the compound names `name_i` and `name_j` as given in the compound input section of the COSMO*therm* input file. The computed similarity factor $S_{i,j}$ is printed to the mixture output section of the COSMO*therm* output file. $S_{i,j}$ is printed below the compound output block of the first compound given in the similarity command (i.e. compound `i` or `name_i`). If several conformers are present for either compund *i* or *j*, the similarity factor will be computed for all possible combinations of the given conformers and the overall compound similarity factor is averaged from the computed conformer similarity factors. The COSMO*therm* σ-profile similarity factor $S_{i,j}$ is defined as the overlap integral of the σ-profiles $p_i(\sigma)$ and $p_j(\sigma)$ of the two compounds *i* and *j* :

$$S_{i,j} = \frac{\int_{-\infty}^{+\infty} p_i(\sigma) p_j(\sigma) d\sigma}{A_i A_j} \tag{2.3.10-1}$$

Thus $S_{i,j}$ will be small if the overlap between the compounds σ-profiles is small. Please see section 5.1 for further information on the definition and use of σ-profiles. In addition, the similarity factor given by eq. (2.3.20) is corrected by a factor $S^{HB}_{i,j}$ taking into account the difference in the apparent hydrogen bonding donor and acceptor capacities of the two compounds and by a factor $S^{A}_{i,j}$ taking into account size differences between the two compounds *i* and *j*. Note that $S^{SIM}_{i,j}$, the overall σ-profile similarity is not normalized, i.e. a similarity compution of a molecule with itself usually will result in a similarity value of $S^{SIM}_{i,j} \neq 1$.

Alternatively COSMO*therm* allows the calculation of a molecular σ-profile similarity by the "Sigma-Match Similarity" (SMS) algorithm[74]. The similarity factor $S^{SMS}_{i,j}$ of two compounds *i* and *j*, can be computed using the `sms={i j}` or `nsms={name_i name_j}` options in the mixture input section of the COSMO*therm* input file (see section 5.1 on σ-profiles). $S^{SMS}_{i,j}$ is normalized. A similarity compution of a molecule with itself will result in a similarity value of $S^{SMS}_{i,j} = 1$.

In addition COSMO*therm* allows the calculation of a molecular σ-potential similarity $S^{P}_{i,j}$ (SPS) of two compounds *i* and *j*, using the `simpot={i j}` or `nsimpot={name_i name_j}` options in the mixture input section of the COSMO*therm* input file (see section 4.2 on σ-potentials). For the `simpot={i j}` option $S^{P}_{i,j}$ is computed for the two compounds with the compound numbers `i` and `j` in the order of compounds given in the compound input section. For the `nsimpot={name_i name_j}` option, $S^{P}_{i,j}$ is computed for the two compounds with the compound names `name_i` and `name_j` as given in the compound input section of the COSMO*therm* input file. In case of the σ-potential similarity it is not only possible to compute the similarity of two given compounds, but also the similarity of two compound mixture phases. This is possible if in the input file the mixture concentrations are given for the two phases between which the sigma-

---

[74] Thormann, M.; Klamt, A.; Hornig, M.; Almstetter, M. *J. Chem. Inf. Model.* **46**, (2006) 1040 -1053.

potential similarity coefficient shall be computed. The input of the concentrations of the two phases is possible either in mole fractions (`xs1={`x_1 x_2 `…}`, `xs2={`x_1 x_2 `…}`) or mass fractions (`cs1={ `c_1 c_2 `…}`, `cs2={ `c_1 c_2 `…}`) of the compounds of the mixture (where x_i are mole fractions and c_i are mass fraction concentrations). The computed similarity factor $S^P_{i,j}$ is printed to the mixture output section of the COSMO*therm* output file. $S^P_{i,j}$ is printed below the compound output block of the first compound given in the similarity command (i.e. compound `i` or `name`$_i$). At a given temperature, the `[n]simpot` option will toggle two COSMO*therm* computations of the σ-potentials $\mu_i(\sigma)$ and $\mu_j(\sigma)$ for the pure compounds *i* and *j*, respectively. The COSMO*therm* σ-potential similarity factor $S^P_{i,j}$ is defined as the sum of the differences between the two pure compound σ-potentials $\mu_i(\sigma)$ and $\mu_j(\sigma)$:

$$S^P_{i,j} = \exp\left(-\sum_{m=-0.02}^{m=+0.02} \left|\mu_i(\sigma_m)-\mu_j(\sigma_m)\right|\right) \tag{2.3.10-2}$$

Thus $S^P_{i,j}$ will be small if the overlap between the compounds σ-potentials is small. Please see section 5.2 for further information on the definition and use of σ-potentials. $S^P_{i,j}$ is normalized. A similarity compution of a compound with itself will result in a similarity value of $S^P_{i,j} = 1$.

As an alternative to the simple and molecular non-specific cutoff function of eq. 2.3.10-1, COSMO*therm* allows the calculation of a *solute-specific* σ-potential similarity. $S^P_{i,j}(p_k)$ is the σ-potential similarity for compounds *i* and *j* weighted by the σ-profile $p_k(\sigma)$ of compound *k*. As shown in eq. 2.3.10-2 the σ-profile of a third compound *k* is used as a weighting function in surface charge density σ during the calculation of the σ-potential difference of compounds *i* and *j*.

$$S^P_{i,j}(p_k) = \exp\left(-\sum_{m=-0.03}^{m=+0.03} \boldsymbol{p_k}(\sigma_m)\left|\mu_i(\sigma_m)-\mu_j(\sigma_m)\right|\right) \tag{2.3.10-3}$$

In COSMO*therm* this solute specific weighting of the σ-potential similarity is viable with the `simpot={i j k}` or `nsimpot={name`$_i$ `name`$_j$ `name`$_k$`}` commands in the mixture input section of the COSMO*therm* input file (see section 4.2 on σ-potentials). For the `simpot={i j k}` option $S^P_{i,j}(p_k)$ is computed for the two compounds with the compound numbers `i` and `j` and weighted by the σ-profile of compound `k`. For the `nsimpot={name`$_i$ `name`$_j$ `name`$_k$`}` option, $S^P_{i,j}(p_k)$ is computed for the two compounds with the compound names `name`$_i$ and `name`$_j$ and weighted by the σ-profile of the compound with the name `name`$_k$. The computed similarity factor $S^P_{i,j}$ is printed to the mixture output section of the COSMO*therm* output file. $S^P_{i,j}(p_k)$ is printed below the compound output block of the first compound given in the similarity command (i.e. compound `i` or `name`$_i$). At a given temperature, the solute-specific `[n]simpot` option will toggle three COSMO*therm* computations for pure compounds *i*, *j* and *k*, respectively.

2.3.9 Solubility of Gases

The **`solgas=p`** option allows for the automatic computation of the solubility of a gas with partial pressure **`p`** in a given solvent. For a given pure solvent (`solvent=i` or `nsolvent=name`$_i$ options, see below) or solvent mixture (`xs={…}` and `cs={…}` options, see below) the `solgas` option will compute the gas solubility of all compounds j in the given solvent or solvent mixture using an iterative procedure: For each compound j the mole fraction x_j is varied until the partial pressure of the compound, which is calculated from eq. 2.3.11-1, is equal to the given reference pressure **`p`**.

$$p_j = p_j^0 x_j \gamma_j \tag{2.3.11-1}$$

The p_j^0 are the pure compound vapor pressures for compounds j. x_j are the mole fractions of the compounds in the liquid, and γ_j are the activity coefficients of the compounds as predicted by COSMO*therm*. Ideal behaviour of the gas phase is assumed. In each iteration step COSMO*therm* will vary x_j, compute γ_j in the given solvent or solvent mixture with a finite portion of compound j according to x_j and from that calculate the partial pressure of the compound p_j according to eq. 2.3.11-1. This process is repeated until the difference of the computed partial vapor pressure p_j and the given reference pressure **`p`** is less than a given threshold. By default this threshold is 0.1 mbar. It can be changed with the `pthresh=thresh` input option (see below).

The computation of a gas solubility requires the knowledge of the vapor pressure of the pure compound p_j^0 at a given temperature. For each compound, there are several possibilities to calculate or approximate this property. In order of increasing accuracy you might: (a) Use the COSMO*therm* approximation of the vapor pressure using the approximated gas phase energy of the compound. This is the default and requires no additional input. (b) Use the COSMO*therm* approximation of the vapor pressure, using the exact gas phase energy of the compound, given via the `e=energy` or `ef=filename` command in the compound input and options section of the input file or the `efile` command in the global options section of the input file. (c) Use the Wagner, DIPPR, or Antoine equation $\ln(p_j^0) = A - B / (T + C)$ to compute the vapor pressure at the given temperature (see section 2.2.1). The Antoine equations' coefficients either can be given directly in the compound input and options section of the input file via the `vpant` command or they can be calculated from three temperature/vapor pressure pairs given via the `vpexp` command. (d) Give the exact value of the vapor pressure for this temperature via the `vpinp` command in the compound input and options section of the input file (see section 2.2.1).

If the `solgas` keyword is given, the reference pressure **`p`** is expected to be a positive real number pressure in [mbar]. If the `solgas_Pa` keyword is given, the reference pressure **`p`** is expected to be a positive real number pressure in [Pa]. If the `solgas_kPa` keyword is given, the reference pressure **`p`** is expected to be a positive real number pressure in [kPa]. If the `solgas_bar` keyword is given, the reference pressure **`p`** is expected to be a positive real number pressure in [bar]. If the `solgas_psia` keyword is given, the reference pressure **`p`** is expected to be a positive real number pressure in [psia]. If the `solgas` keyword is given and the solvent is set correctly (see below), for a given temperature, COSMO*therm* automatically calculates the mole fraction x_j of all compounds in the given solvent or solvent mixture at which the partial pressure according to eq. 2.3.11-1 is equal to the reference pressure. The decadic logarithm of the optimized mole fractions $\log_{10}(x_j)$ is written to the output as well as to the table file. In addition, the natural logarithms of the activity coefficients $\ln(\gamma_j)$ at the optimized solute concentrations x_j will be printed to the output and table files for all compounds j. Please note that the iterative gas solubility computation is ambiguous and physically ill-defined if the gaseous solute is part of the solvent mixture (e.g. if you want to compute the gas solubility of CO_2 in heptane solvent mix with 0.1 mole fractions of CO_2). In such a case COSMO*therm* will assume that the solute concentration in the solvent mixture is zero and compute the gas solubility in the resulting solvent mixture (in the given example COSMO*therm* will compute the gas solubility of the CO_2 in the pure heptane).

Suboptions of the **`solgas`** option are:

`solvent=i` or `nsolvent=name`$_i$	Required for `solgas` computations: Use pure compound *i* as solvent for gas solubility calculation. If the `solvent` or `nsolvent` keyword is used, the gas solubility computation will be done in pure solvent *i*, for all compounds that are given in the compound input section. I.e. the activity coefficient calculations for eq. (2.3.23) will be done in solvent *i*. If the `solvent=i` keyword is given, the ith compound as given in order of the compound input section will be used as solvent. Argument i is expected as integer number between one and the number of compounds given in the compound input section. If the `nsolvent=name`$_i$ keyword is given, the compound with the name `name`$_i$ as given in the compound input section will be used as solvent. Argument name$_i$ is expected as character string. Note, an alternative definition of the solvent in the `solgas` option is possible with the `xs` or `cs` keywords.

`xs={`x_1 x_2 `…}`
or
`cs={`c_1 c_2 `…}`
	Required for `solgas` computations (alternative to `solvent` option): Give finite solvent mixture concentration at which the gas solubility shall be computed. The input of the concentrations is possible either in mole fractions (`xs={}`) or mass fractions (`cs={}`) of the compounds of the mixture (as real numbers x_i and c_i). The arguments are expected as real numbers between zero and one in the same sequence of compounds as given in the second input area. If the values do not add up to one, COSMO*therm* will normalize them. If less mole fractions / concentrations than compounds are given, the missing ones will be assumed zero. If a negative number is given, the concentration for this compound will be inserted automatically using the normalization of the sum of mole fractions. Only one negative number is allowed per mixture input line. Note, an alternative definition of the solvent in the `solgas` option is possible with the `solvent` or `nsolvent` keywords.

`pthresh=thresh`	Optional for `solgas` computations: Give the pressure accuracy threshold for the iterative gas solubility calculation. The gas solubility value will be optimized iteratively until the difference of the predicted partial vapor pressure p_j and the given reference pressure **`p`** is less than `thresh`. The argument `thresh` is expected to be a positive real number. The default value is `pthresh=0.1` mbar.
`max_iter=n`	Optional for `solgas` computations: Give the maximum number of iteration that shall be done in the iterative gas solubility calculation. The argument `n` is expected to be a positive integer number. The default value is `max_iter=1000`.

2.3.10 Density of Liquids

The **DENSITY** option allows for the automatic computation of the pure compound liquid density of a given substance. If toggled in a temperature/mixture line of the COSMO*therm* input file, the `density` option will compute the liquid densities of all compounds that are given in the compound input section at the given temperature.

The liquid density ρ_i of a pure compound *i* is computed from the corrected molar liquid volume $\tilde{V}_i$ of the compound (MW_i is the molecular weight of the compound, N_A is Avogadro's constant):

$$\rho_i = \frac{MW_i}{\tilde{V}_i N_A} \qquad (2.3.12\text{-}1)$$

The corrected molar liquid volume $\tilde{V}_i$ is computed from a Quantitative-Structure-Property-Relationship (QSPR):

$$\tilde{V}_i = c_{H_{MF}} H_i^{MF} + c_{H_{HB}} H_i^{HB} + c_{V_{COSMO}} V_i^{COSMO} + c_{M_2} M_{2_i} + c_{N_{RIng}} N_i^{Ring} + \sum\nolimits_k^{Elements} c_{A_k} A_i^k \qquad (2.3.12\text{-}2)$$

The descriptors for the corrected molar liquid volume are the pure compounds misfit interaction enthalpy H_i^{MF}, the pure compounds hydrogen bonding enthalpy H_i^{HB}, the COSMO-Volume of the compound as given in the compounds COSMO file V_i^{COSMO}, the second σ-moment of the compound M_i^2, the number of ring atoms in the compound N_i^{Ring} and the areas of surface in a given compound that belong to atoms of the same element type A_i^k, where *k* is the element number. Thus the QSPR model for the corrected molar liquid volume and density contains 6 generic parameters plus one additional parameter for each element. Starting with release version C21_0110 of COSMO*therm*, the density QSPR model of eq. 2.3.12-2 has been expanded by one additional descriptor, namely $(M_i^2)^2$ the square of the compounds second σ-moment leading to a sum of 7 generic descriptors and parameters in the model. This additional nonlinear descriptor improves the prediction for the core chemistry of neutral liquid compounds and in addition it widens the application range of the model towards ionic species, namely Ionic Liquids and zwitterionic compounds, which now can be predited with the same QSPR model and at the same quality as simple neutral compounds. The QSPR coefficients in eq. 2.3.12-2 are valid for a specific temperature only, because the density/volume QSPR model does not include a temperature dependency term. If several conformers are present for a given compound, COSMO*therm* will compute the density descriptors of all individual conformers and subsequently do a thermodynamic average of the conformer descriptors at the given temperature condition, from which the averaged density of the compound is predicted. Please note that due to its major importance, water is treated as a special case, in that the experimental value of the density and volume are given by the density option.

If possible the six generic QSPR parameters and the element specific surface area parameters are read from the COSMO*therm* parameterization file (CTDATA-file). All current (i.e. Version `C30_1701`) COSMO*therm* parameterization files include the density/volume QSPR parameters for room temperature as well as optimized element specific surface area parameters for elements H, N, C, O, F, S, Cl, Br and I. For other elements reasonable guesses for the element specific surface area parameters are provided. The parameters were derived from a set of 761 room temperature densities (including Ionic Liquids and zwitterionic compounds such as amino acids). Note that previous version of the COSMO*therm* parameter files may not include density QSPR parameters.

It is also possible to explicitly give the values of the seven density/volume QSPR parameters in the COSMO*therm* input file via the `QSPR_DENS={`c_{HMF} c_{HHB} c_{VCOSMO} c_{M2} c_{NRing} c_0 $c_{M2\wedge 2}$`}` command. The element specific surface area parameters can be given with the `EL_PARAM={k` c^{k}_{Area}`}` command (see below).

The computed liquid densities and volumes will be written to the mixture section of the COSMO*therm* output file and tabulated in the COSMO*therm* table file. In addition, for each compound the density and volume QSPR descriptors will be written to the table file. By default, the element specific surface areas are not written to the COSMO*therm* table file. However, this can be done optionally using the `PRINT_ELEM` keyword. Note that in the output and table, the liquid volume computed by the `density` option will be given as molecular volume V_l in units [Å^3] or [nm^3] depending on the given output `UNIT` format. This property is related to the molar volume $\tilde{V}_l$ via Avogadro's constant N_A: $V_l = \tilde{V}_l/N_A$.

The computation of density of a pure Ionic Liquid (IL) compound is feasible from the individual ions that form the specific IL. Thus a IL is pieced together from its anion(s) and cation(s): the composition of the IL has to be defined in the input. This is possible with the **`IL={i j …}`** command, where `i`, `j`,… are the numbers of the ion compounds (numbers as given in the sequence of compounds in the compound input section) or the **`nIL={name`$_i$ `name`$_j$ `…}`** command, where `name`$_i$, `name`$_j$, … are the names of the ion compounds as given in the compound input section. In addition the IL's stoichiometry has to be defined for the individual anion and cation compounds: The input of the IL stoichiometry factors ν_i, ν_j is possible via the **`IL_n={`ν_i ν_j `…}`** command, where ν_i, ν_j,… are the stoichiometry numbers of the IL defined by the ions as given by the `IL={i j …}` or `nIL={name`$_i$ `name`$_j$ `…}` command. For example the IL [bmim]$^+$[BF$_4$]$^-$ would be defined by the combination of the option `nIL={1-butyl-3-methyl-imidazolium0 bf4}`, defining the IL composition from bmim-cation (`1-butyl-3-methyl-imidazolium0.cosmo`) and tetrafluoroborate-anion (`bf4.cosmo`), and the option `IL_n={1 1}`, defining the stoichiometry of the salt (i.e.simple binary IL with one bmim and one BF$_4$). If these options are given as additional input to an automatic density computation ("`density`") input line, COSMO*therm* will compute the density of the IL of the given composition and stoichiometry via eqs. 2.3.12-2 and write the result to the COSMO*therm* output and table file. If the IL option is used, COSMO*therm* only will compute the density of the given IL compound (this is unlike the regular `density` option for individual compounds, which always computes the densities of all compounds present).

Suboptions of the **density** option are:

QSPR_DENS={c_{HMF} c_{HHB} c_{VCOSMO} c_{M2} c_{NRing} c_0 $c_{M2^\wedge 2}$} or QSPR_DENS_SI={c_{HMF} c_{HHB} c_{VCOSMO} c_{M2} c_{NRing} c_0 $c_{M2^\wedge 2}$}	Optional for density computations: Give the generic parameters for the QSPR approach of the corrected molar liquid volume and density. The arguments are expected as real numbers. The parameters are expected to be used with energy values in [kcal/mol] and volumes in [Å^3]. For the QSPR_DENS_SI command parameters are expected to be used with energy values in [kJ/mol] and volumes in [nm^3] (i.e. in the SI-unit frame).
EL_PARAM={k c^k_{Area}} or EL_PARAM_SI={k c^k_{Area}}	Optional for density computations: Give an element specific surface area parameter for the QSPR approach of the corrected molar liquid volume and density. Argument k is the element symbol or number and is expected as an integer. Argument c^k_{Area} is the surface area parameter and is expected as a real number. The parameters are expected to be used with surface areas in [Å^2]. For the EL_PARAM_SI command parameters are expected to be used with surface areas in [nm^2] (i.e. in the SI-unit frame). It is possible to give several EL_PARAM inputs in the same mixture line.
PRINT_ELEM	Optional for density computations: For each compound, write the element specific surface areas to the COSMO*therm* table file. In the table file the areas are given in the range: H, N, C, O, F, S, Cl, Br, I. If other elements are present in a compound the areas accordings to these elements will be tabulated in additional columns in the table file using the order of their element number.
IL={i j …} or nIL={$name_i$ $name_j$ …}	Optional for density computations: Define a Ionic Liquid from individual anion and cation compounds. The input of the Ionic Liquid composition is possible either via their compound number (IL={i j …} command, where i, j,… are the numbers of the ion compounds numbers as given in the sequence of compounds in the compound input section, the arguments i, j,…are expected to be positive integer numbers) or via their compound number (nIL={$name_i$ $name_j$ …} command, where $name_i$, $name_j$,… are the names of the ion compounds as given in the compound input section).
IL_n={ν_i ν_j …}	Optional for density computations: Define the Ionic Liquid stoichiometry from individual anion and cation compounds. The input of the Ionic Liquid stoichiometry factors ν_i, ν_j is possible via the IL_n={ν_i ν_j …} command, where ν_i, ν_j,… are the stoichiometry numbers of the Ionic Liquid defined by the ions as given by the IL={i j …} or nIL={$name_i$ $name_j$ …} command. The stoichiometry factors ν_i, ν_j,… are expected to be positive integer numbers.

2.3.11 Viscosity of Liquids

The **`VISCOSITY`** option allows for the automatic computation of the pure compound liquid viscosity of a given substance. If toggled in a temperature/mixture line of the COSMO*therm* input file, the `viscosity` option will compute the liquid viscosities of all compounds that are given in the compound input section at the given temperature.

The liquid viscosity η_i of a pure compound *i* is computed from a Quantitative-Structure-Property-Relationship (QSPR):

$$\ln(\eta_i) = c_{Area}A_i + c_{M_2}M_i^2 + c_{N_{Ring}}N_i^{Ring} + c_{TS}TS_i + c_0 \tag{2.3.13-1}$$

The descriptors for the liquid viscosity are the compounds surface area as read from its COSMO file A_i, the second σ-moment of the compound M_i^2, the number of ring atoms in the compound N_i^{Ring} and the pure compounds entropy times temperature TS_i, which is computed from the difference of the total enthalpy of mixture of the pure compound H_i and the chemical potential of the pure compound μ_i: $TS_i = -(H_i - \mu_i)$. This definition implies that the TS_i descriptor used in the viscosity QSPR model is in fact the negative entropy times temperature contribution, i.e. it is the entropic part $-TS$ of the Gibbs free energy $G = H - TS$. Altogether, the QSPR model for the liquid viscosity contains five generic parameters. The viscosity QSPR model eq. 2.3.13-1 was determined for neutral organic compounds. If it is used outside of this application range (e.g. for salts, zwitterionic compounds or metals) the predictive quality may degrade substantially. The QSPR coefficients in eq. 2.3.13-1 are valid for a specific temperature only, because the viscosity QSPR model does not include a temperature dependency term. If several conformers are present for a given compound, COSMO*therm* will compute the viscosity descriptors of all individual conformers and subsequently do a thermodynamic average of the conformer descriptors at the given temperature condition, from which the averaged viscosity of the compound is predicted.

If possible the five viscosity QSPR parameters are read from the COSMOtherm parameterization file (CTDATA-file). All current parameterizations sets of version **`C30_1701`** include the QSPR parameters for room temperature viscosity prediction. The parameters were derived from a set of 175 room temperature viscosities of organic liquids. It is also possible to explicitly give the values of the five viscosity QSPR parameters in the COSMO*therm* input file via the `QSPR_VISC=`{c_{Area} c_{M2} c_{Nring} c_{TS} c_0} command (see below). The computed liquid viscosities will be written to the mixture section of the COSMO*therm* output file and tabulated in the COSMO*therm* table file.

Suboptions of the **`viscosity`** option are:

`QSPR_VISC=`
{c_{Area} c_{M2} c_{Nring} c_{TS} c_0}
or
`QSPR_VISC_SI=`
{c_{Area} c_{M2} c_{Nring} c_{TS} c_0}
	Optional for `viscosity` computations: Give the generic parameters for the QSPR approach of the liquid viscosity. The arguments are expected as real numbers. The parameters are expected to be used with energy values in [kcal/mol] and areas in [Å^2]. For the `QSPR_VISC_SI` command parameters are expected to be used with energy values in [kJ/mol] and areas in [nm^2] (i.e. in the SI-unit frame).

2.3.12 Multi-Component Multi-Phase Extraction Equilibria

The **`LIQ_EX`** option allows for the automatic computation of a multi-component multi-phase liquid-liquid extraction equilibrium. If toggled in a temperature/mixture line of the COSMO*therm* input file, the `LIQ_EX` option will compute the mole or mass based equilibrium partition of an arbitrary number of compounds between a given number of liquid phases and optional solid (precipitation) and gaseous (evaporation) phases. The functionality of the option (two liquid phases and five compounds) is described in Figure 2.3.14-1 below:

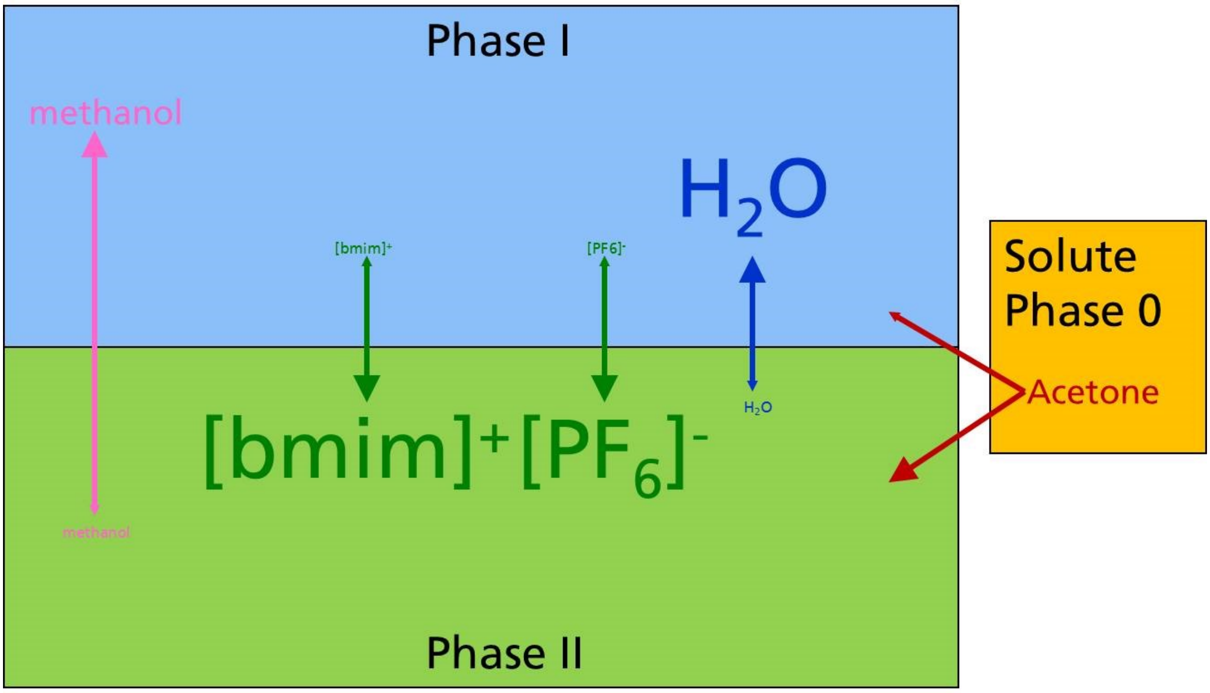


Figure 2.3.14-1: Schematic plot of liquid-liquid extraction equilibrium.

There are two predefined phases I and II both of which may be mixtures of compounds including Ionic Liquids or dissolved salts. Phases I and II are assumed to be immiscible and to separate in thermodynamic equilibrium. Using a given starting concentration in the two phases I and II, for each compound *i* (neutral as well as ions) COSMO*therm* will compute the affinity of each compound to each of the two phases (i.e. the impetus of the compound to stay in the phase where it is or to move to the opposite phase) by means of the thermodynamic equilibrium partition constant K_i^x

$$\mathrm{K}_\mathrm{i}^\mathrm{x} = \exp\left[\left(\mu_i^I - \mu_i^{II}\right)/RT\right] \tag{2.3.14-1}$$

In this expression μ_i^I and μ_i^{II} are the chemical potentials of compound *i* in phase I and phase II, respectively. Now each individual compound is allowed to move between the phases according to its computed affinity to each of the phases, with the side conditions that the sum over the two phases of the mass or mole numbers of the compounds stays the same (mass conservation) and that both phases are charge neutral (neutrality condition). In the example depicted above, phase I is the water rich phase consisting of mainly water, while phase II is a Ionic Liquid (IL) phase consisting of mainly bmim$^+$ cation and PF_6^- anion. Both phases contain a small amount of methanol. Now if the thermodynamic equilibrium partition constants K_i^x are computed and the new equlibrium concentrations of the two phases are computed it can be seen that the majority of the water will stay in phase I and only a small amount of water will move to the IL phase.

The same holds for the IL phase which mainly stays stable and minor portions of bmim$^+$ cation and PF_6^- anion move to the water phase. The polar methanol will prefer the most polar phase that allows for hydrogen bonding and thus mainly move to water rich phase I (i.e. in this example the methanol is extracted from the IL phase by the water phase). At this point the two phases now consist of different compound concentrations than at the start. Thus the chemical potentials of the compounds in the phases will change and consequently the equilibrium partition constants K_i^x will be different. Thus the computation of K_i^x has to be repeated at the new concentrations of both phases. The compounds again are distributed between the two phases due to their computed K_i^x values giving again new compositions of phase I and II. This procedure is repeated until the concentrations of the two phases do not change any more. Thus the thermodynamic equilibrium, the mass balance and, if ions are present, the charge neutrality condition (as a boundary condition of the mass balance) of the two phases are solved simultaneously in an iterative selfconsistent manner until the system converges to a thermodynamic and mass equilibrium of two neutral phases. The converged system thus provides two new phases I and II with all compounds distributed between the two phases according to their thermodynamic equilibrium partition. In this viewpoint there are no explicit solvents or solutes. Each compound (including individual ions) can move freely according to its affinity to each of the two phases - with the side conditions of charge neutrality and mass conservation. This corresponds to the solution of the liquid-liquid extraction equilibrium in this system. Please note that the model is restricted to the equilibrium calculation of liquid phases that are assumed to be immiscible (i.e. show a liquid-liquid phase separation). By default (if the `LIQ_EX` option is given without argument), it is assumed that there are two liquid phases present ($n_{phase}=2$). Using the `LIQ_EX=`n_{phase} option, it is possible to define n_{phase} phases, where $2 \leq n_{phase} \leq l_{xmx}$. The default value for l_{xmx} is 5. It can be increased by the global `lxmx` option (see section 2.1). The `LIQ_EX` option, unlike almost all of the other computation options in COSMO*therm*, can be done in the framework not of relative (mole or mass fraction) concentrations, but of the absolute amounts of substance, namely compound mole numbers (**N**) or compound masses (**W**). Thus the total amount of substances has to be read from the input as a starting point of the phase equilibration. The initial concentrations of the two phases can be given in two different ways in the COSMO*therm* input:

- Give mole numbers (**N**) of the compounds in the n_{phase} phases via options $N1=\{N^{I}_{1}\ N^{I}_{2}\ N^{I}_{3}\ \ldots\}$ and $N2=\{N^{II}_{1}\ N^{II}_{2}\ N^{II}_{3}\ \ldots\}$, where N^{I}_{i} is the mole number of compound *i* in phase I and N^{II}_{i} is the mole number of compound *i* in phase II. If more than two phases are defined via input of n_{phase}, all additional phases can be defined accordingly: $N3=\{N^{III}_{1}\ N^{III}_{2}\ N^{III}_{3}\ \ldots\}$, $N4=\{N^{IV}_{1}\ N^{IV}_{2}\ N^{IV}_{3}\ \ldots\}$, etc.

- Give masses (**W**) of the compounds in the n_{phase} phases via options $W1=\{W^{I}_{1}\ W^{I}_{2}\ W^{I}_{3}\ \ldots\}$ and $W2=\{W^{II}_{1}\ W^{II}_{2}\ W^{II}_{3}\ \ldots\}$, where W^{I}_{i} is the mass [g] of compound *i* in phase I and W^{II}_{i} is the mass [g] of compound *i* in phase II. If more than two phases are defined via input of n_{phase}, all additional phases can be defined accordingly: $W3=\{W^{III}_{1}\ W^{III}_{2}\ W^{III}_{3}\ \ldots\}$, $W4=\{W^{IV}_{1}\ W^{IV}_{2}\ W^{IV}_{3}\ \ldots\}$, etc.

For input options **N** and **W** the phase equilibrium is computed iteratively with the side condition that the total mole numbers ($N^{I}_{i} + N^{II}_{i} + \ldots$) or the total masses ($W^{I}_{i} + W^{II}_{i} + \ldots$) of each compound *i* are conserved over the sum of the all n_{phase} phases. The output of the converged system will return the mole numbers N^{I}_{i}, N^{II}_{i}, … and masses W^{I}_{i}, W^{II}_{i}, … as well as the mole fractions x^{I}_{i}, x^{II}_{i}, … of the compounds in the new equilibrium phases to the COSMO*therm* table file. In addition, for each phase equilibrium iteration step the phase equilibrium constants K_i^x and the associated equilibrium concentrations $x^K_i = K_i^x/(1+K_i^x)$ are written to the COSMO*therm* output file.

In addition to simple equilibration of the two phases starting from n_{phase} given phase concentrations it is possible to predefine the n_{phase} phases as "solvent" phases and add one or several solutes to the two phases from an external "solute" reservoir. This corresponds to adding a certain amount of an external reservoir to the two phases. In practice, the additional “solute” compounds are distributed between the $\mathtt{n_{phase}}$ given “solvent” phases according to their partition coefficient K_i^x in the initial “solvent” phases and thus are fully dissolved in the $\mathtt{n_{phase}}$ phases. Thus in the following course of the iterative solution of the system, the $\mathtt{n_{phase}}$ phases will consist of the initial solvent compounds and the added solute compound. This option may be helpful if it is not clear in advance, which phase will be preferred by a given compound when added to the system. In the above example Figure 2.3.14-1 a certain amount of acetone is added to the two-phase liquid system. Now, in the `LIQ_EX` computation the acetone is automatically equilibrated and distributed between phase I and II, giving the result that acetone prefers the IL phase and that the converged self-consistent phase equilibrium finds about 90% of the acetone in the IL-rich phase II. The “solute” phase can be given to the COSMO*therm* input file via options $\mathtt{N0=\{N^0_1\ N^0_2\ N^0_3\ \ldots\}}$ or $\mathtt{W0=\{W^0_1\ W^0_2\ W^0_3\ \ldots\}}$, where $\mathtt{N^0_i}$ is the mole number of “solute” compound *i* and $\mathtt{W^0_i}$ is the total mass of “solute” compound *i*.

In addition to the absolute mole number or mass based equilibrium calculation (**`N`** and **`W`** basis), COSMO*therm* offers the possibility of using relative amounts, namely mole or mass fractions (**`x`** or **`c`** basis) in the `LIQ_EX` computation:

- Give mole fractions (**x**) of the compounds in the $\mathtt{n_{phase}}$ phases via options $\mathtt{x1=\{x^I_1\ x^I_2\ x^I_3\ \ldots\}}$ and $\mathtt{x2=\{x^{II}_1\ x^{II}_2\ x^{II}_3\ \ldots\}}$, where $\mathtt{x^I_i}$ is the mole fraction of compound *i* in phase I and $\mathtt{x^{II}_i}$ is the mole fraction of compound *i* in phase II. If more than two phases are defined via input of $\mathtt{n_{phase}}$, all additional phases can be defined accordingly: $\mathtt{x3=\{x^{III}_1\ x^{III}_2\ x^{III}_3\ \ldots\}}$, $\mathtt{x4=\{x^{IV}_1\ x^{IV}_2\ x^{IV}_3\ \ldots\}}$, etc.

- Give mass fractions (**c**) of the compounds in the $\mathtt{n_{phase}}$ phases via options $\mathtt{c1=\{c^I_1\ c^I_2\ c^I_3\ \ldots\}}$ and $\mathtt{c2=\{c^{II}_1\ c^{II}_2\ c^{II}_3\ \ldots\}}$, where $\mathtt{c^I_i}$ is the mass fraction of compound *i* in phase I and $\mathtt{c^{II}_i}$ is the mass fraction of compound *i* in phase II. If more than two phases are defined via input of $\mathtt{n_{phase}}$, all additional phases can be defined accordingly: $\mathtt{c3=\{x^{III}_1\ c^{III}_2\ c^{III}_3\ \ldots\}}$, $\mathtt{c4=\{c^{IV}_1\ c^{IV}_2\ c^{IV}_3\ \ldots\}}$, etc.

Clearly, if the input of relative amounts of substance in a phase (i.e. mole or mass fractions `x` or `c`) is used, there is no mole number or mass conservation. Consequently the addition of “solute” compound is not possible for the mole or mass fraction based `LIQ_EX` computations. The converged relative concentration of the compounds in the $\mathtt{n_{phase}}$ phases is based on the thermodynamic equilibrium (and phase neutrality) only! Only the relative amount of compound in a phase is defining the system. This is equivalent to the “free” equlibration that is used in solubility or LLE phase diagram calculations. In fact, a `LIQ_EX` computation in mole or mass fractions `x` or `c` basis of two compounds that show phase separation will converge to the LLE miscibility gap of this system. The output of the converged system computed in the mole or mass fraction basis will return the mole fractions $\mathtt{x^I_i}$, $\mathtt{x^{II}_i}$ of the compounds in the new equilibrium phases I and II to the COSMO*therm* table file. In addition, for each phase equilibrium iteration step the phase equilibrium constants K_i^x and the associated equilibrium concentrations $x^K_i = K_i^x/(1+K_i^x)$ are written to the COSMO*therm* output file. Optionally, the converged phase equilibrium constants K_i^x, the associated phase equilibrium concentrations x^K_i, and the chemical potentials of the compounds at phase equilibrium μ^K_i may be printed to the `LIQ_EX` table in the COSMO*therm* table file. This is toggled by the keywords `pr_K` (K_i^x), `pr_xK` (x^K_i), and `pr_mu` (μ^K_i) respectively.

By default all $\mathtt{n_{phase}}$ phases in a `LIQ_EX` calculation are assumed to be liquid phases. It is however, possible to define one of the given phases as a solid (precipitation phase). This is possible with the **`SOLID`**`=k` command, where argument `k` is the phase presumed to be the container for the precipitation of solid compounds. The functionality of the solid container is described in Figure 2.3.14-2 below. It describes the dissolution/partition of a solid solute between two separated liquid phases. This requires a three-phase `LIQ_EX=3` calculation, where phase III is defined as solid precipitation phase via input option `SOLID=3`. In this example, the solid compound aspirin is dissoluted into two liquid phases consisting of a water-rich phase and a wet octanol phase consisting of 1-octanol and water. Because the aspirin solute barely is soluble in water and only moderately soluble in wet octanol, the solid cannot be dissoluted fully in both of the phases. Thus some aspirin remains in the solid phase III. The final distribution of the aspirin between the three phases corresponds to the solid-liquid-liquid equilibrium (SLLE) of the aspirin.

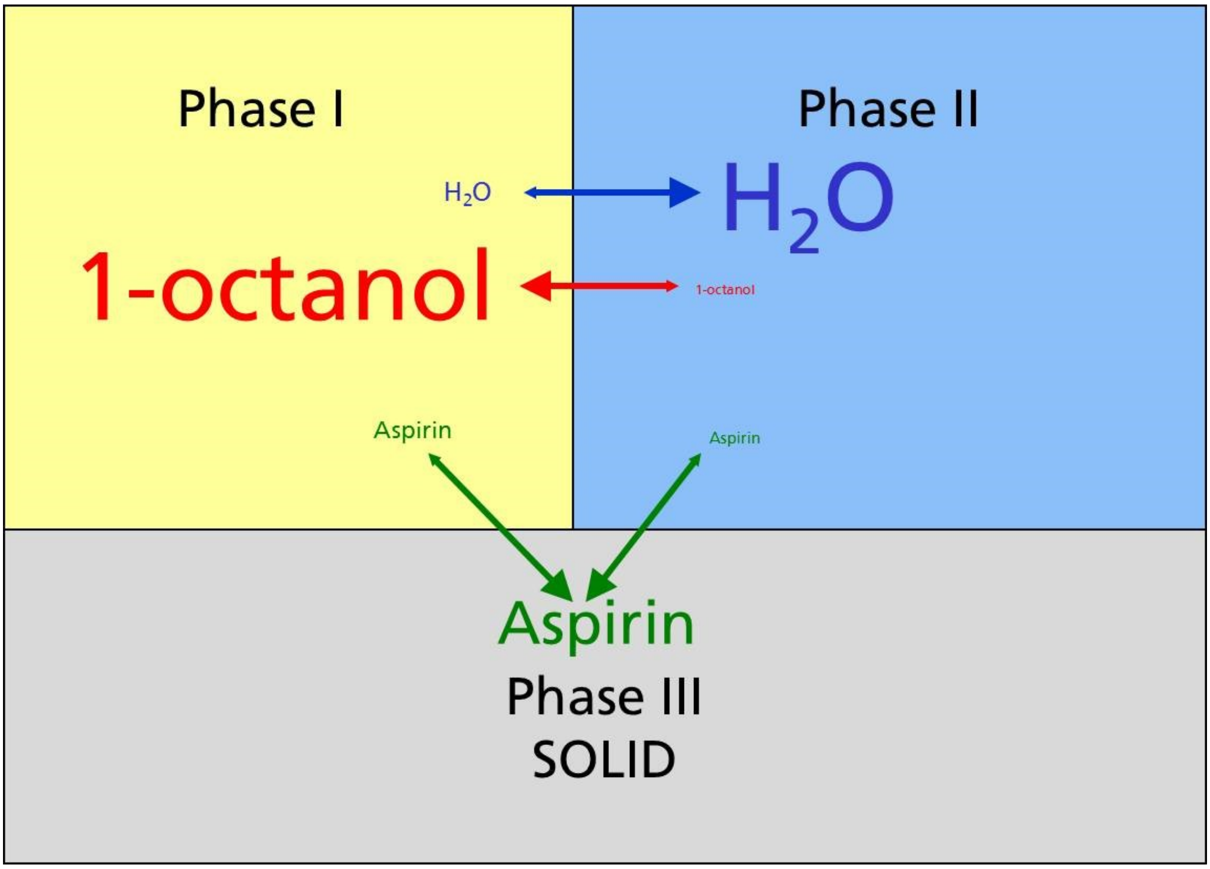


Figure 2.3.14-2: Schematic plot of solid-liquid-liquid extraction equilibrium.

The phase equilibrium constants K_i^x of the transition between a liquid phase and the solid precipitation phase are computed according to the solid state free energy of the compound (i.e. $\mu^k_i = \mu^0_i + \Delta G^i_{fus}$). Thus for a `LIQ_EX` calculation with solid phase, the Gibbs free energy of fusion ΔG_{fus} has to be taken into account for all compounds that are assumed to be solid. The solid compounds Gibbs free energy of fusion can be given in the compound input lines or read from the compounds vapor pressure/property files (vap-files). The input and processing of the Gibbs free energy of fusion data is fully equivalent to the input and processing guidelines described in section 2.3.4 "Automatic Solubility Calculation" of this manual. For further particulars on input and computation of Gibbs free energies of fusion we refer you to this section.

In addition to the definition of individual neutral compounds as solids by means of the compounds ΔG^i_{fus} it is also possible to define solid salts by means of a salts ΔG^{Salt}_{fus}. To define possible salt precipitation into the solid phase `k`, the solid salt has to be given by means of the salts composition (i.e. the ions of which it is made of), its stoichiometry, and the Gibbs free energy of fusion of the salt ΔG^{Salt}_{fus}. The input of the salt composition, stoichiometry and heat of fusion is fully equivalent to the input of these properties in the Salt Solubility option. See subsection "Solubility of Salts, Complexes and Cocrystals" of section 2.3.4 of this manual for further details. Note that in contrast to the salt solubility option, the `LIQ_EX` calculation allows the definitions of several salts. If several salt definitions are to be given these definitions simply can be given subsequently in a row, in the same `LIQ_EX` input line. By default, all salts will precipitate into the phase `k` that is defined as solid via the `solid=k` command. Alternatively, each salt can be forced to precipitate into its own separate salt phase $\mathtt{k_{salt}}$. This can be toggled by the `salt_phase=`$\mathtt{k_{salt}}$ command, where argument $\mathtt{k_{salt}}$ is the phase presumed to be the container for the precipitation of salt compound. The `salt_phase=`$\mathtt{k_{salt}}$ keyword has to be given within a salt input block (i.e. subsequent to the salts component definition, stoichiometry and $\Delta G^{Salt}{}_{fus}$ information). A salt thus defined will precipitate exclusively into its denoted salt phase $\mathtt{k_{salt}}$. No other compound can precipitate into a defined salt phase.

If ionic species are considered in the phase equilibrium in terms of dissoluted salts, or ionic liquid (IL) compounds, the convergence of the algorithm that conducts the phase equilibration may be compromised if the individual ions that form the salt or IL have strongly different affinities with respect to the different phases. This is quite common for e.g. aprotic IL's where typical IL anions such as aorganic sulfates or sulfonylimides are polar and thus tend towards polar solvent or mixture phases, while typical IL cations such as alkylimidazolium compounds are quite unipolar and tend towards nonpolar solvent or mixture phases. In such a case, the boundary condition of phase neutrality may become the dominating force in the phase equilibration, which can slow down the convergence of the `LIQ_EX` option considerably. It may even lead to divergence in severe cases. If this kind of behavior happens during a `LIQ_EX` calculation with charged components, it can be helpful to define solid or liquid salts as a salt input block (in terms of the salts component definition, stoichiometry and $\Delta G^{Salt}{}_{fus}$ information), as described above. To define a liquid salt (i.e. an IL or a salt that is completely dissoluted in the solvent phases), the Gibbs free energy of fusion $\Delta G^{Salt}{}_{fus}$ should be set to zero in the salt input block. Now, given a proper and complete salt definition, the `LIQ_EX` option can be forced to use the average chemical potential of the salt components instead of the individual chemical potentials of the salt's ion components. This option is toggled automatically for solid salts, if a separate salt precipitation phase is defined with the `salt_phase=`$\mathtt{k_{salt}}$ command. If no solid or salt phase is defined, the use of the average salt chemical potential for the salt's components can be toggled with the `avsaltmu` command. By means of using the salt's average chemical potential for all of it's ionic components, possible converge problems are avoided, which can be caused by individual ion's chemical potentials that show a tendency towards different phases. As a consequence `LIQ_EX` performance may be sped up in austere cases.

It is also possible to define one of the given phases as a gaseous vaporization phase. This can be done with the **`GASEOUS`**`=k` command, where argument `k` is the phase presumed to be the container for the evaporation of the volatile compounds. The functionality of the gas container is described in Figure 2.3.14-3 below. It describes the vapor-liquid-liquid equilibrium of two separated liquid phases, a water phase, and a wet octanol phase, with an additional vaporization container. This system can be described by a three-phase `LIQ_EX=3` calculation, where phase III is defined as vaporization phase via input option `GASEOUS=3`. The final distribution of the compounds between the three phases corresponds to the vapor-liquid-liquid equilibrium (VLLE) of the octanol-water system. The phase equilibrium constants K_i^x of the transition between a liquid phase and the gaseous vaporization phase are computed according to the gas phase free energy of the compound at the given mixture concentrations (i.e. $\mu^k{}_i = \mu_i^{Gas}$). Thus the chemical potential of the compound in the gas phase is computed in this case. The chemical potential in the gas phase is μ_i^{Gas} is computed in accordance with the pure compound vapor pressure estimate used, i.e. it is either computed from a given experimental pure compound vapor pressure value, or, if no experimental data is available for the compound, estimated by COSMO*therm* (please see section 2.2.1, 2.2.2, and 2.3.1 of this manual for details on the computation/estimation of compounds chemical potential in the gas phase μ_i^{Gas} and vapor pressure).

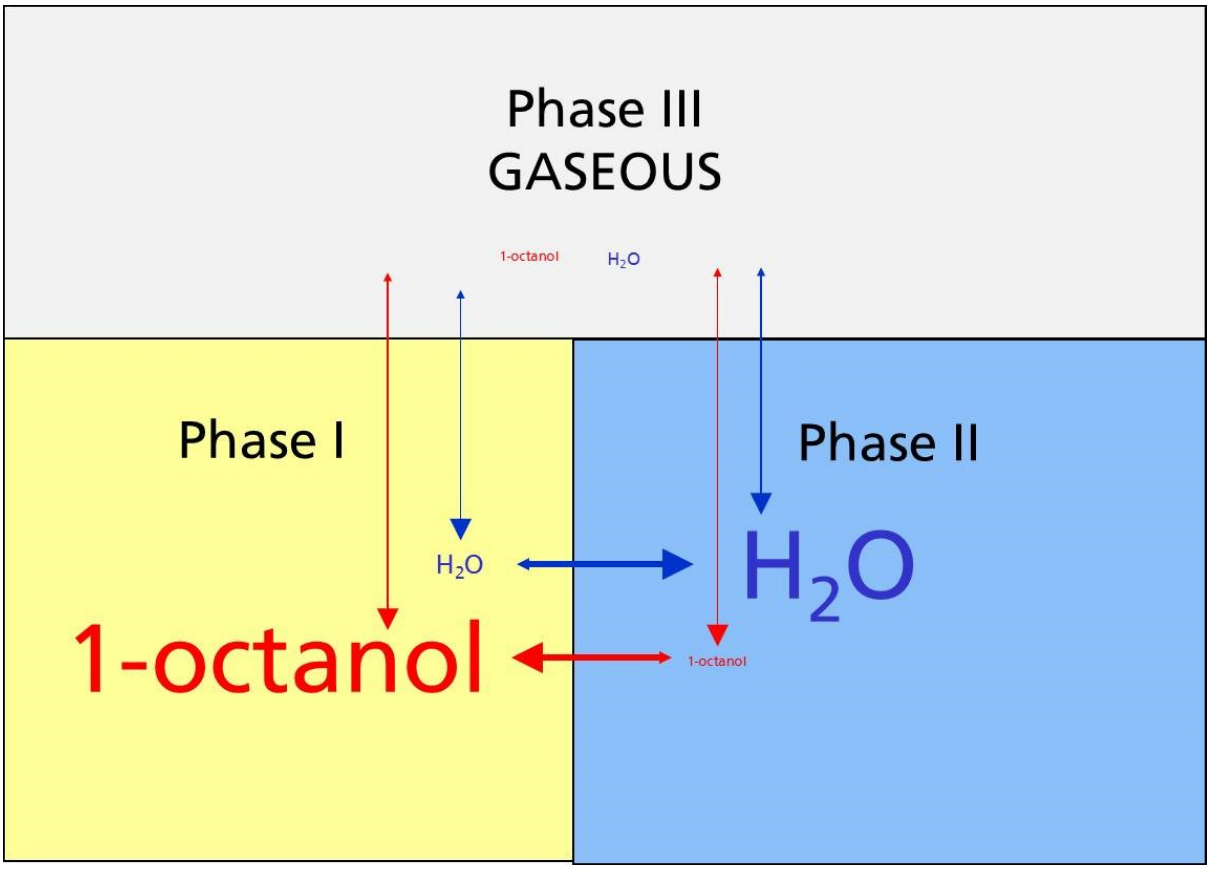


Figure 2.3.14-3: Schematic plot of liquid-liquid-vapor extraction equilibrium.

By default it is assumed that each gaseous compound in the gas container can reach an individual pure compound pressure of 1 bar. I.e. the chemical potentials of the μ_i^{Gas} of all compounds are computed due to their partial pressure above the liquid phase that is contacting the gas container. Alternatively, it is possible to define a reference pressure for the whole of the gas container. If such a reference total pressure of the gaseous container $\mathbf{p}_{\mathbf{total}}$ is given, the chemical potentials of the gas phase μ_i^{Gas} of all compounds are computed with respect to this reference pressure of the gas container as a whole (i.e. all μ_i^{Gas} values are varied in order to reproduce the given reference $\mathbf{p}_{\mathbf{total}}$). The reference pressure of the gaseous container may be given with the **PREF=**$\mathbf{p}_{\mathbf{total}}$ [mbar] keyword, or by one of its pressure unit variants `pref_Pa=`$\mathtt{p}_{\mathtt{total}}$ [Pa], `pref_kPa=`$\mathtt{p}_{\mathtt{total}}$ [kPa], `pref_psia=`$\mathtt{p}_{\mathtt{total}}$ [psia], and `pref_bar=`$\mathtt{p}_{\mathtt{total}}$ [bar], respectively. Alternatively, the molar Volume can be given via keyword `VMOL=`$\mathtt{V}_{\mathtt{mol}}$ [l/mol], which is converted to $\mathtt{p}_{\mathtt{total}}$ via ideal gas equation of state $p_{total}=RT/V_{mol}$. In addition it is possible to give an absolute reference Volume of the gaseous container using `VREF=`$\mathtt{V}_{\mathtt{Ref}}$ [l].Please note that `liq_ex` calculations currently are isothermal. This means that if one of the reference pressure, molar volume or volume options is used, the system can only try to match the given reference pressure or volume of the gaseous phase by variation of the compound concentrations in the phases. The reference pressure or volume can not be matched through variation of the temperature, which is not a degree of freedom in isothermal calculations. This implies that the reference pressure, molar volume or volume options typically will converge only if the phase equilibration is run at a temperature and concentration range that is inbetween the systems dew- and bubble-point curves. It is only at these conditions that a variation of concentrations alone can yield the given reference pressure or volume.

Temperature / mixture line input that toggles the liquid extraction calculations:

`LIQ_EX`	Toggles a multi-component two-phase liquid-liquid extraction equilibrium calculation.
$LIQ\_EX=n_{phase}$	Toggles a multi-component multi-phase liquid-liquid extraction equilibrium calculation in n_{phase} phases. Argument n_{phase} can be any integer number between 2 and 5.

Suboptions of the **`liq_ex`** option:

$N1=\{N^{I}_{1}\ N^{I}_{2}\ N^{I}_{3}\ \ldots\}$	Required for `liq_ex` computations: Give mole numbers (**N**) of the compounds initially in phase I. N^{I}_{i} is the mole number of compound *i* in phase I.
$N2=\{N^{II}_{1}\ N^{II}_{2}\ N^{II}_{3}\ \ldots\}$	Required for `liq_ex` computations: Give mole numbers (**N**) of the compounds in the phase II. N^{II}_{i} is the mole number of compound *i* in phase II.
$Nk=\{N^{k}_{1}\ N^{k}_{2}\ N^{k}_{3}\ \ldots\}$	Required for $liq\_ex=n_{phase}$ computations: Give mole numbers (**N**) of the compounds in the additional phases `k`, where $k=3\ldots n_{phase}$. N^{k}_{i} is the mole number of compound *i* in phase `k`.
$W1=\{W^{I}_{1}\ W^{I}_{2}\ W^{I}_{3}\ \ldots\}$	Required for `liq_ex` computations (alternative to `N1` option): Give absolute masses (**W**) of the compounds initially in phase I. W^{I}_{i} is the mass of compound *i* in phase I.
$W2=\{W^{II}_{1}\ W^{II}_{2}\ W^{II}_{3}\ \ldots\}$	Required for `liq_ex` computations (alternative to `N2` option): Give absolute masses (**W**) of the compounds in the phase II. W^{II}_{i} is the mass of compound *i* in phase II.
$Wk=\{W^{k}_{1}\ W^{k}_{2}\ W^{k}_{3}\ \ldots\}$	Required for $liq\_ex=n_{phase}$ computations (alternative to `Wk` option):: Give masses (**W**) of the compounds in the additional phases `k`, where $k=3\ldots n_{phase}$. W^{k}_{i} is the mass of compound *i* in phase `k`.
$N0=\{N^{0}_{1}\ N^{0}_{2}\ N^{0}_{3}\ \ldots\}$	Optional for `liq_ex` computations using mole numbers `N` or absolute masses `W`: Give mole numbers (**N**) of compounds present in an additional "solute" phase 0. N^{0}_{i} is the mole number of compound *i* in phase 0.
$W0=\{W^{0}_{1}\ W^{0}_{2}\ W^{0}_{3}\ \ldots\}$	Optional for `liq_ex` computations using mole numbers `N` or absolute masses `W`: Give absolute masses (**W**) of compounds present in an additional "solute" phase 0. W^{0}_{i} is the mass of compound *i* in phase 0.

Suboptions of the **liq_ex** option (continued):

x1={x^{I}_{1} x^{I}_{2} x^{I}_{3} …}	Required for `liq_ex` computations (alternative to `N1` or `W1` option): Give mole fractions (**x**) of the compounds initially in phase I. x^{I}_{i} is the mole fraction of compound *i* in phase I.
x2={x^{II}_{1} x^{II}_{2} x^{II}_{3} …}	Required for `liq_ex` computations (alternative to `N2` or `W2` option): Give mole fractions (**x**) of the compounds in the phase II. x^{II}_{i} is the mole fraction of compound *i* in phase II.
xk={x^{k}_{1} x^{k}_{2} x^{k}_{3} …}	Required for liq_ex=n_{phase} computations (alternative to `Nk` or `Wk` option):: Give mole fractions (**x**) of the compounds in the additional phases `k`, where k=3…n_{phase}. x^{k}_{i} is the mole fraction of compound *i* in phase `k`.
c1={c^{I}_{1} c^{I}_{2} c^{I}_{3} …}	Required for `liq_ex` computations (alternative to `N1` or `W1` option): Give mass fractions (**c**) of the compounds initially in phase I. c^{I}_{i} is the mass fraction of compound *i* in phase I.
c2={c^{II}_{1} c^{II}_{2} c^{II}_{3} …}	Required for `liq_ex` computations (alternative to `N2` or `W2` option): Give mass fractions (**c**) of the compounds in the phase II. c^{II}_{i} is the mass fraction of compound *i* in phase II.
ck={c^{k}_{1} c^{k}_{2} c^{k}_{3} …}	Required for liq_ex=n_{phase} computations (alternative to `xk` option):: Give mass fractions (**c**) of the compounds in the additional phases `k`, where k=3…n_{phase}. c^{k}_{i} is the mole fraction of compound *i* in phase `k`.
xthresh=xt	Optional for `liq_ex` computations: Set threshold for the self-consistent convergence of the liq_ex computation. Argument xt is.expected to be a positive real mole fraction number. A liq_ex computation will be considered converged if the changes of all compound mole fractions in all phases are below this threshold (default: xt = 10-5)
maxiter=imax	Optional for `liq_ex` computations: Set maximum number of iterations in the self-consistent convergence of the liq_ex computation. Argument imax is.expected to be a positive integer number (default: imax = 500).
pr_K	Optional for `liq_ex` computations: Print the converged phase equilibrium constants K_i^x to the `LIQ_EX` table in the COSMO*therm* table file.
pr_xK	Optional for `liq_ex` computations: Print the converged phase equilibrium concentrations x^K_i to the `LIQ_EX` table in the COSMO*therm* table file.
pr_mu	Optional for `liq_ex` computations: Print μ^K_i, the chemical potentials of all compounds in all phases in converged phase equilibrium to the `LIQ_EX` table in the COSMO*therm* table file.

Suboptions of the **`liq_ex`** option (continued):

`solid=k`	Optional for `liq_ex=`n_{phase} computations: Define phase `k` of the `liq_ex` computation as solid phase. Argument `k` is.expected to be a positive integer number between `1` and n_{phase}.
`Salt_phase=`k_{salt}	Optional for `liq_ex=`n_{phase} computations: Define phase k_{salt} of the `liq_ex` computation as salt precipitation phase. Argument k_{salt} is.expected to be a positive integer number between `1` and n_{phase}.
`gaseous=k`	Optional for `liq_ex=`n_{phase} computations: Define phase `k` of the `liq_ex` computation as gaseous vaporization phase. Argument `k` is.expected to be a positive integer number between `1` and n_{phase}.

`pref=p`
or
`pref_Pa=p`
or
`pref_kPa=p`
or
`pref_psia=p`
or
`pref_bar=p`
	Optional for `liq_ex=`n_{phase} computations with defined `gaseous=k` phase: use given reference pressure p for the computation of the vapor-liquid-liquid equilibrium. The input of the reference pressure `p` is possible via the `pref=p` keyword, where `p` is the pressure value that will be used to compute the equilibrium. The pressure `p` is expected to be a positive real number. For the keyword `pref` the given pressure are expected to be in [mbar], for the keyword `pref_Pa` it is expected to be in [Pa], for the keyword `pref_kPa` it is expected to be in [kPa], for the keyword `pref_psia` it is expected to be in [psia], and for the keyword `pref_bar` it is expected to be in [bar], respectively.

`vmol=`V_{mol}	Optional for `liq_ex=`n_{phase} computations with defined `gaseous=k` phase: use given molar reference volume V_{mol} for the computation of the vapor-liquid-liquid equilibrium.Argument V_{mol} is the molar volume value that will be used to compute the equilibrium pressure of the gaseous phase. V_{mol} is expected to be a positive real number volume in [l/mol].
`vref=`V_{Ref}	Optional for `liq_ex=`n_{phase} computations with defined `gaseous=k` phase: use given reference volume V_{Ref} for the computation of the vapor-liquid-liquid equilibrium. Argument V_{Ref} is the absolute volume value that will be used to compute the equilibrium pressure of the gaseous phase. V_{Ref} is expected to be a positive real number volume in [l].
`avsaltmu`	Optional for `liq_ex=nphase` computations with salt compounds: Use the average chemical potential of the salt to compute the phase equilibrium constants of the salt's components.

2.3.13 Pure Compound Critical Properties

The **PCCP** (Pure Compound Critical Property) option allows for the automatic computation of some pure compound critical properties of a given substance, namely the critical temperature T_C, the critical volume V_C, the critical pressure p_C, plus some critical-point related auxiliary properties. If toggled in a temperature/mixture line of the COSMO*therm* input file, the `PCCP` option will compute the critical data of all compounds that are given in the compound input section at the given temperature. All critical point properties are computed via different QSPR-type correlations that employ descriptors computed on-the-fly by COSMO*therm*.

The **critical temperature T_C** of a pure compound *i* is computed from the COSMO*therm* estimate of the compounds normal boiling point temperature T_{Boil}:

$$T_C = c_0 + c_1 T_{Boil} + c_2 T_{Boil}^2 \tag{2.3.15-1}$$

The three generic parameters c_0, c_1, and c_2 are stored in the COSMO*therm* parameter file (.ctd-file). All current COSMO*therm* parameterization files include the PCCP parameters for the critical temperature. Parameters c_0, c_1, and c_2 can also be read from the input line with option `tcrit_param={c0 c1 c2}` overriding the default in the ctd-file. If several conformers are present for a given compound, the overall compound T_{Boil} value will be used as descriptor in the correlation eq. 2.3.15-1.

The **critical volume V_C** of a pure compound *i* is computed from the COSMO*therm* estimate of the compounds hard-sphere volume, as defined by it's COSMO-volume V_{COSMO}:

$$V_C = c_0 + c_1 V_{COSMO} + c_2 V_{COSMO}^2 \tag{2.3.15-2}$$

The three generic parameters c_0, c_1, and c_2 are stored in the COSMO*therm* parameter file (.ctd-file). All current COSMO*therm* parameterization files include the PCCP parameters for the critical volume. Parameters c_0, c_1, and c_2 can also be read from the input line with option `vcrit_param={c0 c1 c2}` overriding the default in the ctd-file. The computation of the critical volume V_C is performed at the critical temperature as computed by eq. 2.3.15-1, above. If several conformers are present for a given compound, the compound V_{COSMO} as given by the Boltzmann-average of the conformer V_{COSMO} values at T_C will be used as descriptor in the correlation eq. 2.3.15-2. In the output and table, the critical volume V_C thus computed will be given as molecular volume in units [Å^3] or [nm^3] depending on the given output `UNIT` format. To convert this property to molar volume $\tilde{V}_C$, Avogadro's constant N_A has to be multiplied $\tilde{V}_C = V_C N_A$. Similarly, the critical density ρ_C of a pure compound *i* can be computed from the predicted critical volume V_C of the compound as $\rho_C = MW_i / V_C N_A$ where MW_i is the molecular weight of the compound, and N_A is Avogadro's constant.

The **critical pressure p_C** of a pure compound *i* is correlated with the critical temperature T_C and the critical volume V_C of the compound in a logarithmic manner: $\log_{10}(p_C) \sim \log_{10}(RT_C/V_C)$. Thus the critical pressure can be expressed as:

$$p_C = 10^{c_0 + c_1 \log_{10}(RT_C/V_C)} \tag{2.3.15-3}$$

The two generic parameters c_0, and c_1 are stored in the COSMO*therm* parameter file (.ctd-file). All current COSMO*therm* parameterization files include the PCCP parameters for the critical pressure. Parameters c_0, and c_1 can also be read from the input line with option `pcrit_param={c0 c1}` overriding the default in the ctd-file. Note that equation 2.3.15-3 does not involve any descriptors specific to COSMO*therm*. Hence the critical pressure parameters c_0, and c_1 are independent of the quantum chemistry level of the COSMO-files as well as of the version of the associated COSMO*therm* parameter set. If possible, COSMO*therm* will use experimental data of the compounds critical temperature T_C and critical volume V_C as given in the vap-file, or in the compound input line of the compound. If no experimental data is available, COSMO*therm* will use the estimates computed by equations 2.3.15-1, and 2.3.15-2, respectively.

The **critical compressibility Z_C** of a pure compound *i* is computed from the correlated critical pressure p_C, critical temperature T_C, and critical volume V_C as calculated via equations 2.3.15-1 to 2.3.15-3. I.e. the critical compressibility Z_C is not computed from a separate correlation, but assembled from the three critical property estimates shown above. Z_C is computed as:

$$Z_C = p_C V_C / R T_C \tag{2.3.15-4}$$

Note that equation 2.3.15-4 does not involve any descriptors specific to COSMO*therm*. The critical compressibility is a characteristic function of a given compound. It's value typically ranges between Z_C ~ 0.23 – 0.31 for liquid organics. A closely related characteristic function of pure fluids has been defined by Pitzer[75]. The **acentric factor** ω is defined as

$$\omega = -1 - \log_{10}\left[p_{vap}(T = 0.7T_C)/p_C\right] \tag{2.3.15-5}$$

In analogy to the critical compressibility, the acentric factor does not involve any additional correlations. It is computed from the critical pressure p_C as predicted by eq. 2.3.15-3 and a pure compound vapor pressure calculation at a temperature of 0.7_*T_C, which typically is near the compounds boiling point temperature, where the critical temperature T_C is taken from the prediction of eq. 2.3.15-1.

In addition to the critical point data proper, the PCCP option offers estimates for two other pure compound properties that are related to nonideality and the critical point, namely a nonideality correction to the temperature-dependent enthalpy of vaporization $\Delta H^{vap}{}_{corr}(T)$, and the temperature-dependent alpha-parameter $\alpha(T)$, which is a general measure for the nonideality of a system (or for the proximity to the critical point) at a given temperature. Both of these properties depend on the system temperature and are computed for the given input temperature.

The **nonideality-corrected enthalpy of vaporization $\Delta H^{vap}{}_{corr}(T)$** of a pure compound *i* is computed from the COSMO*therm* estimate of the critical temperature as defined by equation 2.3.15-1. It serves as a correction term on the regular enthalpy of vaporization $\Delta H^{vap}(T)$ as computed by the vapor pressure (`pvap`) option as described in section 2.3.1, which is computed on the basis of the assumption that the gas phase is ideal and the liquid is incompressible:

$$\Delta H^{vap}_{corr}(T) = \Delta H^{vap} * \left[1 - \ln\left(1 + c_1 e^{c_2 T/T_C}\right)\right] \tag{2.3.15-6}$$

---

[75] Pitzer, K.S., *Thermodynamics*, 3rd ed. (McGraw-Hill, New York, 1995) appendix 1.

The two generic parameters c_1 and c_2 are stored in the COSMO*therm* parameter file (.ctd-file). All current COSMO*therm* parameterization files include the PCCP parameters for $\Delta H_c^{vap}(T)$. Parameters c_1 and c_2 can also be read from the input line with option `HTcorr_param={c1 c2}` overriding the default in the ctd-file. The computation of the nonideality-corrected enthalpy of vaporization is performed for the given input temperature T. The functional form of eq. 2.3.15-6 ensures that the enthalpy of vaporization becomes zero if the compound is at or beyond the critical point temperature, which is not the case for the regular $\Delta H^{vap}(T)$ computed with the ideality assumption.

The **alpha parameter $\alpha(T)$** of a pure compound i is computed from the COSMO*therm* estimate of the critical temperature as defined by equation 2.3.15-1. $\alpha(T)$ of a compound i is estimated with help of the critical temperature T_C as defined by equation 2.3.15-1, and the nonideality-corrected enthalpy of vaporization $\Delta H^{vap}{}_{corr}(T)$ as defined by equation 2.3.15-6:

$$\alpha(T) = c_0 + c_1 \Delta H_{corr}^{vap}(T) + c_2(T/T_C) + c_3\sqrt{(T/T_C)} \tag{2.3.15-7}$$

The four generic parameters c_0, c_1, c_2, and c_3 are stored in the COSMO*therm* parameter file (.ctd-file). All current COSMO*therm* parameterization files include the PCCP parameters for $\alpha(T)$. The computation of the alpha parameter is performed for the given input temperature T. The compounds alpha parameter $\alpha(T)$ is a measure for the nonideality of the compound at the given temperature T. For compounds at or beyond the critical point $\alpha(T)$ by definition becomes 1. The value increases as the gas phase becomes more ideal. The alpha parameter was derived in the framework of the Patel-Teja equation of state (PT-EoS)[76], where it serves as temperature dependency of the cohesion pressure (attraction parameter *a*). Thus $\alpha(T)$ can be used to determine the adjustable parameters of the PT-EoS[77].

Temperature / mixture line input that toggles a PCCP calculation:

`PCCP`	Toggles a Pure Compound Critical Property (PCCP) calculation for all given pure compounds.

Suboptions of the **PCCP** option:

`tcrit_param={c0 c1 c2}`	Optional for PCCP computations: input of parameters for the critical temperature estimation.
`vcrit_param={c0 c1 c2}`	Optional for PCCP computations: input of parameters for the critical volume estimation.
`pcrit_param={c0 c1}`	Optional for PCCP computations: input of parameters for the critical pressure estimation.
`HTcorr_param={c1 c2}`	Optional for PCCP computations: input of parameters for the nonideality-corrected enthalpy of vaporization estimation.

---

[76] Patel, N.; Teja, A., *Chem. Eng. Sci.* **37**, 463 (1981).

[77] Forero, L.A.; Velázquez, J.A., *J. Chem. Eng. Data* **55**, 5094 (2010).

2.3.14 Flash Point

The **`flashpoint`** option allows for the automatic computation of the flash point (FP) of pure compounds and mixtures. The flash point is the temperature at which vapor over a liquid will combust completely upon ignition. The flash point T_{FP} corresponds to a saturation pressure p_{FP}, which by definition is the theoretical lower flammability limit at the FP. Thus for a pure compound *i* we have $p_{i,FP} = p_i^0(T_{FP})$. The prediction of the pure compound's T_{FP} thus is a variational minimization of $| p_i^0(T) - p_{i,FP} |$, wherein $p_i^0(T)$ is the pure compound vapor pressure of compound *i*, which either can be predicted by COSMO*therm*, or computed from experimental vapor pressure data as provided in the input (see section 2.2.3 for details on the latter). The convergence threshold for the optimization of $p_{i,FP}$ (i.e. the numerical accuracy of the flash point prediction) is 10^{-4} mbar by default. This value can be changed with the `thresh_pflash=Δp` option.

The variational minimization of the FP saturation pressure is started at the given input temperature *T*. By default, COSMO*therm* will use it's own pure compound vapor pressure estimate for $p_i^0(T)$ as computed by eq. 2.3.1-1 of section 2.3.1 of this manual. This prediction value can be improved with experimental data via eq. 2.3.1-4 in section 2.3.1, using a reference pressure and temperature, a temperature dependent pressure curve, or the boiling point temperature of the compound. Most of the possibilities to include a reference pressure and temperature into the `pvap` prediction option that are explained in section 2.3.1 can also be applied in the `flashpoint` prediction option: it is possible to use a reference temperature T_{Ref} and pressure p_{Ref} pair, which can be given in the temperature / mixture input line together with the `flashpoint` option. Input of a reference temperature T_{Ref} is possible via keywords `use_tref`, `use_tref_C`, `use_tref_K`, or `use_tref_F`, processing reference temperature values in units degree Celsius (for the first two keywords), Kelvin, and Fahrenheit, respectively. Input of reference pressure p_{Ref} is possible via keywords `use_pref`, `use_pref_Pa`, `use_pref_kPa`, `use_pref_bar`, or `use_pref_psia`, processing reference pressure values in units millibar, Pascal, kiloPascal, bar, and psia, respectively. Further, there is the possibility to use a compound's experimental normal boiling point temperature as reference point. The compound's boiling point temperature, as read from the compound input line, or from the vapor pressure property file (see section 2.2.3) is used as reference temperature $T_{Ref} = T_{Boil}$ assuming normal boiling point conditions, thus the reference pressure used is $p_{Ref}(T_{Ref}) = 1$ atm = 1013.25 mbar. The usage of pure compound boiling points in a pure compounds vapor pressure prediction (corresponding to the global `use_tboil` option) is done by default in `flashpoint` calculations. It can be switched off locally with the keyword `use_tboil=OFF` in the `flashpoint` mixture input line, where the option is deactivated for the given mixture. In addition to the use of boiling points as reference pressures COSMO*therm* offers the possibility to use the compound's experimental vapor pressure as reference point. The temperature dependent vapor pressure, as read from the compound input line, or from the vapor pressure property file (see section 2.2.3) in terms of pairs of experimental $p_i^0(T)$, or, Antoine, extended Antoine, or Wagner equation coefficients, directly can be used in the variational minimization of $| p_i^0(T) - p_{i,FP} |$. The usage of temperature dependent pure compound vapor pressures (corresponding to the global `use_pvapt` option) is done by default in `flashpoint` calculations. It can be switched off locally with the keyword `use_pvapt=OFF` in the `flashpoint` mixture line, where the option is deactivated for the given mixture.

The pure compound FP saturation pressure $p_{i,FP}$ is estimated from a simple linear correlation[78]:

$$\ln(p_{i,FP}) = c_0 + c_1 ln(a_i) \qquad (2.3.16\text{-}1)$$

The only descriptor involved in equation 2.3.16-1 is the molecular hard-sphere area a_i, which is read from the compound COSMO-file. The two generic fit parameters c_0, and c_1 are stored in the COSMO*therm* parameter file (.ctd-file). All current COSMO*therm* parameterization files include the FP parameters for p_{FP}. A first estimation of the parameters can be found in Ref. 78. The $p_{i,FP}$ fit parameters also can be read from the temperature / mixture input line via keyword `flashpoint_param={c`$_0$ `c`$_1$`}`, overriding the default values read from the parameter file. Alternatively, it is also possible to give the pure compound FP saturation pressure $p_{i,FP}$, or the pure compound FP temperature $T_{i,FP}$ in the compound input line or vap-file via keywords `flash_psat=p`$_i$ for $p_{i,FP}$ in [mbar], `flash_psat_bar=p`$_i$ for $p_{i,FP}$ in [bar], `flash_psat_kPa=p`$_i$ for $p_{i,FP}$ in [kPa], `flash_psat_psia=p`$_i$ for $p_{i,FP}$ in [psia], `flash_temp_K=T`$_i$ for $T_{i,FP}$ in [K], or `flash_temp_C=T`$_i$ for $T_{i,FP}$ in [°C], respectively (see section 2.2.3 for details on pure compound property input). The pure compound FP data thus given will be used with priority in the FP prediction, overriding the estimate of eq. 2.3.16-1. Thus in the case of a pure compound FP calculation, the given experimental data will just be reproduced on output.

In the case of mixtures the flash point is computed with the mixing rule of Liaw *et. al.*[79] The FP of the mixture is computed by a variational minimization of the general flash point condition stating that at the FP, the lower flammability limit should be identical to the saturation vapor pressure. In the case of liquid compound mixtures this condition becomes:

$$\frac{1}{T^{flame}-T}\sum_{i\neq k}\frac{y_i(T^{flame}-T_{i,FP})}{p_{i,FP}} = 1 \qquad (2.3.16\text{-}2)$$

The flame temperature T^{flame} is assumed to be an adjustable parameter. By default it is taken to be T^{flame} = 1300 °C = 1573 K[78]. It is also possible to provide T^{flame} in the temperature / mixture input of the FP calculation via keyword `flash_Tflame_C=T`^{flame} or `flash_Tflame_K=T`^{flame}, where argument T^{flame} is excpected to be given in [°C], and [K], respectively. The mole fraction concentration of the mixture components in the gas phase y_i is computed by COSMO*therm* at the given temperature T as $y_i = x_i p_i^0 \gamma_i / p_{tot}$. In addition, equation 2.3.16-2 requires the pure compound FP temperature $T_{i,FP}$ and the pure compound saturation pressure $p_{i,FP}$ of all flammable molecules that are part of the mixture. The pure compound FP properties either are read from the compound input line or vap-file as described above (see section 2.2.3 for details on pure compound property input), or are estimated on the fly by COSMO*therm* using eq. 2.3.16-1 and the variational optimization procedure described above. In the case of mixture FP predictions it is possible to define inert components of the mixture (denoted by compound index k in eq. 2.3.16-2). Common examples for inert compounds are water and carbon dioxide. An inert compound by definition

---

[78] Reinisch, J.; Klamt, A., *Ind. Eng. Chem. Res.* **54**, 12974 (2015).

[79] Liaw, H.-J.; Lu, W.-H.; Tang, C.-L.; Hsu, H.-H.; Liu, J.-H. *J. Loss Prev. Process Ind.* **15**, 429 (2002); Liaw, H.-J.; Chiu, Y.-Y. *J. Hazard Mater.* **101**, 83 (2003); Liaw, H.-J.; Lu, W.-H.; Gerbaud, V.; Chen, C.-C. *J. Hazard Mater.* **153**, 1156 (2008); Liaw, H.-J.; Chen, C.-T.; Gerbaud, V. *Chem. Eng. Sci.* **63**, 4543 (2008); Chen, H.-Y.; Liaw, H.-J. *Procedia Eng.* **45**, 507 (2012).

must have no influence on the computed flash point and thus is excluded from the summation in eq. 2.3.16-2. Inert mixture components can be defined in the temperature / mixture input line of the FP calculation with the keywords `flash_inert={`k_1 k_2 …`}` or `flash_ninert={`$name_1$ $name_2$ …`}`, where arguments k_n are compound numbers, and $name_n$ are compound names as given in the compound input.

Temperature / mixture line input that triggers a flash point calculation:

`flashpoint[=i]`
or
`nflashpoint[=`$name_i$`]`
	Toggles a Flash Point (FP) calculation. If no argument is provided, the FP calculation will be done for the mixture concentration that is given in the temperature/mixture line. If the `flashpoint=i` keyword is given, a pure compound FP calculation will be done for the i^{th} compound in the compound input section. If the `nflashpoint=`$name_i$ keyword is given, a pure compound FP calculation will be done for the compound named $name_i$ as found in the compound input section.

Suboptions of the **`flashpoint`** option (pure compound and mixture FP):

`thresh_pflash=Δp`	Optional for `flashpoint` computations: input of convergence threshold for the optimization of the pure compound FP saturation pressure $p_{i,FP}$. Argument Δp is expected to be a pressure in [mbar]. By default COSMO*therm* uses Δp = 10^{-4} mbar.
`flashpoint_param={`c_0 c_1`}`	Optional for `flashpoint` computations: input of parameters for the pure compound FP saturation pressure estimation.
`use_tboil`	Optional for `flashpoint` computations: Use the pure compound boiling points T_{Boil} (as given in the compound input section or read from a compounds vapor-pressure property file, cf. section 2.2.3) as a reference point for scaling the vapor pressure prediction that is part of the FP temperature prediction. The `use_tboil` option is valid for pure compound and mixture vapor pressure predictions.
`use_pvapt`	Optional for `flashpoint` computations: Use the temperature dependent experimental pure compound vapor pressure $p_i^0(T)$, as given in the compound input section or read from a compounds vapor-pressure property file, cf. section 2.2.3, as a reference point for scaling the vapor pressure prediction that is part of the FP temperature prediction. The `use_pvapt` option is valid for pure compound and mixture vapor pressure predictions.

Suboptions of the **`flashpoint`** option (pure compound and mixture FP, continued):

`use_tref=value`
or
`use_tref_C=value`
or
`use_tref_K=value`
or
`use_tref_F=value`
	Optional for `flashpoint` computations: Use a reference point for scaling the vapor pressure prediction that is part of the FP temperature prediction. This is possible for pure compound and mixture FP predictions. To be active it is necessary to input a pair of given reference temperature T_{Ref} and pressure p_{Ref} (see below). The argument of the `use_tref=value` option is expected to be a real number temperature, which is expected to be in [°C] for keyword `use_tref` and `use_tref_C`, in [K] for keyword `use_tref_K`, and in [F] for keyword `use_tref_F`.

`use_pref=value`
or
`use_pref_Pa=value`
or
`use_pref_kPa=value`
or
`use_pref_bar=value`
or
`use_pref_psia=value`
	Optional for `flashpoint` computations: Use a reference point for scaling the vapor pressure prediction that is part of the FP temperature prediction. This is possible for pure compound and mixture FP predictions. To be active it is necessary to input a pair of given reference temperature T_{Ref} (see above) and pressure p_{Ref}. The argument of the `use_pref=value` option is expected to be a real number pressure, which is expected to be in [mbar] for keyword `use_pref`, in [Pa] for keyword `use_pref_Pa`, in [kPa] for keyword `use_pref_kPa`, in [bar] for keyword `use_pref_bar`, and in [psia] for keyword use_pref_`psia`.

Suboptions of the **`flashpoint`** option specific to mixtures:

`flash_Tflame_C=value`
or
`flash_Tflame_K=value`
	Optional for `flashpoint` computations of mixtures: input of the flame temperature T^{flame}. The argument value is expected to be a temperature in [°C] for option `flash_Tflame_C` and in [K] for option `flash_Tflame_K`. By default COSMO*therm* uses T^{flame} = 1300 °C = 1573 K.

`flash_inert={`k_1 k_2 `…}`
or
`flash_ninert={`$name_1$ $name_2$ `…}`
	Optional for flashpoint computations of mixtures: define inert mixture components. The arguments k_n of option `flash_inert` are compound numbers as given in the compound input. The arguments $name_n$ of option `flash_ninert` are compound names as given in the compound input.

2.4 Input Examples

As it may be quite cumbersome to learn all of COSMO*therm*'s capabilities by trial in the graphical user interface, or, which may be even more time-consuming, by systematically processing the vast amount of different input options and property computation methods in the previous sections, the COSMO*therm* release package also includes a large number of example inputs. These example inputs serve two purposes: 1) demonstrating the syntax of the COSMO*therm* input, and the applicational possibilities of COSMO*therm*, in different properties, in particular for users of the command line, and 2) demonstrate typical applications, properties and tasks with the help of specific examples that are close to inputs used for practical application work with COSMO*therm*.

The example inputs can be found in the `\COSMOlogicAppData\COSMOthermX15\Examples\` directory, which is located in the current user directory (`\Users`) in Windows OS and in the home directory (~) in Linux and Mac OS. Thus for a typical installation with default paths and default program package name you will find the example inputs in

`C:\Users\*YourUserName*\COSMOlogicAppData\COSMOthermX15\Examples\` (Windows)

`~/COSMOlogicAppData/COSMOthermX15/Examples/` (Linux/Mac)

Users of the graphical user interface COSMO*thermX* can access the example inputs via the pull down menu **"Help"** → **"Open Example"**, which will open an example input file chooser in the …`\COSMOlogicAppData\COSMOthermX15\Examples\` directory thus allowing the user to directly read in and run the example input of his choice.

Note that the input examples in the …`\COSMOlogicAppData\COSMOthermX15\Examples\` directory may be modified or even deleted. A backup of the original example inputs as delivered with the COSMO*therm* release can be found in the COSMO*therm* release installation, which for a default installation is `\COSMOlogic\COSMOthermX15\COSMOtherm\EXAMPLE-INPUTS\`. The COSMO*therm* release installation is located in the program installation directory of the given OS. I.e. for a default installation the install directory is in `C:\Program Files (x86)\` or `C:\Program Files\`for 32-bit and 64-bit Windows OS respectively, and in the home directory ~ for Linux and Mac OS.

The example inputs are given as command line input file `name.inp`. The example inputs are named by the thermodynamic property that is calculation with the example. The example inputs can be run both from the command line as well as from COSMO*thermX*. As demonstrated in example 2.4-1, the example inputs do not only hold the bare input options required for the running of the compound in COSMO*therm* or COSMO*thermX,* but they also include some comments on the purpose of the specific input, and a comprehensive explanation of the options and keywords used in the input example. Users unexperienced with the command line syntax of COSMO*therm* can use these inputs as template or model for the generation of their own inputs for their own specific tasks.

`Example 2.4-1:` **Example input file `HENRY.inp` as taken from the `\Examples\` directory:**

```
ctd=BP_TZVP_C30_1701.ctd cdir=../CTDATA-FILES ldir=../../licensefiles # Global command line 1
fdir=../DATABASE-COSMO/BP-TZVP-COSMO efile vpfile UNIT=SI             # Global command line 2
! Automatic Henry law coefficient computation                         # Comment line
f = h2o_c0.cosmo                                                    # Compound input solvent water
f = hexane_c0.cosmo                                                 # Compound input solute  Exp: k_H = 9217 [MPa]
f = co_c0.cosmo                                                     # Compound input solute  Exp: k_H = 5843 [MPa]
f = ch2cl2_c0.cosmo                                                 # Compound input solute  Exp: k_H = 13.8 [MPa]
f=1-octanol_c0.cosmo [ comp=octanol                                 # Compound input solute conformer 1 Exp: k_H = 0.14 [MPa]
f=1-octanol_c1.cosmo ]                                              # Compound input solute conformer 2 Exp: k_H = 0.14 [MPa]
tc=25 Henry=1 wcomp={2 3 4 5}                                       # Automatic Henry law constant computation
#
# Computation of Henry law constants k_H with automatic procedure.
#
# High quality COSMO-files (BP-TZVP-COSMO) and appropriate parametrizations are used !
#
# The computed k_H values can be found in the mixture output section of output-
# file HENRY-LAW-CONSTANT.out and in the table-file HENRY-LAW-CONSTANT.tab.
#
#
# Options used      - Explanation
#
# ctd=...           - give name of parameterization file
# cdir=...          - give directory (search path) of parameterization file
# fdir=...          - give directory (search path) of compounds COSMO/CCF-files
# efile             - use gas phase energy file "compound.energy" from "fdir" in the
#                     prediction of the Henry law constants. If no "compound.energy"
#                     file is found in "fdir", COSMOtherm will do an empirical estimate
#                     for the pure compound vapor pressure (which is required in the
#                     Henry law constant computation). Thus if no gas phase energy values
#                     is available for a given compound, the quality of the COSMOtherm
#                     prediction will be lower.
# vpfile            - use pure compound vap pressure/property file "compound.vap" from
#                     "fdir" in the prediction of the Henry law constants. The experimental
#                     vapor pressure (pvExp) and the Henry law coefficient that is computed
#                     from the experimental vapor pressure of the vap-file (HpvExp) are
#                     printed to the last two columns of the HENRY table in the table file.
# UNIT=SI           - print option: all output will be in SI-UNITS
#
# f=*.cosmo         - compound input
# comp=name         - change the name of the compound. This name will be written to the table file.
# [...]             - conformer input, square brackets identify start and end of an conformer block.
#                     Each conformer has to be given in a separate line. All molecules given within
#                     this conformer block will be regarded by COSMOtherm as one single compound!
#                     The contribution of the different conformers are weighted due to their
#                     relative free energies in the mixture.
#
# Henry=1           - toggles automatic k_H computation in solvent 1 (i.e. water)
#                     See COSMOtherm user manual section 2.3.3.
# tc=25             - temperature in [°C]
# wcomp={2 3 4 5}   - print option: write output to output file only for compounds 2, 3, 4
#                     and 5 (i.e. not for solvent water = compound 1). This option is
#                     valid only for the output file, it does not target the table file.
#
```

In addition to the command line input examples demonstrated in example 2.4-1, a number of input examples specific to the graphical user interface COSMO*thermX* can be found in the `/SCREENING/` subdirectory of the `/Examples/` directory. These example inputs are given as `name.inpx`, i.e. in XML-format, which can only be processed by COSMO*thermX* not by command line COSMO*therm*. XML-format input files can be identified by their extension `.inpx`. See example 2.4-2. Note that the screening examples are created by COSMO*thermX* and should not be edited manually.

`Example 2.4-2:` Example input file `SOLUB-SCREEN-SOLID-REFERENCE.inp` as taken from the `\Examples\SCREENING\` directory:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!--
    Document   : SOLUB-SCREEN-SOLID-REFERENCE.xml
    Created on : 1. September 2013, 14:58
    Author     : COSMOthermX
    Description: Solubility solvent screening for solute theophylline using a reference solubility.
-->
<screening>
    <property state="solid">Solubility</property>
    <kindOfScreening>4</kindOfScreening>
    <compoundList conformer="yes" parametrization="TZVP">
    <compound>
         <filename>theophylline_c0.cosmo</filename>
         <filepath>..\COSMOtherm\DATABASE-COSMO\BP-TZVP-COSMO\t</filepath>
      </compound>
     …
```

2.5 The COSMO Database

COSMO*therm* allows for simple and efficient processing of large numbers of compounds (i.e. a database of molecular COSMO files; e.g. the COSMO*base* database[80]).

One aspect of the easy processing of large datasets is the use of the auxiliary program **`CT_CREATE`** that is shipped with COSMO*therm*. `CT_CREATE` is a simple tool that is able to automatically create COSMO*therm* input files over a dataset of COSMO files, run these COSMO*therm* jobs and scan their output for the required data. For further information, cf. the `CT_CREATE` user's manual[81].

The second possibility to process large datasets of COSMO files is the **`$DATABASE=filename`** command which can be used in the compound input section of the COSMO*therm* input file. This option reads in a database list file of the name `filename` which can be used in connection with the `f=filename.cosmo` command (which then is replaced by `f=$DATABASE=listfile`), the `rn=aaaaaa-bb-c` command/RN (which then is replaced by `rn=$DATABASE=listfile`) or the `dbn=name` command (which then is replaced by `dbn=$DATABASE=listfile`). Basically the database list file that is read in with this option is a simple text file holding a list of COSMO file filenames, Chemical Abstracts / Registry Numbers (CAS/RN) or trivial names, respectively. The filenames, CAS/RN or trivial names in the database list file are then processed as if they were directly given in the COSMO*therm* input file. It is required that the first column of the database list file either holds COSMO file filenames, the CAS/RN, or the trivial name. Additional columns in the database list file (separated by blank spaces) will also be processed by COSMO*therm* and can be used to give additional data for the actual compound, for example vapor pressure information, conformer identifiers "`[`" and "`]`" or gas phase energies. Thus a typical database list file might look like this:

```
methanol.cosmo vpexp={0.10 -69.00  10.00 -20.40  1000.00 64.20}
ethanol.cosmo  vpexp={0.10 -56.00  10.00  -7.00  1000.00 78.00}
…
```

Please note that all additional commands given in the compound input line of the COSMO*therm* input file that holds the `$DATABASE=filename` command will be processed for each of the compounds given in the database list file. COSMO*therm*s processing of a list of compounds involves the simultaneous storage of all the COSMO information in the memory of the computer. The number of COSMO files to be processed by the `$DATABASE=filename` option is restricted to 200 since larger number of molecules processed simultaneously would unnecessarily lead to a strong increase of COSMO*therm*s requirements regarding computer memory as well as slow down the performance of COSMO*therm*. If larger amount of compounds shall be processed it is much more efficient to use the auxiliary program `CT_CREATE` (see above).

COSMO*therm* additionally allows the usage of Chemical Abstracts / Registry Numbers (CAS/RN) to identify compounds via the **`rn=registry-number`** command/RN in the compound input section. It also allows the usage of trivial names to identify compounds via the **`dbn=name`** command in the compound input section.

---

[80] COSMO*base* is a database of molecular COSMO files available from COSMO*logic* GmbH & Co KG. Currently COSMO*base* consists of over 2000 compounds including a large number of industrial solvents plus a wide variety of common organic compounds. All compounds in COSMO*base* are indexed by their Chemical Abstracts / Registry Number (CAS/RN), by a trivial name and additionally by their sum formula and molecular weight, allowing a simple identification of the compounds. Currently COSMO*base* is available for the following quantum chemical methods and basis sets: Turbomole BP-TZVP, Turbomole BP-SVP-AM1.

[81] Eckert, F., *CT_CREATE Users Manual*, Version C3.0 Release 17.01, COSMO*logic* GmbH & Co. KG (2014).

If the `rn` command is used, COSMO*therm* will assign a compound name to the given CAS/RN and read in the corresponding molecular COSMO file[82]. Trivial names given by the `dbn=name` command are processed similarly. In any case, the use of the `rn=registry-number` or the `dbn=name` option requires the processing of a COSMO **database index file**, which maps the CAS/RN and the compounds trivial names to the name of the according COSMO file. In COSMO*therm* Version C21 Revision 0109 and later, the database index file is searched for in a hierachical manner: with highest priority the database index file is read from a directory that is one branch above the global COSMO file directory as given by the `fdir` command (this matches the database handling of COSMO*therm*X, the graphical user interface of COSMO*therm*, which expects the database index file to be located one directory above the actual COSMO file directory); if it is not found there, it is looked for in the directory of the COSMO*therm* parameter files given via the `cdir` command or read from the environment variable `$COSMOTHERM_HOME`; if neither found above `fdir` nor in `cdir`, it is searched for directly in fdir and, if not found there either, in the current working directory. For all older versions of COSMO*therm* the COSMO database index file is expected to be in the directory of the COSMO*therm* parameter files (i.e. the current working directory or the directory given via the `cdir` command or read from the environment variable `$COSMOTHERM_HOME`). By default COSMO*therm* expects a COSMO database index file of the name **`DATABASE-COSMO.csv`**. However, this filename can be changed via the **`dbas=filename`** command in the global command section of the COSMO*therm* input file. The COSMO database index file is expected as a plain ASCII text file in the „comma separated file" (CSV) format, i.e. all entries are separated by commas „`;`". The CSV format can be written by all common spreadsheet programs, such as Microsoft-Excel™ or Lotus-123™. The first entry in the index file has to be the compound name, followed by the CAS/RN, the molecular weight, the sum formula and a trivial name:

```
1-butanol;000071-36-3;74.1224;C4H10O;BUTANOL;
2-butanol;000078-92-2;74.1224;C4H10O;S-BUTANOL;
…
```

The CAS/RN is expected in the form `aaaaaa-bb-c` and must not exceed 11 digits including the separators. The leading zeroes of the CAS/RN might be left out (i.e. it is possible to give `79-09-4` instead of `000079-09-4`). This also holds for the `rn=registry-number` command in the COSMO*therm* input file. Please also note, that the trivial name in the last field of the database index file must not contain any blank spaces. This also holds for the trivial name given by the `dbn=name` option. Thus, if the above COSMO database index file is processed, the compound input expression `f=1-butanol.cosmo` in the COSMO*therm* input file can be replaced by the term `rn=71-36-3` or by the term `dbn=butanol`. Please note, that the `dbn=name` command is case-insensitive.

Starting with Version C1.2 Revision 01.03 COSMO*therm* is able to read in additional information about different conformers of one compound in the database index file. This new database index file has the format: `COSMO-Name ; CAS-Number ; MW ; Formula ; Alternative_Name ; Conformer1_Name ; Conformer1_Alternative_Name; Conf2_Name ; Conf2_AltName ; Conf3_Name ;`

---

[82] For a given CAS/RN and its corresponding compound name „`name`" , COSMO*therm* will search for the file `name.cosmo` in the COSMO file directory (i.e. the current working directory or the directory given with the `fdir` command in the global command section). If no `name.cosmo` could be found, COSMO*therm* will search for `name.cos` (MOPAC/COSMO format). If such a file also cannot be found, COSMO*therm* will search for a COSMO-metafile of the corresponding name `name.mcos`. If none of the three file-types were found for the compound name, COSMO*therm* will terminate and return an error message.

`Conf3_AltName ; Conf4_Name ; Conf4_AltName ;` … I.e. the additional conformers are attached to the database index list shown above as additional entries, with two additional fields for each conformer: first the conformers COSMO filename (without extension) and then, separated by a comma „`;`“, the conformers trivial name. Up to nine additional conformers can be processed. For example, the compound valine that consists of two conformers is given in the database index file as

```
VALINE0;000072-18-4;117.1474;C5H11NO2;L-VALINE-conformer-0;VALINE1;L-VALINE-
conformer-1;;;;;;;;;;;;;
```

By default only the first conformer is read in if the `rn=xxxxxx-xx-x` or `dbn=trivialname` options are used. However, it is also possible to automatically process all conformers of a compound that are given in the database index file of the new format. This is toggled by the command "**`dbco`**" given in the global command section of the COSMO*therm* input file.

Starting with Version C3.0 Revision 14.01 COSMO*therm* is able to read an alternative file format of the COSMO database index file, which is created by the "COSMO*base*-Editor" functionality of COSMO*therm*X. Database index files created by the COSMO*base*-Editor functionality are written in the extensible markup language (XML) format. They are recognized by the extension `.xml`. The **`dbas=filename.xml`** input allows a COSMO file handling that is fully equivalent to the usage of a database index file in CSV format: the `dbco`, `dbn`, and `rn` commands can be used the same way as for a `.csv` database index file. The main difference of the `.xml` and the `.csv` index files is that the latter is restricted to 10 conformers, while for the XML format index file there is no such restriction.

2.5.1 Creating COSMO files with a quantum chemistry program

COSMO*therm* (Version C3.0 Release 17.01) is able to read COSMO-information created by the following program packages and method / basis set combinations:

- **Turbomole**[7]: RI-DFT with BP (=B88-VWN-P86) functional and "def-TZVP" basis set.
- GAMESS[10], PQS[11], Molpro[12], Columbus[13], ORCA[14], Q-Chem[15]: B88-VWN-P86 and Ahlrichs TZVP basis set[83].
- Gaussian03/09[84]: DGA1-DFT with BP86 (=B88-VWN-P86) functional and Ahlrichs TZVP basis set[83].
- DMOL3[8]: DFT with PBE functional and numerical DNP basis set[85].
- Gaussian98[9] (deprecated feature[86]): DFT with B3-LYP functional and 6-31+G(d,p) basis set.
- DMOL3[8] (deprecated feature[86]): DFT with VWN-BP functional and numerical DNP basis set.

---

[83] This density functional method and basis set combination is equivalent to the Turbomole method. Thus, the COSMO*therm* parameter set optimized for the according Turbomole DFT method can be used with COSMO files produced by this quantum chemical program package.

[84] Gaussian, M. J. Frisch, *et al.*, Gaussian, Inc., Pittsburgh PA, (2011). Please note that only Gaussian09 Revision C.01 of September 2011, and later, or Gaussian03 revisions B.01 up to C.01 as released between October 2003 and March 2006 are able to produce COSMO files that can be read by COSMO*therm*.

[85] The PBE/DNP/COSMO model is available in the 2006 version of DMOL3 as implemented in Accelrys Materials Studio 4.3 as well as in all later versions of DMOL3 and Accelrys Materials Studio.

[86] Please note that it is not recommended to use COSMO files computed at this level of theory in COSMO*therm*. This calculation method and the according COSMO*therm* parameter file is supported for reasons of downward compatibility only. The method and parameter file will not be improved or updated in future versions of COSMO*therm*.

The choice of the correct DFT functional, basis set and COSMO options is explained in the user manuals of the different quantum chemistry programs. However there is a general set of element-specific descriptors that are required for the construction of the COSMO-surfaces by the quantum chemistry programs: The COSMO-radii. We recommend to use the following optimized COSMO-radii [Å] for elements[1,2,5]:

Element	r_{COSMO}
H	1.300
B	2.048
C	2.000
N	1.830
O	1.720
F	1.720
Al	2.153
Si	2.200
P	2.106
S	2.160
Cl	2.050
Zn	1.626
Ge	2.700
As	2.350
Se	2.200
Br	2.160
Sn	2.550
I	2.320
Pb	2.360

If there is no optimized COSMO-radius for a certain element, the radius parameter can be estimated reasonably from r_{vdW}, the elements van der Waals radius ($r_{COSMO} = r_{vdW} * 1.17$)[1,2,5]. Recommended values for van der Waals radii of the elements can be found in the review article of Bondi[87] and Mantina *et al.*[88]. For elements that are used as atomic ions only (i.e. no covalent bonding) the same estimate ($r_{COSMO} = r_{vdW} * 1.17$) can be used. However in this case we recommend the covalent radii r_{vdW} as provided by Sutton[89].

Atom-Ions	r_{COSMO}
Li	1.570
Na	1.800
K	2.290

---

[87] Bondi, A. *J. Phys. Chem.* **68**, 441 (1964).

[88] Mantina, M., Chamberlin, A.C., Valero, R., Cramer, C.J., and Truhlar, D.G., *J. Phys. Chem. A* **113**, 5806 (2009).

[89] Sutton, L., Ed. *Tables of Interatomic Distances andConfiguration in Molecules and Ions*, 18, Spec. Publ., London, 1965.

3 The COSMOtherm Parameter File

The COSMO*therm* Parameter files (**CTDATA**-files, discernible by the ending `.ctd`) contain all the parameters, which are required by COSMO*therm* in order to produce reliable, high quality calculations of physicochemical data. The original set of parameters is described in detail in references 1 and 3. These parameters are partly intrinsic parameters of COSMO*therm* as well as element specific parameters. In addition a few numerical thresholds are set in the CTDATA-files.

Each CTDATA-file has a header line, in which the properties of the parameterization are listed. The body of the file is not intended to be modified by the user. Therefore we do not give a detailed explanation of the numbers here.

3.1 Parameterization of COSMOtherm

Because the quality, accuracy, and systematic errors of the electrostatics resulting from the underlying quantum chemical COSMO calculations depend on the quantum chemical method (e.g. DFT-functional or SCF, MP2 or semi-empirical Hamiltonians) as well as on the basis set, COSMO*therm* needs a special parameterization for each of these method / basis set combinations. Currently (COSMO*therm* Version C3.0 Release 17.01) parameterizations for the following program packages and method / basis set combinations are available:

- **Turbomole**[7]: RI-DFT with BP (=B88-VWN-P86) functional and def-TZVP basis set.
- **Turbomole**[7]: RI-DFT with BP functional, def2-TZVPD basis set and novel fine grid cavity.
- **DMOL3**[8]: DFT with PBE functional and numerical DNP basis set.
- GAMESS[10], PQS[11], Molpro[12], Columbus[13], ORCA[14], Q-Chem[15]: B88-VWN-P86 and Ahlrichs TZVP basis set[83].
- Gaussian03/09[84]: DGA1-DFT with BP86 (=B88-VWN-P86) functional and Ahlrichs-TZVP basis set[83].
- Gaussian98[9] (obsolescent feature[86]): DFT with B3-LYP functional and 6-31+G(d,p) basis set.
- DMOL3[8](obsolescent feature[86]): DFT with VWN-BP functional and numerical DNP basis set.

All of these parameterizations are based on geometries quantum chemically optimized at the given method / basis set level. For large molecules where a full optimization of the geometry is too expensive at DFT/COSMO level there are two additional parameterization available which are based on single point energy calculations at DFT/COSMO level upon geometries optimized at semi empirical MOPAC-AM1/COSMO level:

- **Turbomole**[7]: RI-DFT with BP (=B88-VWN-P86) functional and "def-SVP" basis set
- GAMESS[10], PQS[11], Molpro[12], Columbus[13], ORCA[14], Q-Chem[15]: B88-VWN-P86 and Ahlrichs SVP basis set[83].
- Gaussian03/09[84]: DGA1-DFT with BP86 (=B88-VWN-P86) functional and Ahlrichs-SVP basis set[83].

Starting with Version C1.2, the COSMO*therm* program includes a new generic expression for the combinatorial contribution to the chemical potential (see section 1.1). The new parameterizations of COSMO*therm* that use the new generic combinatorial contribution replace all older parameterizations based on the old generic combinatorial term or the Stavermann-Guggenheim term. In addition, also the special parameterizations for ionic species are obsolete with Version C1.2 of COSMO*therm*. Thus, there is only one COSMO*therm* parameterization for each quantum chemical level of theory and basis set. Please note that it is still possible to use any of the older parameterizations. The COSMO*therm* program is fully downward compatible to older parameterizations. Nevertheless, it is strongly recommended to use the new "`C30_1701`" parameterizations shipped with Version C3.0 Release 17.01 of the COSMO*therm* program.

Please note that starting with version C3.0 Release 17.01 of the COSMO*therm* program, the unit conversions and fundamental physical constants used in the COSMO*therm* parameterizations (Avogadro constant N_A, Boltzmann constant k, Elementary charge e, Electron mass m_e, Planck constant h, and Electric constant e^0) were updated according to the latest NIST/CODATA recommendations[90]. In older parameterization files, the original values of the physical constants and unit conversions have been retained, to ensure downward compatibility of the predictions with these parameter sets.

3.2 Parameterization Usage

All COSMO*therm* parameter files (CTDATA-files `name.ctd`) are identified by the quantum chemical calculation method they were parameterized for (plus eventually the basis set of the quantum chemical calculation). Additionally, they are denoted by the program version and release number of the COSMO*therm* version they are shipped with. Thus any CTDATA-file is identified `METHOD_BASIS_VERSION_RELEASE.ctd`. For example, the CTDATA-file for Turbomole COSMO files with BP functional and Ahlrichs-SVP basis set shipped with COSMO*therm* Version C3.0 Release 17.01, is denoted `BP_SVP_C30_1701.ctd`. The parameterizations based on single point DFT calculations upon semiempirically optimized geometries are denoted by the additional shorthand index `_AM1` (e.g. `BP_SVP_AM1_C30_1701.ctd`). Note, that it is still possible to use the older (pre-Version-C1.1) COSMO*therm* parameterization file format (files which are named `crsdata_*`). However, the old parameterizations will not be updated or enhanced in future releases.

Parameterizations shipped with COSMO*therm* Version C3.0 Release 17.01:

- **`BP_TZVP_C30_1701.ctd`**: Use with quantum chemical COSMO calculations (**Turbomole**[7], GAMESS[10], PQS[11], Molpro[12], Columbus[13], ORCA[14], or Q-Chem[15] (RI-)DFT with BP-functional and def-TZVP basis set or Gaussian03/09[84] DGA1-DFT with BP86 functional and Ahlrichs-TZVP basis set) based on fully optimized geometries.
- **`BP_SVP_AM1_C30_1701.ctd`**: Use with quantum chemical single point COSMO calculations (**Turbomole**[7] (RI-)DFT with BP-functional and def-SVP basis set or Gaussian03/09[84] DGA1-DFT with BP86 functional and Ahlrichs-SVP basis set) based on geometries optimized by MOPAC/AM1/COSMO.
- **`BP_TZVPD_FINE_C30_1701.ctd`**: Quantum chemical level "**BP-TZVPD-FINE**" parameter set with a novel Hydrogen Bond interaction term ("HB2012" term[91]) and a novel van der Waals dispersion term based on the "D3" method of Grimme *et al.*[92]. This parameter set is considered to be the best quality set currently offered by COSMO*logic*. Use it with quantum chemical COSMO calculations based on

---

[90] Mohr, P.J.; Taylor, B.N.; Newell, D.B. *CODATA Recommended Values of the Fundamental Physical Constants: 2010*, National Institute of Standards and Technology, Gaithersburg, Maryland 20899-8420, USA. Web: `physics.nist.gov/constants`.

[91] Reinisch, J.; Klamt, A.; Eckert, F. *A Comprehensive Description of the Current State of COSMO-RS and its Hydrogen Bond Expression,* in preparation for *J. Comp. Chem.* (2015).

[92] Grimme, S.; Antony, J.; Ehrlich, S.; Krieg, H. *J. Chem. Phys.*, **132**, 154104 (2010).

Turbomole[7] BP86 functional and def2-TZVPD basis set single point calculations with the novel fine grid marching tetrahedron cavity (**FINE**[91]) COSMO, based upon geometries optimized with BP86 functional and TZVP basis set.

- **`DMOL3_PBE_C30_1701.ctd`**: Use with quantum chemical COSMO calculations (DMOL3[8] PBE functional, DNP basis set) on fully optimized geometries.

The COSMO*therm* program is fully downward compatible to older parameterizations. Nevertheless, it is strongly recommended to use only the new parameterizations shipped with Version C3.0 Release 17.01 of the COSMO*therm* program. Moreover, please note that the use of older parameterization versions may lead to a reduced functionality and/or prediction quality in COSMO*therm*. This is the case for all `G98_*` parameterization files as well as for pre-2008 DMOL3 parameterization files (i.e. parameter file "`DMOL3_C21_0107.ctd`" and previous ones): These files were parameterized on old and inferior COSMO cavity versions as implemented in old program versions of Gaussian (Gaussian98) and DMOL3 (DMOL3 in Accelrys Materials Studio 4.1 and previous; in addition, a different DFT functional, VWN-BP instead of PBE was used there), respectively, and they are kept for downward compatibility reasons only. The `G98_*` and old `DMOL3_*` parameterization files are not updated or enhanced any more and thus do not support the full current COSMO*therm* functionality.

3.3 Recommended Parameterizations

The choice of the appropriate quantum chemistry method and basis set level (and thus also the choice of the appropriate parameter set of COSMO*therm*) generally depends upon the required quality and the later application of the predictions. For a given problem setting the optimal results and fastest computation times are achieved if an appropriate combination of quantum chemistry method and COSMO*therm* parameterization is used, as recommended below. Please note that parameterizations for quantum chemical methods other than the recommended ones (e.g. B3-LYP or DMOL3 VWN-BP parameterizations) will still be shipped with this and future COSMO*therm* releases. I.e. it is still possible to use the COSMO files computed at these levels of quantum chemical theory within COSMO*therm*. However these parameterizations will not be enhanced or updated for future releases of COSMO*therm*. In addition, COSMO*base*[80] the database of COSMO files will not be shipped for such quantum chemical levels.

There are three main areas of application which require different proceeding regarding quantum chemistry:

3.3.1 High quality prediction of thermophysical data for chemical engineering

The application of COSMO*therm* in chemical and engineering thermodynamics (e.g. prediction of binary VLE or LLE data, activity coefficients in solution or vapor pressures) typically requires high quality of the predictions of the properties of mixtures of small to medium sized molecules (up to 25 non-Hydrogen atoms). The quantum chemical method of choice for such a problem is a full **Turbomole BP-RI-DFT** COSMO optimization of the molecules geometry using the large **TZVP** basis set[7]. Alternatively, COSMO files created by the PQS[11], Molpro[12] or ORCA[14] program (at the same level of theory and basis set) can be used with the same COSMO*therm* parameter set. Starting with version **Gaussian03**[84] it is also possible to use the Gaussian program package **DGA1-DFT** with **BP86** (=B88-VWN-P86) functional and Ahlrichs-**TZVP** basis set. Although the charge surfaces of COSMO files produced by Gaussian and Turbomole are not identical, the charge surface of Gaussian COSMO files can be converted into a charge surface that is equivalent to the charge surface produced by Turbomole using a molecule of the same geometry. COSMO*therm* automatically does this conversion if COSMO files produced by Gaussian are used. Thus the same COSMO*therm* parameter set can be used for COSMO files of the two program packages. If the vapor pressure of the compound also has to be predicted by COSMO*therm*, a full optimization of the molecular geometry in the gas phase (i.e. without the COSMO option) is also strongly recommended in order to obtain the gas phase energy of the molecule which then can be utilized by COSMO*therm* to obtain a reasonable prediction of the compounds vapor pressure. Based on such quantum chemical COSMO (and gas phase) calculations, the best quality of the COSMO*therm* prediction can be achieved with the `BP_TZVP_C30_1701.ctd` parameterization. A similar quality can be reached with the **DMOL3**[8] program package using full COSMO and gas phase geometry optimization with the PBE DFT functional and the numerical DNP basis set on the quantum chemistry level and the `DMOL3_PBE_C30_1701.ctd` parameterization in COSMO*therm*.

Recommended for **high quality predictions** of thermophysical data for chemical engineering purposes:

Program Package	DFT functional	Basis Set	COSMO*therm* Parameterization
Turbomole[93]	BP (RI-DFT)	**TZVP**	BP_TZVP_C30_1701.ctd
DMOL3[8]	PBE	DNP	DMOL3_PBE_C30_1701.ctd

In 2012 a new quantum chemical calulation level **BP-TZVPD-FINE** was introduced to COSMO*therm*[91] and TURBOMOLE[94]. This method is based on a Turbomole BP-RI-DFT COSMO single point calculation with TZVPD basis set on top of an optimized BP//TZVP/COSMO geometry. The COSMO single point calculation is using the large TZVPD basis set with additional diffuse basis functions and a novel type of molecular surface cavity construction (fine grid marching tetrahedron cavity, **FINE**[91]), which creates a COSMO surface whose segments are more uniform and evenly distributed compared to the standard COSMO cavity. The gas phase energy files of this level are optimized on Turbomole BP-RI-DFT level with TZVP basis set, followed by a single point BP-RI-DFT calculation with larger TZVPD basis set. The associated `BP_TZVPD_FINE_C30_1701.ctd` parameter set additionally incorporates a novel hydrogen bonding term (HB2012[91]) and a novel van der Waals dispersion term based on the "D3" method of Grimme *et al.*[92], which shows improved thermodynamic property prediction results for compound classes where the classical COSMO-RS hydrogen bonding term showed weaknesses, e.g. with secondary and tertiary aliphatic amines and polyether compounds. In addition, the wider realm of organic liquid and gas phase thermodynamics is predicted with the same quality or slightly better than the standard BP-TZVP-COSMO and DMOL3_PBE methods. Hence the **BP-TZVPD-FINE** level in combination with the `BP_TZVPD_FINE_C30_1701.ctd` parameterization is considered to be "best quality" calculation method that we currently offer. Note that the BP-TZVPD-FINE level is subject to ongoing revision and improvement. Thus, the actual predictions of the FINE level may differ more strongly from parameterization to parameterization, than the other, standard parameter sets. Moreover, FINE level is computationally somewhat more demanding than the other standard methods, which concerns the quantum chemistry calculation as well as the COSMO*therm* calculation itself. Nevertheless, all COSMO*therm* users should feel encouraged to use this level: it probably fits their special demands in property prediction better than the recommended standard levels BP-TZVP-COSMO and DMOL3-PBE, in particular if the mentioned compound classes are involved.

Recommended for **best quality predictions** of thermophysical data for chemical engineering purposes:

Program Package	DFT functional	Basis Set	COSMO*therm* Parameterization
Turbomole	BP (RI-DFT/**FINE**)	**TZVPD**	BP_TZVPD_FINE_C30_1701.ctd

---

[93] As an alternative to Turbomole, the COSMO files for this level of theory and basis set can be calculated with the Gaussian03[84], Gaussian09[84], PQS[11], Molpro[12], GAMESS-US10, Q-Chem[15], Columbus[13], or ORCA[14] quantum chemistry programs.

[94] The **BP-TZVPD-FINE** methodology is available in TURBOMOLE release 6.4 (Spring 2012) and later versions.

3.3.2 High-Throughput Screening

The application of COSMO*therm* for the purpose of screening a large number of compounds (e.g. prediction of solubility of compound in various solvents or prediction of solvent partition coefficients like $\log P_{\text{Octanol-Water}}$ for a large number of solutes as often are demanded in life science applications such as agent/drug design) typically requires a predictive quality that is somewhat lower than for typical chemical engineering applications. However, the molecules involved, are often larger (>100 atoms) and an overall large number of compounds has to be computed by quantum chemistry. Thus a compromise between computational demands of quantum chemistry and quality of the COSMO*therm* predictions has to be made: A very good compromise is the optimization of molecular geometry on the computationally very cheap semiempirical **MOPAC AM1-COSMO** level[18] with a subsequent single point COSMO calculation on **Turbomole BP-RI-DFT** or **Gaussian03 DGA1-BP86-DFT** COSMO level using the small **SVP** basis set[7]. Alternatively, COSMO files created by the PQS[11], Molpro[12] or ORCA[14] program (at the same level of theory and basis set) can be used with the same COSMO*therm* parameter set. Based on such quantum chemical COSMO calculations, the best quality of the COSMO*therm* prediction can be achieved with the `BP_SVP_AM1_C30_1701.ctd` parameterization. This parameterizations also includes the optimized QSPR parameters that can be utilized by COSMO*therm* to predict solids solubilities in arbitrary solvents.

Recommended for the screening of a large number of compounds using COSMO*therm*:

Program Package	Quantum Chemical Method	COSMO*therm* Parameterization
MOPAC//Turbomole[93]	**AM1 (opt) // BP (RI-DFT) / SVP (single)**	BP_SVP_AM1_C30_1701.ctd

3.3.3 Ionic Species

The treatment of ionic species does not require any special COSMO*therm* parameterization. The standard parameterizations can be used. Although it should be noted that a high quality quantum chemistry method in combination with a large basis set is required to capture the strong polarity of the ionic species. The quantum chemical method of choice for such a problem is our "best quality" calculation method **BP-TZVPD-FINE** level[94] in combination with the `BP_TZVPD_FINE_C30_1701.ctd` parameterization.

Recommended for the computation of ionic species:

Program Package	DFT functional	Basis Set	COSMO*therm* Parameterization
Turbomole[94]	**BP (RI-DFT/FINE)**	**TZVPD**	BP_TZVPD_FINE_C30_1701.ctd

The COSMOtherm Output File

The structure of the COSMO*therm* output file `filename.out` closely resembles that of the input file as described in section 2. After some header lines that include the job identifier as well as the date/time-stamp of the COSMO*therm* run, the calculated information for the pure compounds are listed. Example 4-1 shows the ''pure compound'' output for propanone:

`Example 4-1:`

```
Compound Information for molecule  1 (COSMO file propanone.cosmo)
Atomic weights         :  1111111111
E_COSMO+dE             :   -121263.1257 kcal/mol
E_gas                  :   - 21256.8251 kcal/mol
E_COSMO-E_gas+dE       :        -6.4292 kcal/mol
E_diel                 :        -8.2210 kcal/mol
Averaging corr dE      :         0.3015 kcal/mol
EvdW in continuum      :        -3.5091 kcal/mol
Area                   :       102.6328 A^2
Volume                 :        86.2969 A^3
Molecular Weight       :        58.0798 a.m.u.
Total COSMO charge     :         0.0008 a.u.
Dipole moment (t,x,y,z):         4.2408    0.0032   -0.0024    4.2408 Debye
Sigma moments (1-6)    :         0.0000   47.9057   36.0270   66.2727   83.0209  126.2395
H-bond moments (accept):         0.0564    5.7119    2.7906    0.9717
H-bond moments (donor) :         0.0000    0.0000    0.0000    0.0000
```

First, the atomic weights are given (default values of one in this example), then the energy of the solvated molecule (`E_COSMO`) plus the averaged correction for the dielectric energy (`dE`), and the gas phase energy (`E_gas`) as read from the input file or estimated by COSMO*therm* are given. The following lines contain the net energy difference between the ideally screened state and the gas phase (`E_COSMO-E_gas+dE`), the total dielectric energy (`E_diel`; Eq. (7) of reference 6), the averaged correction for the dielectric energy (`dE`; Eq. (15) in reference 3) and the van der Waals energy of this molecule in a continuum (`EvdW`). Please note, that this `EvdW` is a purely hypothetical free energy and in contrast to the enthalpy of mixture that is given in the mixture output section (`H_vdW` see below) the `EvdW` in continuum is not observable or available experimentally. The next lines give the total area of the surface segments as computed by the quantum chemical COSMO calculation and (if provided in the `.cosmo` file) the volume enclosed by this area. Below, the molecular weight and the total screening charge of this molecule are given, followed by the molecules dipole moment (the four numbers are first, the total sum of the dipole moment and then the x-, y- and z-components of the dipole moment vector, all given in [Debye]). The last lines of the ''pure compound''-output contain the σ-moments M_l^X for this molecule (cf. Section 5.5) as well as the corresponding sigma moments resulting from the ability of the molecule to be an acceptor or donor of hydrogen bond. The value of `0.0564` for propanone in the above example indicates a slight hydrogen bonding acceptor capacity, as can be expected for a molecule with a carbonyl group.

The third part of the output file contains the information about the mixed compounds. For each mixture ratio and temperatures given in the input file, the information for all compounds in the mixture and optionally (if the compound consists of several conformers and the `wconf`-command is given) for all of the individual molecules in the mix. Example 4-2 shows the results for propanone in a mixture with methylene chloride at 273.0 K:

Example 4-2:

```
 Results for mixture   1
 -----------------------
 Temperature          :    273.000 K
 Compound Nr.         :          1         2
 Compound             :  propanone    ch2cl2
 Mole Fraction        :      1.000     0.000

 Compound:  1  (propanone)
 Chemical potential of the compound in the mixture :         -0.89620 kcal/mol
 Log10(partial pressure) [mbar]                    :          2.08578
 Free energy of molecule in mix (E_COSMO+dE+Mu)    :    -121263.72041 kcal/mol
 Total mean interaction energy in the mix (H_int)  :         -4.74018 kcal/mol
 Misfit interaction energy in the mix (H_MF)       :          2.47774 kcal/mol
 H-Bond interaction energy in the mix (H_HB)       :          0.00000 kcal/mol
 VdW interaction energy in the mix (H_vdW)         :         -7.21792 kcal/mol
 Ring correction                                   :          0.00000 kcal/mol
```

The first line gives the chemical potential of the compound in the mixture i.e. the converged value of μ_i of Eq. (1.6). Below, the common logarithm of the fugacity (partial vapor pressure/x_i) of the compound is given. Next is the total free energy of the compound G_i (i.e. `E_COSMO+dE` of the pure compound plus the chemical potential in the mix μ_i). The closing lines contain the mean interaction enthalpy of the compound with its surrounding (`H_int`), i.e. the interaction enthalpy of the compound, which can be used to derive heats of mixing and heats of vaporization. Finally, this interaction enthalpy is separated into different contributions arising from electrostatic misfit (`H_MF`), hydrogen bonding (`H_HB`), van der Waals interactions (`H_vdW`) and a <contribution resulting from rings in molecules (`Ring correction`) (compare section 1.1). Please note that if conformers are used, `H_int` is not the sum of the misfit, hydrogen bond and van der Waals enthalpy contributions, because it contains an additional energy contribution resulting from conformational excitation (`H_conf`), which is not written to the output file.

If the automatic calculation of binary or ternary mixtures is toggled (`binary` or `ternary` keywords), additional output is produced. The initial two/three mixtures produced by the `binary` or `ternary` options are mixtures consisting of only one of the concerned compounds. The output for these mixtures does not differ from the usual as described above. However, the outputs for the following mixtures differ. For each compound *i* two more output lines are given: The activity coefficient γ_i and the partial pressure of this compound in the gas phase p^V_i in [mbar]. In addition, for each mixture the excess enthalpy H^E [kcal/mol] and excess free energy G^E [kcal/mol] as well as the composition of the mixture in the gas phase (mole fractions y_i) are given. This information is also written to the file `filename.tab` in condensed, tabulated form, which then can directly be visualized by a spreadsheet program.

Example 4-3 shows the table-file of the binary system propanone-methylene chloride at 0 °C. The table-file lists the mole fractions of both compounds (`xi`), the excess enthalpy and excess free energy (`H^E, G^E`) [kcal/mol], the total vapor pressure above the mixture (`ptot`) [mbar], the partial free energies of the compounds in the mixture (`mui+RTln(xi)`)[95] [kcal/mol], the logarithm of the activity coefficient (`ln(gammai)`) and the mole fraction of the compound in the gas phase (`yi`). Note: If not stated otherwise, all energies in the COSMO*therm* output- or table-files are given in [kcal/mol] (and in [kJ/mol] if `UNIT=SI` is used). All pressures are given in [mbar] (and in [kPa]). All temperatures are in [K].

`Example 4-3:`

```
Results for binary mixture of propanone (1) + ch2cl2 (2) at T =  273.15 K - energies are in kcal/mol - temperature is in K - pressure is in mbar

       x1       x2           H^E           G^E            ptot  mu1+RTln(x1)  mu2+RTln(x2)    ln(gamma1)    ln(gamma2)         y1         y2
  0.00000  1.00000   -0.00000002   -0.00000002      164.178922  -11.68211827   -2.92362731   -1.48397517   -0.00000002   0.000000   1.000000
  0.00001  0.99999   -0.00001739   -0.00000808      164.177329   -7.93194468   -2.92363274   -1.48394630   -0.00000004   0.000000   1.000000
  0.00100  0.99900   -0.00173772   -0.00080556      164.019027   -5.43028662   -2.92417195   -1.48108609   -0.00000276   0.000032   0.999968
  0.01000  0.99000   -0.01723917   -0.00798484      162.540353   -4.16615949   -2.92916929   -1.45516100   -0.00015797   0.000336   0.999664
  0.02000  0.98000   -0.03416940   -0.01581217      160.815493   -3.77430768   -2.93492503   -1.42652086   -0.00060761   0.000718   0.999282
  0.05000  0.95000   -0.08307495   -0.03834306      155.153272   -3.23081039   -2.95349855   -1.34169474   -0.00372928   0.002206   0.997794
  0.10000  0.90000   -0.15810329   -0.07268951      144.297902   -2.77975720   -2.98896557   -1.20400624   -0.01499196   0.006248   0.993752
  0.15000  0.85000   -0.22478782   -0.10297503      132.148097   -2.48800367   -3.03029611   -1.07206410   -0.03396402   0.013324   0.986676
  0.20000  0.80000   -0.28277002   -0.12914633      119.323528   -2.26350429   -3.07777919   -0.94622045   -0.06080277   0.025305   0.974695
  0.25000  0.75000   -0.33177236   -0.15116381      106.500807   -2.07766787   -3.13169212   -0.82705508   -0.09557137   0.044978   0.955022
  0.30000  0.70000   -0.37161778   -0.16900260       94.368163   -1.91802739   -3.19226333   -0.71532039   -0.13815009   0.076165   0.923835
  0.35000  0.65000   -0.40211320   -0.18265251       83.514917   -1.77771621   -3.25988257   -0.61101921   -0.18859611   0.123697   0.876303
  0.40000  0.60000   -0.42319354   -0.19211964       74.464937   -1.65296948   -3.33484059   -0.51476831   -0.24662537   0.192206   0.807794
  0.45000  0.55000   -0.43485015   -0.19742545       67.583632   -1.54129906   -3.41754078   -0.42685550   -0.31194696   0.284046   0.715954
  0.50000  0.50000   -0.43714741   -0.19860659       63.069955   -1.44100143   -3.50850619   -0.34746872   -0.38419420   0.396388   0.603612
  0.55000  0.45000   -0.43021576   -0.19571400       60.952100   -1.35083314   -3.60845700   -0.27668976   -0.46294213   0.519786   0.480214
  0.60000  0.40000   -0.41426377   -0.18881166       61.099584   -1.26984189   -3.71841420   -0.21451602   -0.54769920   0.640571   0.359429
  0.65000  0.35000   -0.38954269   -0.17797522       63.247717   -1.19724482   -3.83990819   -0.16083562   -0.63795859   0.746361   0.253639
  0.70000  0.30000   -0.35635613   -0.16329027       67.031779   -1.13237899   -3.97529398   -0.11546136   -0.73318724   0.830441   0.169559
  0.75000  0.25000   -0.31504527   -0.14485076       72.024711   -1.07465991   -4.12837868   -0.07813623   -0.83284625   0.892261   0.107739
  0.80000  0.20000   -0.26597888   -0.12275731       77.774848   -1.02355858   -4.30574112   -0.04854659   -0.93640263   0.935113   0.064887
  0.85000  0.15000   -0.20954394   -0.09711543       83.840041   -0.97858763   -4.51997583   -0.02633515   -1.04333887   0.963548   0.036452
  0.90000  0.10000   -0.14613711   -0.06803426       89.815534   -0.93929328   -4.79972053   -0.01111374   -1.15316084   0.981789   0.018211
  0.95000  0.05000   -0.07615738   -0.03562506       95.354357   -0.90525056   -5.23695929   -0.00247458   -1.26540332   0.993148   0.006852
  0.98000  0.02000   -0.03117929   -0.01462873       98.349825   -0.88717868   -5.77149875   -0.00027694   -1.33372919   0.997682   0.002318
  0.99000  0.01000   -0.01570698   -0.00737689       99.282550   -0.88152858   -6.16024457   -0.00002189   -1.35664812   0.998903   0.001097
  0.99900  0.00100   -0.00158117   -0.00074329      100.091720   -0.87659917   -7.42152638    0.00000821   -1.37733214   0.999896   0.000104
  0.99999  0.00001   -0.00001583   -0.00000744      100.178931   -0.87606584   -9.92286860    0.00000009   -1.37961058   0.999999   0.000001
  1.00000  0.00000   -0.00000002   -0.00000001      100.179810   -0.87606047  -13.67303899    0.00000000   -1.37963358   1.000000   0.000000
```

---

[95] Note, that the chemical potential μ_i as given by COSMO*therm* is the pseudo chemical potential as defined by Ben-Naim[17]. In order to obtain the partial free energy (i.e. the experimentally available chemical potential), an entropic term **RTln(x_i)** has to be added to the COSMO*therm* μ_i (see section 1.1). This entropic correction to μ_i is done in the table-file, whereas the COSMO*therm* output file always contains the uncorrected μ_i.

3.4 Error Codes

If COSMO*therm* discovers an unrecoverable error in the input file, or during the course of the COSMO*therm* run, it will write a text message to the bottom of the COSMO*therm* output file and stop execution. In addition an error code `ierror` will be sent to the command line shell window, or - if the graphical user interface COSMO*thermX* is used - a separate message window with the error code will pop up. The following error codes can be given by COSMO*therm*:

COSMO*therm* Error Codes:

```
error=  1 :  ERROR: Input file missing ! Usage: cosmotherm file[.inp]
error=  2 :  ERROR: COSMOTHERM input file not found !
error=  3 :  ERROR: COSMOTHERM input file is damaged or empty !
error=  4 :  ERROR: COSMOtherm parameter file not found !
error=  5 :  ERROR: COSMOtherm license key not found !
error=  6 :  ERROR: COSMOtherm database file not found !
error=  7 :  ERROR: Maximum number of $DATABASE entries exceeded !
error=  8 :  ERROR: Parameter file version is newer than program version !
             ERROR: Please use appropriate parameter file !
error=  9 :  ERROR: Parameter file has a wrong format !
             ERROR: Please use appropriate parameter file !
error= 10 :  ERROR: Compound not found in database index file !
error= 11 :  ERROR: Missing argument for given compound input !
error= 12 :  ERROR: Could not find .cosmo, .cos, .ccf or .mcos file for given compound input !
error= 13 :  ERROR: Cannot read file.
             ERROR: Unknown/Illegal COSMO file format !
error= 14 :  ERROR: No AUTOC conformers found within the number range requested by USEC command !
             ERROR: Exiting with FILE NOT FOUND error (Error 15) !
error= 15 :  ERROR: Could not find file!
error= 16 :  ERROR: Problem occured in reading of COSMO file !
error= 17 :  ERROR: Cannot write uncompressed COSMO-file for compound ! Molecule is too large !
error= 18 :  ERROR: Problems in creating VRML-file for ISOCAV surface of this molecule !
             ERROR: Please check your cosmo-file !
error= 20 :  ERROR: Illegal file format of compressed cosmo (ccf) file!
error= 21 :  ERROR: Cannot use compressed COSMO-file with used COSMOtherm parameter set !
             ERROR: Please use COSMOtherm parameter set for Turbomole COSMO-file:
                    COSMOtherm Revision C12_0702 or later !
error= 22 :  ERROR: CCF files are limited to less atoms !
error= 23 :  ERROR: This option can not be used with secure ccf-file !
error= 25 :  ERROR: COSMO-Metafile empty or damaged !
error= 26 :  ERROR: Metafile damaged. COSMO filename missing !
error= 27 :  ERROR: Could not find compound input file !
             ERROR: This file is required by metafile !
error= 28 :  ERROR: COSMO-metafile is not charge neutral !
error= 30 :  ERROR: The DBAS-RN=/DBN= input can not be combined with the explicit conformer block "[]"
                    input !
error= 31 :  ERROR: More than one conformer block identifier "[" was found !
error= 32 :  ERROR: Conformer block close identifier "]" was  found, but the block was never opened !
error= 35 :  ERROR: QM computation level found in a molecule differs from QM level of the parameter set !
             ERROR: Stopping COSMOtherm execution. You may override this error message with the global
                    AWPS option !
error= 36 :  ERROR: The QM method in the QSPR-file does not match with the parameter set used !
             ERROR: Stopping COSMOtherm execution. You may override this error message with the global
                    AWPS option !
```

COSMO*therm* Error Codes (continued):

```
error= 40 :  ERROR: Illegal character found in the input of atomic weights !
             ERROR: Only blank spaces are allowed !
error= 41 :  ERROR: Illegal character found in the input of sigma moment coefficients !
             ERROR: Only blank spaces are allowed !
error= 42 :  ERROR: Atomic Weigths are all zero for one molecule !
error= 43 :  ERROR: No atoms found for a molecule !
error= 44 :  ERROR: Unknown element detected !
error= 45 :  ERROR: Problem in calculation of molecular volume !
error= 46 :  ERROR: No segments found in COSMO file !
error= 47 :  ERROR: Maximum number of segments exceeded !
error= 48 :  ERROR: Extreme sigma=charge/area values found in a molecule !
             ERROR: COSMO-file of this molecule may be damaged !
             ERROR: Cannot run COSMOtherm calculation with this molecule !
error= 49 :  ERROR: Something went wrong in cavity construction for vdw calculation !
error= 50 :  ERROR: Unreasonable gas phase energy read !
error= 51 :  ERROR: Unreasonable COSMO or gasphase energy for a conformer of one compound was detected !
             ERROR: Please check your COSMO-files !
error= 52 :  ERROR: Unreasonable COSMO energy for a conformer of one compound was detected !
             ERROR: Please check your COSMO-files !
error= 55 :  ERROR: Total number of processed compounds is too large !
error= 57 :  ERROR: Out of memory: Boundarys exceeded !
error= 58 :  ERROR: Memory allocation fault !
error= 59 :  ERROR: Memory deallocation fault !
error= 60 :  ERROR: The computation of analytic gradients is not possible if conformers are used !
error= 61 :  ERROR: The computation of analytic gradients is not possible for COSMOtherm parameter set !
error= 62 :  ERROR: Could not reach self-consistency in chemical potential calculation !
error= 67 :  ERROR: Multiple compound input errors found !
error= 70 :  ERROR: Too many mixture options in a single mixture line !
error= 71 :  ERROR: The mixture is not charge neutral !
             ERROR: It is possible to override this error break with the IGNORE_CHARGE keyword !
error= 72 :  ERROR: All concentrations are zero !
error= 75 :  ERROR: Invalid temperature input !
error= 76 :  ERROR: Ambiguous temperature input !
error= 77 :  ERROR: Missing argument for temperature input !
error= 78 :  ERROR: Temperature is less or equal to zero Kelvin !
error= 79 :  ERROR: Temperature T is unreasonable for a COSMOtherm computation !
error= 80 :  ERROR: No concentrations or mole fractions found !
error= 81 :  ERROR: Missing argument for pure compound X input !
error= 82 :  ERROR: Could not find compound  given in pure compound computation !
error= 83 :  ERROR: Illegal compound number given in pure compound computation !
error= 84 :  ERROR: wrong argument for X_PURE input. X_PURE=argument needs to be a compound number
                    or MICELLE (if COSMOmic is used) !
error= 85 :  ERROR: Concentrations and pure compound input can not be mixed !
error= 86 :  ERROR: Concentrations and mole fractions can not be mixed !
error= 87 :  ERROR: Illegal concentration input !
```

COSMO*therm* Error Codes (continued):

```
error= 90 :  ERROR: Invalid COSMOtherm license !
error= 91 :  ERROR: COSMOtherm license expired !
error= 93 :  ERROR: COSMOtherm license key is empty !
error= 94 :  ERROR: COSMOtherm license key is damaged !
error= 95 :  ERROR: COSMOtherm license does not allow this type of calculation !
error= 96 :  ERROR: The COSMOtherm license does not include COSMObase ! The given COSMObase cosmo-file
                    can not be used !
error= 97 :  ERROR: COSMOtherm license does not include COSMOmic !
error= 98 :  ERROR: COSMOtherm license: Illegal compound file in RESTRICTED operating mode
                    (Education/Demo) !
error=100 :  ERROR: No LFER parameters found for pKa computation !
             ERROR: PKA option will be ignored !
error=101 :  ERROR: The charge difference between molecules A and B used in pKa computation is not one !
             ERROR: PKA computation not possible !
error=102 :  ERROR: The pKa LFER parameters in the parameter file were determined for a specific solvent !
             ERROR: These parameter values are not valid for solvents or solvent mixtures other than
                    the given one !
             ERROR: PKA calculation not possible !
error=105 :  ERROR: At this temperature, the vapor pressure can not be extrapolated from the
                    WAGNER coefficients !
             ERROR: Please use a different method for the  estimation of the pure compound vapor pressure !
error=110 :  ERROR: Illegal input in contact interaction computation !
error=111 :  ERROR: Illegal input in contact interaction computation !
             ERROR: The compound for contacts is required to have a finite concentration in the
                    given mixture !
error=120 :  ERROR: Concentration input XM/CM missing or damaged in BINARY/TERNARY $MIX input !
error=121 :  ERROR: BINARY / TERNARY computation not possible! $MIX components are overlapping !
error=122 :  ERROR: BINARY/TERNARY computation not possible ! $MIX and $IL options can not be mixed !
error=123 :  ERROR: Automatic computation of the thermodynamics of a BINARY/TERNARY/MULTINARY system
                    not possible ! Number of processed compounds is too small or too large !
error=124 :  ERROR: MULTINARY concentration vector missing !
error=125 :  ERROR: IL input not consistent BINARY VLE/LLE computation not possible !
error=126 :  ERROR: BINARY computation not possible ! $IL components overlapping in SLE computation !
error=130 :  ERROR: No parameters found for automatic solubility computation of solids !
             ERROR: SOLUB option will be ignored !
error=131 :  ERROR: Automatic solubility computation via QSPR requires compound water in the compound
                    input section !
             ERROR: Please add water in the compound input section !
             ERROR: SOLUB option will be ignored !
error=132 :  ERROR: Illegal reference solubility value log(x_Ref) !
error=133 :  ERROR: Ambiguous input of mass based solubility definitions !
error=135 :  ERROR: Missing SOLVENT for gas solubility computation !
error=136 :  ERROR: Could not find compound required for automatic gas solubility computation !
error=137 :  ERROR: Illegal compound number found for automatic gas solubility computation !
error=138 :  ERROR: Illegal solvent concentration found for automatic gas solubility computation !
error=139 :  ERROR: Missing pressure for gas solubility computation !
error=140 :  ERROR: Invalid pressure argument in gas solubility computation !
error=141 :  ERROR: Illegal pressure  found for automatic gas solubility computation !
```

COSMO*therm* Error Codes (continued):

```
error=145 :  ERROR: Missing QSPR parameters for density calculation !
error=146 :  ERROR: Missing QSPR parameters for viscosity calculation !
error=150 :  ERROR: Missing starting concentration in LIQ_EX computation !
error=151 :  ERROR: Illegal starting concentration  in LIQ_EX computation !
error=152 :  ERROR: Illegal solute concentration  in LIQ_EX computation !
error=153 :  ERROR: In LIQ_EX computation phase numbers for GASEOUS phase and SOLID phase cannot
                    be the same !
error=154 :  ERROR: Too many compounds in LIQ_EX computation !
error=160 :  ERROR: COSMOmic micelle file could not be read ! ERROR: Could not find COSMOmic file !
error=161 :  ERROR: COSMOmic micelle file could not be read ! ERROR: Could not open COSMOmic file !
error=162 :  ERROR: COSMOmic missing argument in micelle file !
error=163 :  ERROR: COSMOmic micelle file corrupted !
error=164 :  ERROR: COSMOmic concentrations of all compounds are zero in at least one layer !
error=165 :  ERROR: COSMOmic memory problem: real weights not possible!
error=166 :  ERROR: COSMOmic micelle file could not be read !
             ERROR: COSMomic number of layers is too large !
```

A more detailed description of the actual error can be found at the bottom of the COSMO*therm* output file of the crashed COSMO*therm* job.

4 Advanced Features of COSMOtherm

4.1 σ-Profiles

In COSMO-RS theory (which is the basis of COSMO*therm*) all molecular interactions consist of local pair wise interactions of segments of molecular COSMO-surfaces. Basically quantum chemical COSMO calculations provide a discrete surface around a molecule embedded in a virtual conductor. Of this surface each segment *i* is characterized by its area a_i and the screening charge density (SCD) σ_i on this segment which takes into account the electrostatic screening of the solute molecule by its surrounding (which in a virtual conductor is perfect screening) and the back-polarization of the solute molecule. In addition, the total energy of the ideally screened molecule E_{COSMO} is provided. Within COSMO-RS theory a liquid is now considered an ensemble of closely packed ideally screened molecules. Thermodynamic properties of compounds are won from the statistical averaging in the ensemble of interacting surface pieces. To describe the composition of the surface segment ensemble with respect to the interactions (which depend on σ only), only the probability distribution of σ has to be known for all compounds X_i. Such probability distributions $p^X(\sigma)$ are called "σ-profiles". The σ-profile of the whole system/mixture $p_S(\sigma)$ is just a sum of the σ-profiles of the components X_i weighted with their mole fraction in the mixture x_i. σ-profiles give a detailed and vivid description of molecules polarity properties. Some examples for the usage and interpretation of σ-profiles are given in section 5.2. A comprehensive overview over the properties and usage of σ-profiles is given in references 1 and 2.

4.2 σ-Potentials

The chemical potential $\mu_S(\sigma)$ of a surface segment with screening charge density (SCD) σ in an ensemble described by normalized distribution function $p_S(\sigma)$ is a measure for the affinity of the system *S* to a surface of polarity σ. It is a characteristic function of each system and is called "σ-potential". A comprehensive overview over the properties and usage of σ-potentials is given in references 1 and 2.

The COSMO-RS representations of molecular interactions namely the σ-profiles and σ-potentials of compounds and mixtures, respectively, contain valuable information - qualitatively as well as quantitatively. Figures 5.2-1 and 5.2-2 show the σ-profiles and the room temperature σ-potentials of the four solvents water, acetone, chloroform and hexane, respectively. Of these, hexane is the least polar compound. This is reflected in the narrow distribution of the charge densities around zero in Figure 5.2-1. The two peaks can be assigned to the carbon atoms for positive σ and to the hydrogen atoms for negative σ values (please keep in mind that negative partial charges of atoms cause positive screening charge densities and vice versa). The corresponding σ-potential, which is a measure for the affinity of the solvent to a molecular surface of polarity σ, is a simple parabola centered at $\sigma=0$ (see Figure 5.2-2). Such a shape arises from misfit contributions only (no hydrogen bonding) and is equivalent to purely dielectric behavior. The other extreme is represented by the σ-profile of water: It is very broad and the probability for σ is almost zero at the center of the σ-profile. The broad peak around –0.015 $e/Å^2$ arises from the two very polar hydrogen atoms whereas the peak around +0.015 $e/Å^2$ results from the lone pairs of the oxygen. This reflects the excellent ability of water to act as a donor as well as an acceptor for hydrogen bonding. In addition such a symmetric shape of the σ-profile indicates a favorable electrostatic interaction of water with itself, explaining its high boiling point and surface tension. The corresponding σ-potential has a much higher value around zero reflecting an unfavorable interaction with nonpolar surface. This is reflected the much stronger hydrophilicity of water in comparison to hexane. The shape of the outer regions of the σ-potential is due to hydrogen bonding: If a hydrogen bond donor in another compound has a SCD that is greater than 0.01 $e/Å^2$ or if a hydrogen bond acceptor has a SCD that is below -0.01 $e/Å^2$, it can build hydrogen bonds with water. The σ-profile of acetone is not symmetric. The peak at +0.012 *e/Å²* resulting from the carbonyl oxygen indicates hydrogen bonding acceptor capacity. However, unlike water there is no corresponding peak in the hydrogen bonding donor area. Therefore the interaction of acetone with itself is very unfavorable, explaining its relatively low boiling point and surface tension. This is also reflected in the σ-potential: While on the positive side it shows almost parabolic behavior (no hydrogen bonding donor capacity), on the negative side it quickly becomes strongly negative. Compared to water, the hydrogen bonding acceptor capacity of acetone is stronger, which is reflected in the smaller σ-values at which the σ-potential becomes negative. The σ-profile of chloroform shows three peaks in the region around zero that derive from the chlorine atoms. The peak at –0.013 *e/Å²* correspond to an acidic hydrogen atom. But due to quite small area of this peak, no significant hydrogen bonding donor capacity can be expected from this hydrogen atom. This is clearly visible from the σ-potential, which is not getting negative in the region of large positive σ values. As for the acetone, the asymmetric shape of the σ-profile indicates an unfavorable interaction of chloroform with itself, again resulting in a relatively low boiling point. It should be noted, that the σ-profiles of acetone and chloroform are almost complementary in the region of misfit interactions (i.e. for σ values between –0.008 and +0.008 *e/Å²*). This means that they should mix quite favorably. This is in fact the case as can be seen from the strongly negative excess enthalpy of acetone-chloroform mixtures (see also section 3.1). To sum up, one can say that σ-profiles and σ-potentials can be used to qualitatively interpret the interactions in a compound or a mixture, e.g. to assert a certain solvent or co-solvent which has a certain effect on the activities in a solution or mixture.

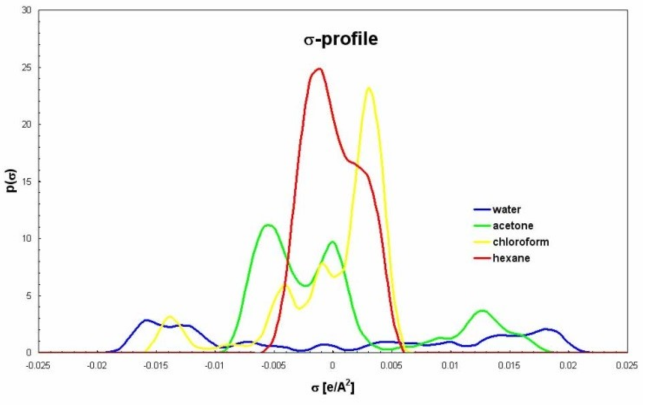


Figure 5.2-1: σ-profile plot.

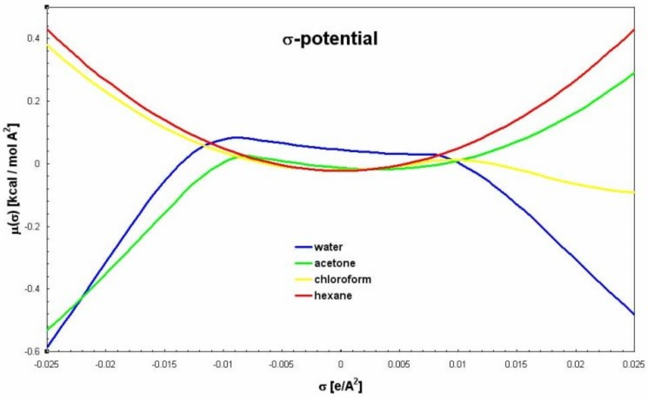


Figure 5.2-2: σ-potential plot.

4.3 Atomic Weights and COSMO-Metafiles

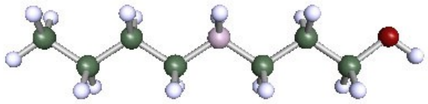
In most of its typical applications, COSMO*therm* is intended to treat complete molecules as one entity. In such a case, obviously all atoms of the compound have to be taken into account in the calculation of the sigma profile. This corresponds to a weight of 1 for all atoms.

However, if we consider very large molecules like polymer chains, it may be impossible to do a single quantum chemical calculation for the entire molecule. Instead, it is useful to compose the large molecule out of molecular fragments which are calculated independently, but which have to be sufficiently saturated by e.g. one or more monomeric units at all sites where the fragment is cut. Hence, the COSMO file of such a compound will contain atoms of the saturation region, which should not be taken into account in the COSMO*therm* calculation. This can be achieved by setting the weight of these atoms to zero, while the weight of the atoms of the fragment itself are kept one.

Another possible application of the explicit modification of the atomic weight factors is the case of large symmetric molecules: In order to speed up the quantum chemistry calculation, the symmetry-equivalent parts of the molecule might be replaced by a suitable smaller saturation fragment (e.g. in a metal-(ligand)$_4$ complex, replace three of the large ligands by smaller saturation groups such as hydrogens). The original, untruncated molecule then can be re-established in the COSMO*therm* calculation by weighting the atoms of the symmetry-equivalent part of the molecule with the number of symmetry-equivalent groups (i.e. four in the above example), the unique atoms with one and the atoms of the saturation groups with zero. Note that, due to the basic principle of COSMO*therm* (i.e. ensemble averaging over molecular surface segments, see section 1.1), such a representation of symmetry-equivalent parts of molecules is no approximation in the framework of the COSMO*therm* theory. However, if the saturation group is not chosen properly, the replacement of symmetry-equivalent groups might introduce errors in the quantum chemical calculation, thus also introducing errors into COSMO*therm*.

In some cases, it even can be useful to virtually elongate an alkane chain in a molecule by weighting one middle CH_2-group by an appropriate replication factor, in order to mimic a molecule with a longer chain in COSMO*therm*. For example, it is not necessary to calculate an octadecane explicitly. One could as well take a decane and weight one of the middle CH_2 groups by a factor 9 in order to get the correct number of CH_2-groups. Such an approximation normally will not have any significant effect on the outcomes of the COSMO*therm* calculation. For example, 1-nonanol can be constructed from a 1-octanol COSMO file, by double weighting of a central CH_2 group:

```
f=1-pentanol.cosmo  w={112111111122111111}
```



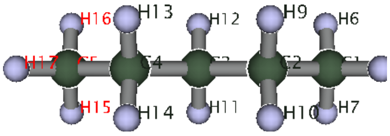
Obviously, if you can afford the calculation for the entire molecule it is best to use that and work without explicit atomic weights, in order to avoid any artefacts, which may arise if the saturation is not realistic. By default, all weight strings are assumed to be one. This means, if in the weight string less weight numbers are given than there are atoms in the molecule, the weights for the remaining atoms are assumed to be one. This default can be changed by the command **`wdfl=value`**. If given, either in the same compound input line as the weight string, or within the meta-file (see below), the `value` given, will override the default weight number of 1.

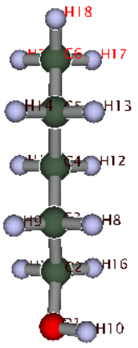
Technical note: In the first version of COSMO*therm* the atomic weighting was applied to the atoms prior to the ensemble averaging process that produces the σ-values used in COSMO*therm*. In some situations this lead to small problems. We now (starting with COSMO*therm* Version C1.0) apply the weighting after the averaging, which avoids these errors. In addition, we now apply a round-off of the total screening charge for each molecule. This ensures that the effect of an eventually non-perfect neutrality of fragments is further reduced.

The so-called **COSMO-metafiles** denoted `name.mcos` are a conceptual extension of the atomic weigthing approach: Using COSMO-metafiles it is possible to construct large molecules via molecular fragments taken from different `.COSMO files`. The syntax of the metafiles is the same as the syntax of the compound input lines in the COSMO*therm* input files. Each line of the metafiles represents one molecular fragment which is described by two necessary input commands: the COSMO file filename given by the `f=name.cosmo` card and the atomic weights input given by the `w={iii}` card for integer weights, or the `rw={`r_1 r_2 r_3`}` card for real number weights. Alternatively, for the `iw1={`n_1 n_2 n_3 …`}` option all weights are zero by default and only the atoms with the numbers n_1 n_2 n_3 explicitly given with this option are set to one. The input otions, `aw={`n_1:w_1 n_2:w_2 …`}`, `awu={`n_1:w_1 n_2:w_2 …`}`, and `awz={`n_1:w_1 n_2:w_2 …`}` provide a more flexible alternative to the `iw1` option, in that real number weights w_i can be given for atom with number n_i. For option `aw`, the default weight of the atoms not explicitly given as argument of the keyword, are all one, or w_0, as given by `wdfl=`w_0 keyword. For option `awu`, the default weight is one, and for option `awz`, the default weight is zero. Thus the `iw1` and the `aw`, `awu`, and `awz` options allow for a simple cutout of a small area (e.g. an active site) in a large molecule (e.g. a protein).

Example 5.3-1 shows a COSMO-metafile for 1-octanol constructed from 1-pentanol and n-pentane having zeroed out the terminal CH_3 groups.

<u>`Example 5.3-1:`</u>





```
f = 1-pentanol.cosmo w = {111110011111111100}
f = pentane.cosmo    w = {11110111111111000}
```

Another possibility of constructing the same molecule would be the use of the 1-heptanol COSMO file as a basis file and weighting one of the middle CH_2 groups doubly.

Some general precautions have to be taken if molecules shall be constructed via metafiles:

First, the quality of the COSMO*therm* results critically depends on a chemically reasonable choice of the fragments. For example, functional groups should not be cut in between. In addition, COSMO*therm* results are vastly improved if complete groups (the CH_3 groups in the above example) and not only single atoms (for example single H-atoms) are weighted or zeroed out. Currently at COSMO*logic,* a program (COSMO*quick*/COSMO*frag*)[96] is available, which is able to automatically construct metafiles that are reasonable in this respect. The COSMO-metafile generation in the COSMO*quick*/COSMO*frag* program is based upon a very large database of molecular fragments (CFDB).

---

[96] Loschen, C.; Klamt A., COSMO*quick*, COSMO*logic* GmbH & Co. KG (2014).

Second, the weights provided in the COSMO-metafiles depend on the sequence of atoms in the underlying COSMO files. I.e. a COSMO-metafile is not necessarily transferable between different sources of COSMO files (for example if a molecule was optimized using different starting geometries at different levels of quantum-chemical theory, the sequence of atoms in the resulting COSMO files might differ. However, in COSMO*base*[80], the database of COSMO files provided by COSMO*logic*, the metafiles are fully transferable between the COSMO files of all the quantum chemical methods. In addition, it should be noted that the molecular volume of molecules constructed via COSMO-metafiles or atomic weights can only be determined approximately. However, the volume is usually approximated within <10 % deviation from the exact COSMO volume. Note that, the quality of the approximated volume also slightly depends on the choice of the molecular fragment, i.e. an insensible choice of the fragments will also lead to a bad volume estimate.

Third, if complex compounds are built up from several molecular fragments, or, if large polymer compounds are represented by a monomeric repeat unit represented by an atom-weighted cut-out of a monomer COSMO-file, there may occur a certain amount of charge mismatch of the COSMO-charges at the cutting edges of the molecular fragments. If a COSMO-metafile is built up from a large number of fragments, or, if a single cut-out repeat unit is weighted up by a large atom weight factor, the (usually small) partial cutting charges of the fragment(s) may add up to a considerable overall fault "cutting" charge of the meta-compound as a whole. Such faulty cutting charges will be reneutralized by COSMO*therm* to guarantee the overall exact neutrality of the given meta-compound. By default, all meta-compounds (i.e. COSMO-metafiles and atom weighted COSMO-files) will be reneutralized to neutrality, to a charge of q = 0 [a.u.]. This holds even if the cutting charges of the COSMO-metafile or the atom weighted COSMO-file add up to more than 0.5 a.u. Hence, even if the meta-compound would be formally a charged due to fault cutting charges, it will by default be reneutralized by COSMO*therm* to exact neutrality. This behavior can be changed by the **`cmet`**`[=`**`q`**`]` option: this options allows the use of charged meta-compounds and optionally sets it's "neutralization" charge. The `cmet[=q]` keyword can be given either in a global input line, where it will be active for all given COSMO-metafiles and atom weighted compounds (see section 2.1.3), in a compound input line, where it will be active for the actual given meta-compound only (see section 2.2.1), or, it can be given inside the COSMO-metafile, where it will be active for this specific COSMO-metafile. If the `cmet` option is given without argument, the meta-compound will be "neutralized" to the nearest integer charge of the sum of the fragment charges given in the COSMO-metafile or by the atom weights. E.g. a supposedly neutral meta-compound with cutting charges of -0.4 a.u. would still be neutralized to a neutral meta-compound of charge 0, but a meta-compound with cutting charges of 0.9 a.u. now would be neutralized to charge +1 a.u. If the `cmet=q` keyword is given with an argument **`q`**, the meta-compound will be "neutralized" to match the given charge `q`. The argument (neutralization charge `q`) is expected to be an integer charge in atomic units [a.u.]. Thus the `cmet=q` keyword can be used to force the processing of meta-compounds of a given specific charge.

Atomic weights will also be identified in any VRML file of the molecular structure or the COSMO and property surface created by COSMO*therm*. If atomic weights are set to zero, the corresponding atomic labels will be marked red in the molecular VRML file `name_mol.wrl`. If atomic weights are set to values larger than one, the labels of the corresponding atoms will be marked yellow in the molecular VRML file. Surface parts corresponding to atoms zeroed out via atomic weights will not be shown in COSMO charge or property surface VRML files `name_sig.wrl` and `name_prop.wrl`. Atomic weights larger than one are represented in the corresponding surface areas of the charge or property VRML files via a lighter color (which is the lighter the higher the atomic weight factor). The same holds also if COSMO-metafiles are used. In this case, COSMO*therm* will write individual VRML files for all of the compounds given in the COSMO metafile. Such VRML files of COSMO files fragments from COSMO metafiles are denoted `name_meta_mol.wrl` for the molecular geometry files and `name_meta_sig.wrl` or `name_meta_prop.wrl` for the molecular charge and property surface files respectively.

It is possible to use several COSMO metafiles together to form a conformer block, the same way it is done for regular COSMO files. However, putting together COSMO metafiles in a conformer block has a severe pitfall: the quantum chemical COSMO energy (unlike all the other properties that are pieced together with the metafile methodology) is not additive linearly. The quantum chemical COSMO-energy of the "meta"-compound defined by the metafile is not the sum of the COSMO energies of the fragment COSMO files! The relative weight factor of the individual conformers, however, is determined from the Boltzmann distribution of the compounds total free energy, which is the sum of the COSMO energy of the compound E_{COSMO}, its chemical potential in the mixture μ, and its dielectric energy correction *dE*. Now because the quantum chemical COSMO energy of a COSMO metafile can not be known, its value by default is set to zero, if COSMO metafiles are used, assuming that the quantum chemical COSMO energy is the same for all metafile conformers. Thus strictly speaking, the usage of COSMO-metafiles as conformers can only be done consistently if the conformers that are defined by the different metafiles would all have (nearly) the same quantum chemical COSMO energy. This is a very serious restriction that should never be forgotten, when using metafiles as conformers. Otherwise errors are introduced into the Boltzmann distribution of the metafile conformers. However there are special cases, where the quantum chemical COSMO energy of the fragments can be considered as closely related to the (unknown) quantum chemical COSMO energy of the metafile: if you have a compound that is built from several conformeric metafiles, where all of the conformer metafiles show the same fragmentation patterns and atomic weight patterns, but are built from related fragment files, where the fragment files used are different conformers of a compound. E.g. if you have a target compound that includes a fragment of 1-butanol you may create two COSMO metafiles that include `1-butanol0.cosmo` conformer from the database and one metafile that includes the `1-butanol1.cosmo` conformer from the COSMO database. Both metafiles need to have the same fragmentation pattern and it is important that the actual fragmentation does not take place in any group that is substantial for the difference between two conformations (in the case of `1-butanol0.cosmo` and `1-butanol1.cosmo` you may safely cut off the terminating CH_3 group of the conformer, it is essentially the same in the conformers, but not the hydroxy group: it is exactly in the hydroxy group where these conformers differ). Given these conditions it is safe to assume that the value of the quantum chemical COSMO energy of the metafile can be approximated by the sum of fragment molecule energies, because in the Boltzmann conformer equilibrium we are looking at the energies only in a relative way, i.e. only the difference between the conformer energies is accounted for. The usage of the sum of the fragment COSMO-energies instead of the default zero in determination of the the total free energy of the metafile conformers is toggled by the **`mcse`** (Metafile Conformer Sum of Energy) command, which can be used as a global command or as local command in the compound input lines of the COSMO*therm* input file. Please be aware that the usage of the `mcse` command only makes sense if all the conditions given above are met. If this is not the case, then the COSMO energies of the fragments that are summed up will not be consistent and thus may lead to severe errors in the Boltzmann distribution of the conformers.

4.3.1 Difference Charges

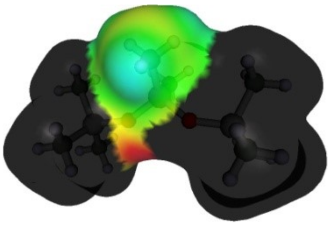
In addition to atomic weights, it is also possible to give a number of atomic "**difference charges**" in the compound lines of a COSMO-metafile or the COSMO*therm* input file via command **`del={q1 q2 … }`**. The difference charges `qi` must be given in the same sequence as the atoms in the corresponding COSMO file of the compound. The difference charges can be used to do an atom-wise correction of the charges in the COSMO file as they were computed by quantum chemistry. Please note that in the COSMO*therm* version C21_0110 and older COSMO*therm* versions, the numbers given with the `del` option were interpreted as surface charges $\sigma = q/a$ not as charges q. This has been corrected in COSMO*therm* version C30_1201 and following versions. Now by default the numbers given in the `del` option are interpreted as charges q. If you want the numbers to be interpreted as surface charges σ, this can ba achieved with the additional keyword **`delqa`**. The "*a posteriori*" correction of atomic charges can be useful mainly in two cases. First, if the surface charges of the given compound can not be calculated properly by the quantum chemical COSMO level that is used. An example for this case is compound dimethylsulfoxide (DMSO), the charges of which are not represented very well if density functional theory is used as quantum chemical method. Thus also the COSMO*therm* results are quite poor if a COSMO file is used that was computed on such a quantum chemical level (e.g. Turbomole BP, or DMOL3 PBE). Difference charges can be used to correct such defects. For example the difference charge for DMSO in sequence of atoms of the COSMO file of the COSMO*base*[80] database of COSMO files is:

```
del={0.03 -0.023 -0.023 -0.023 0.085 -0.007 0.030 -0.023 -0.023 -0.023} delqa
```

Fortunately, the number of molecules where DFT quantum chemical methods show such defects is small, the most prominent cases being DMSO, the molecular class of secondary amides and a few phosphorous containing compounds like hexamethylphosphoramide (HMPT). Another field of application of the "difference charge" concept is fragmentation of complicated molecules using COSMO-metafiles. For example, inductive and electronic effects of substituents on aromatic rings can be simulated with the help of difference charges. Thus, complicated systems of substituted heterocyclic rings might be replaced by much simpler fragments that are equipped with the according difference charges.

4.3.2 Treatment of Polymers

For large molecules like polymer chains it is currently not possible to do a single quantum chemical calculation of the entire molecule. Instead, it is useful to compose the large molecule from smaller fragments which are calculated indepently. The molecules from which the fragments are taken have to be chosen such that the fragments' direct environments correspond to the environment in the untruncated molecule. For instance, for the calculation of silicon, we use a -O-Si(CH_3)$_2$- fragment saturated by a Si(CH_3)$_3$ group and a OSi(CH_3)$_3$ group. In a COSMO*therm* calculation, a fragment of a molecule can be defined by atom weight factors: The atom weight for atoms which should be taken into account must be set to one, while the weights of the atoms which should not be taken into account must be set to zero. The chemical potential of the polymer is calculated from a monomer (or larger) repeat unit, and hence the corresponding solvent area and volume do not reflect the real surface area and volume of the polymer. It is therefore recommended to switch off the combinatorial contribution to the chemical potential with the `combi` command in a global command line of the COSMO*therm* input file. COSMO*therm* allows for the calculation of solubilities of gaseous and liquid compounds in polymers if the polymer is treated as a liquid solvent of monomeric (or larger) repeat units. This appoximation is only valid for non-crystalline polymers. Furthermore, polymer swelling cannot be taken into account, because the employed relations are valid only for low solute load.



Computation of *relative* solubility in a polymer:

The solubility of a gaseous compound in a polymer can be calculated from its partial vapor pressure and activity coefficient:

$$p_i = p_i^0 x_i \gamma_i \text{ and } x_i = \frac{p_i}{p_i^0} \cdot \frac{1}{\gamma_i} \tag{5.3.2-1}$$

For a liquid compound, the mole fraction of the solute in the polymer can be approximated by the inverse activity coefficient:

$$x_i = \frac{1}{\gamma_i} \tag{5.3.2-2}$$

Thus, the polymer solubilities of gaseous and liquid compounds are obtained from an activity coefficient (cf. section 2.3.2) or Henry law coefficient ($k_H = p_i^0 \gamma_i$, cf. section 2.3.3) calculation rather than a solubility calculation. In addition, the iterative procedure for the calculation of solubility should not be used with polymers, since this procedure treats the polymer like a liquid. Instead, the activity coefficient at infinite dilution can be used as a reasonable approximation for polymers at low solute load. I.e. the solute's solubility in the polymer is approximated from an infinite dilution activity or Henry law coefficient of the solute compound in a given monomeric repeat unit of the polymer. Obviously, this approximative procedure does not allow for the calculation of absolute or quantitative solubility values of gaseous or liquid solutes in the polymer. But the solubility thus computed can provide qualitative or "relative" information about the solubility of different solutes in the polymer, e.g. in terms of a relative solubility ranking of the solutes, or in terms of the relative selectivity S_{ij}^{rel} (the ratio of two relative solubilities or absorption coefficients) between two solutes. For example, the selectivity of two liquid solutes can be calculated as

$$S_{ij}^{rel} = x_i / x_j = \gamma_j / \gamma_i. \tag{5.3.2-3}$$

In the case of gaseous solutes, the partial pressures of the solutes have to be taken into account and the selectivity is defined as

$$S_{ij}^{rel} = x_i / x_j = p_j^0 \gamma_j / p_i^0 \gamma_i. \tag{5.3.2-4}$$

The "relative" polymer solubility calculation method thus sketched typically is not sufficient to provide absolute or quantitative predictions of the solubility in polymers. However, with sufficient experimental data available, a regression analysis can be done outside COSMO*therm*, (e.g. a empirical regression of computed $1/\gamma_i$ with experimental x_i data), which then can be used to predict quasi-absolute solubilities for a given polymer solvent.

4.3.3 Computation of *absolute* solubility in a polymer:

In addition to the relative (or, with the help of external data regression: semi-quantitative) prediction polymer solubilities presented above, COSMO*therm* also offers an absolute and quantitative prediction method for polymer properties. The absolute prediction of systems that contain polymers involves two modifications of the COSMO-RS theory. First, because in COSMO*therm* the polymer is represented by a small molecular fragment (e.g. by a monomeric repeat unit of the polymer chain), the absolute prediction of the polymer properties needs to take into account the apparent molecular weight of the polymer $MW_{polymer}$. Second, the combinatorial contribution to the chemical potential, which was derived from molecular size and shape considerations (see section 1.1) has to be replaced by a special combinatorial term that accounts for free volume effects and thus can handle macromolecules. Of the numerous free volume terms that were published in the chemical engineering literature it was found[97] that the free volume method of Elbro *et al.*[98] is particular useful in combination with COSMO-RS. If dealing with macromolecules in an absolute and quantitative way, it is therefore recommended to switch on the free volume combinatorial contribution of Elbro *et al.*[98] using the `combi=ELBRO` command, which either can be given locally, in a temperature/mixture line, or generally, in a global command line of the COSMO*therm* input file. Unlike the default combinatorial contribution of COSMO*therm*, which accounts for shape and size effects of molecules in terms of their molecular hard-core (i.e. COSMO cavity) volumes and areas (see section 1.1), the free volume method of Elbro *et al.* also allows for the handling of compounds that do not have a well-defined surface area or volume, in particular macromolecules such as polymers. The combinatorial contribution to the excess free energy of the Elbro free volume (FV) term closely resembles the Flory-Huggins combinatorial contribution:

$$G_i^{C,FV} = RT \sum_i x_i \ln\left(\frac{\varphi_i^{FV}}{x_i}\right) \tag{5.3.2-5}$$

Wherein φ_i^{FV} is the free volume fraction of compound *i* with molecule fraction x_i.

---

[97] Loschen, C.; Klamt, A. *Ind. Eng. Chem. Res.* **53**, 11478 (2014).

[98] Elbro, H. S.; Fredenslund, A.; Rasmussen, P. A. *Macromolecules* **23**, 4707 (1990).

$$\varphi_i^{FV} = \frac{x_i\left(\widetilde{V}_i - r_i\right)}{\sum_j x_j\left(\widetilde{V}_j - r_i\right)}$$

(5.3.2-6)

The free volume $V_i^F = \tilde{V}_i - \mathrm{r}_i$ is defined as the difference between the molar volume $\tilde{V}_i$ (as derived from the compound or polymer density) and the molecular hard-core volume[99] r_i (which can be approximated very well by the molecule's COSMO-volume [97]). Using the free volume fraction φ_i^{FV} thus defined, the free volume contribution to the chemical potential of compound *i* reads:

$$\mu_i^{FV} = RT\left[\ln\left(\frac{\varphi_i^{FV}}{x_i}\right) + 1 - \frac{\varphi_i^{FV}}{x_i}\right] \qquad (5.3.2\text{-}7)$$

Please note that the value of x_i used in equations 5.3.2-5 to 5.3.2-7 is the mole fraction with respect to the polymer chain (not the molecular fragment/repeat unit). This implies that in a COSMO*therm* calculation that utilizes the **`combi=ELBRO`** term and the input of the polymer's molecular weight $MW_{polymer}$, all mole fractions x_i are converted to the polymer framework with the help of the given $MW_{polymer}$ value. Hence in practice, the absolute prediction of solubilities and related properties in polymer solvents requires the input or assessment of the molecular weight of the polymer $MW_{polymer}$ as well as the input or estimation of the free volumes V_i^F of all compounds involved, including the "molecular" species such as the solutes or other non-macromolecular solvents.

The input of all polymer-related properties is done in the compound input section, i.e. the polymer definitions and properties can be given either in the compound input line of the COSMO*therm* input file, or in the vapor pressure/property `name.vap` file of the compound (see section 2.2.3 Vapor Pressure / Property Input). A compound can be flagged as polymer using the **`ispolymer`** term. Please note that the `ispolymer` flag does not affect the results of the COSMO*therm* calculation. It simply is used as an identifier, which allows COSMO*therm* to refer to the given compound as a polymer in the output and in potential warning or error messages. The apparent molecular weight of a polymer, $MW_{polymer}$ can be given with the **`expmw`**`=value` option. The argument $MW_{polymer}$ will be used by COSMO*therm* to scale up the molecular weight of the given molecular fragment to match the actual/apparent polymer weight using the atomic weights concept explained in this section, directly above. The concept of real number weights (similar to the `rw` option that can be used in a compound input line and in COSMO-metafiles) is used to "blow up" the molecular fragment or monomeric repeat unit that is given in the COSMO-file or COSMO-metafile in order to match the input $MW_{polymer}$ value. Please note that this scaling procedure implies that the caveats described for the construction of the molecular fragments in COSMO-metafiles above also hold if the `expmw` option is used. If the scaling procedure results in a large charge mismatch and thus in a nonzero "neutralization charge" (as described in section 5.3, above) COSMO*therm* will produce a warning message. This implies that for a typical polymer case, where a large macromolecule is scaled up from a monomeric (and thus very small) repeat unit in a COSMO-metafile via the `expmw` command, the charge neutrality warning may occur quite regularly if no special care is taken in the preparation of the molecular

---

[99] Kouskoumvekaki, I. A.; Michelsen, M. L.; Kontogeorgis, G. M. *Fluid Phase Equilib.* **202**, 325 (2002).

fragment. However, this large charge mismatch and the resulting neutrality issues and according warning message typically are relevant only if the resulting polymer is used as a *solute* in the COSMO*therm* calculation. If the scaled up polymer is used as a *solvent* (i.e. if the predicted properties of the polymer itself are not looked at, but only the properties of other molecular compounds dissoluted in the polymer) the neutrality faults are not as relevant as in the solute case, and the neutrality warning may be ignored as a start. It should be kept in mind however that if the charge mismatches are blown out of proportion the prediction quality of the COSMO*therm* calculation may suffer even if the polymer is used as solvent only. If no `expmw` is given, COSMO*therm* will use the molecular weight of the molecular fragment (i.e. the monomeric repeat unit) as $MW_{polymer}$ value.

The free volume of a compound (polymer or molecule) can be entered in several different ways. In the compound input section of the COSMO*therm* input file or the compound vapor pressure / property file, it is possible to give the free volume V_i^F itself with the **`freevol`**`=value`, or **`freevol_SI`**`=value` options, which expect as argument the compound's free volume V_i^F in units [Å^3], and [nm^3], respectively. Alternatively the experimental molar volume $\tilde{V}_i$ can be given via the **`expmolvol`**`=value`, or **`expmolvol_SI`**`=value` keywords, which expect as argument the compound's molar volume $\tilde{V}_i$ in units [Å^3], and [nm^3], respectively. The free volume of compound *i* is computed from the input molar volume $\tilde{V}_i$ as $V_i^F = \tilde{V}_i - \mathrm{r}_i$. Moreover, it is possible to use the experimental density of compound *i* as input via keywords **`expdensity`**`=value`, **`expdensity_SI`**`=value` , or **`expdensity_Brit`**`=value`, which expect as argument the compound density in units [g/ml], [g/cm^3], and [lbm/ft^3], respectively. The experimental density ρ_i thus given is converted into the molar volume $\tilde{V}_i = MW_i/(N_A\rho_i)$, which serves to compute the free volume as $V_i^F = \tilde{V}_i - \mathrm{r}_i$. In order to be thermodynamically consistent the `combi=ELBRO` term has to be applied to all compounds in the given system. This implies that the free volume, or alternatively, the experimental molar volume or density has to be known (and greater than zero) for all compounds that are present in a given COSMO*therm* calculation.

The direct input of free volume, molar volume, or density for all of the compounds present in the COSMO*therm* calculation may not be convenient or even possible at certain situations. In that case COSMO*therm* will compute an estimate of the free volumes. This estimation method is called automatically if the `combi=ELBRO` keyword is given and free volume data is missing for a given compound. It utilizes the room-temperature QSPR density estimate for neutral compounds as described in section 2.3.10 of this manual. No additional or separate input is required for the free volume estimate. The use of the `combi=ELBRO` option is demonstrated in the example input below:

```
ctd=BP_TZVP_C30_1701.ctd cdir=../CTDATA-FILES ldir=../../licensefiles     # Global command line 1
fdir=../DATABASE-COSMO/BP-TZVP-COSMO UNIT=SI sprf                           # Global command line 2
!! Automatic activity coefficient calculation for a polymer                 # Comment line
f = pdms.mcos ispolymer ExpDensity=0.97 ExpMW=6800 comp=pdms-polymer        # Compound input: polymer
f = ethane_c0.cosmo                                                         # Compound input: solute
f = propane_c0.cosmo                                                        # Compound input: solute
f = butane_c0.cosmo                                                         # Compound input: solute
tc=25 ngamma=pdms-polymer COMBI=ELBRO         # Automatic activity coeffcient calculation in polymer solvent
```

In the example input shown, the activity coefficients of three gaseous solutes in the polymer PDMS (polydimethylsiloxane) are computed. The polymer combinatorial contribution is toggled with the `combi=ELBRO` keyword. Experimental density and molecular weight values are provided for the polymer compound only. For the solute compounds no experimental free volumes or densities are given in the input, hence they will be estimated. The computed densities ρ_i internally are converted to molar volumes $\tilde{V}_i = MW_i/(N_A\rho_i)$ and ultimately free volumes $V_i^F = \tilde{V}_i - \mathrm{r}_i$. Note, that the free volume estimates thus computed have a lower priority than any input of experimental free volumes, molar volumes, or densities in the compound input line or vapor pressure / property file. No experimental data will be overwritten.

4.4 Chemical Equilibrium and Reactions

The equilibrium of a chemical reaction in solution can be decribed in terms of the reaction free energy of the system. For an example reaction

$$\nu_A \mathrm{A} + \nu_B \mathrm{B} \rightarrow \nu_C \mathrm{C} + \nu_D \mathrm{D} \tag{5.4-1}$$

the Gibbs free energy of reaction is defined as difference of the total free energies of the product compounds and the reactant compounds, which for the example reaction 5.4-1 reads

$$\Delta G_{REACT} = \nu_C G_C + \nu_D G_D - (\nu_A G_A + \nu_B G_B) \tag{5.4-2}$$

where ν_k are the stoichiometry indices of the reacting compounds and G_k are the total Gibbs free energies of the reacting compounds in the given solution. Please note that the total Gibbs free energy G_k contains an entropic ideal mixture contribution of $RT\ln(x_k)$ if compound k is present in the mixture at a finite concentration.

In a similar way, the enthalpy of reaction is defined as difference of the liquid enthalpies of the product compounds and the reactant compounds, which for the example reaction 5.4-1 reads

$$\Delta H_{REACT} = \nu_C H_C + \nu_D H_D - (\nu_A H_A + \nu_B H_B) \tag{5.4-3}$$

where ν_k are the stoichiometry indices of the reacting compounds and H_k are the liquid enthalpies of the reacting compounds in the given solution. The equilibrium constant of the reaction can be computed from the Gibbs free energy of reaction:

$$K_{REACT} = \exp(-\Delta G_{REACT} / RT) \tag{5.4-4}$$

Using eqs. 5.4-1 to 5.4-4 it is also possible to estimate reaction barriers and kinetic constants of reactions. To do this, a pseudo reaction has to be defined wherein the reactions transition state $[\mathrm{A\text{-}B}]^{\ddagger}$ is the product compound: $\nu_A \mathrm{A} + \nu_B \mathrm{B} \rightarrow \nu_{AB} [\mathrm{A\text{-}B}]^{\ddagger}$. Thus the transition state geometry of the reacting compounds has to be optimized on quantum chemical cosmo and gas phase level and the transition state's COSMO file has to be used used as the product compound of this pseudo reaction. The reaction Gibbs free energy as computed by eq. 5.4-2 then corresponds to reaction barrier height, which via Arrhenius equation can be used to estimate the kinetic constant of the reaction.

To assess the expectable quality of a COSMO*therm* prediction of reaction equilibrium constant, reaction Gibbs free energy, and reaction enthalpy with eqs. 5.4-2 to 5.4-4, it is important to consider the quality of the contributions of which the free energies and enthalpies in COSMO*therm* are built: a compounds Gibbs free energy G_k and enthalpy H_k as computed by COSMO*therm* are formed from a COSMO*therm* contribution ("liquid" contribution) and from a quantum chemical contribution that results from the quantum chemical energy of the compound. This has practical consequences for the expectable accuracy: the accuracy of the *absolute* number of the predicted reaction energy and enthalpy mainly is determined by the accuracy of the underlying quantum chemical calculation. Because currently all COSMO*therm* parameterizations rely on a density functional theory (DFT) model, the accuracy of the reaction energy basically will be the accuracy of the DFT method, which for the DFT functionals and basis sets used may be as bad as 5 kcal/mol. Thus the quality of the prediction of *absolute* reaction energies and enthalpies will be quite low if only the DFT based quantum chemical energies are used in the reaction calculation.

If it is sufficient just to consider the *relative* reaction energies of e.g. the same reaction in different solvents (i.e. if we are looking at the "liquid" COSMO*therm* contribution only) the prediction quality is much better. When looking only at the *relative* differences between the reaction energies of a reaction in different solvents, these differences will be predicted with the usual expectable quality of COSMO*therm* for properties of liquids, which is about 0.5 kcal/mol. The large absolute number error resulting from the underlying DFT method is cancelled out in this case.

If however, high accuracy for *absolute* predictions of the reaction energy or enthalpy is required, this can be achieved by introducing "external" energy data for the compounds from high quality quantum chemistry calculations into the calculation of the compounds G_k and H_k values. This can be achieved most conveniently by separating the COSMO*therm* and the quantum chemical (QM) contributions E_k^{QM} to G_k and H_k by means of the compounds Gibbs free energy of solvation ΔG_k^{SOLV} and enthalpy of vaporization $-\Delta H_k^{vap}$ which describe the the change of the compounds total free energy and energy when going from gas phase (the reference state of the quantum chemical calculation) to the liquid phase (the state of COSMO*therm*) . Thus ΔG_k^{SOLV} and $-\Delta H_k^{vap}$ describe the change of the free energy and enthalpy, which occurs if compound is dissoluted from the gas phase into the liquid phase.

$$G_k = E_k^{QM} + \Delta G_k^{SOLV} \tag{5.4-5}$$

$$H_k = E_k^{QM} - \Delta H_k^{vap} \tag{5.4-6}$$

The ΔG_k^{SOLV} and $-\Delta H_k^{vap}$ terms describe the transition from gas to liquid state. The terms are computed by COSMO*therm* from the chemical potential difference between the gas phase μ_k^{Gas} and liquid μ_k^S (cf. equation 1.1-11 in the "Theory" section 1.1 of this manual). If available, it is also possible to use experimental data for ΔG_k^{SOLV} and ΔH_k^{vap} to compute the reaction energy and enthalpy.

Further quantitative improvement of the total free energy or enthalpy value can be achieved if the zero-point vibrational energy E_k^{ZPE} is included for each compound:

$$G_k = E_k^{QM} + E_k^{ZPE} + \Delta G_k^{SOLV} \tag{5.4-5a}$$

$$H_k = E_k^{QM} + E_k^{ZPE} - \Delta H_k^{vap} \tag{5.4-6a}$$

If the reactant or product compounds are present in the mixture at a finite concentration with a mole fraction x_k (e.g. if the reaction takes places in bulk reactant liquid) an entropic contribution $RT\ln(x_k)$ of the compound is added to the compounds free energy G_k.

By default the reaction is assumed to occur at the COSMO*therm* standard reference pressure of 1 [bar] in the ideal gas phase. Different reference pressures for the reaction can be taken into account by utilizing the pressure dependency of the free energy of solvation:

$$\Delta G_k^{SOLV}(p_{ref}) = \Delta G_k^{SOLV}(1\ \text{bar}) - RT\ln(p_{ref} / 1\ \text{bar}) \tag{5.4-7}$$

For a given temperature the computation of a chemical equilibrium is toggled by the keyword **`reaction`**. The solvent liquid wherein the reaction takes place can be defined with the **`reaction=i`** or the **`nreaction=name`**$_i$ keywords, where **`i`** is the compound number of the solvent, and **`name`**$_i$ is the compound name, respectively. Alternatively a solvent or compound mixture can be defined where the reaction takes place, via input of the mixture concentration mole fractions **x_i** with the keyword **`xr={`**x_1 x_2 …**`}`** or via input of the mixture concentration mass fractions **c_i** with the keyword **`cr={`**c_1 c_2 …**`}`**. The reactant compounds either can be given by their numbers **i_k**, as given in the sequence of the compound input, using the keyword **react={**i_1 i_2 …**}** or they can be given by their compound names **$name_k$**, using the keyword **nreact={**$name_1$ $name_2$ …**}**. The product compounds either can be given by their numbers **j_k**, as given in the sequence of the compound input, using the keyword **prod={**j_1 j_2 …**}** or they can be given by their compound names **$name_k$**, using the keyword **nprod={**$name_1$ $name_2$ …**}**. The stoichiometry numbers ν_k of the reactant and product compounds can be given with the keyword **react_n={** ν_{i1} ν_{i2} …**}** and **prod_n={** ν_{j1} ν_{j2} …**}**, respectively. If the reaction is set up this way (i.e. defining solvent concentration, reactant and product compounds, and stoichiometry) the reactions Gibbs free energy ΔG_{REACT}, enthalpy ΔH_{REACT}, and equilbrium constant K_{REACT} are computed using the quantum chemical energies as read from the compounds cosmo and gas phase energy files (i.e. DFT-quality energies), which means that E_k^{QM} as used in eqs. 5.4-5 and 5.4-6 is equivalent to the E_k^{QM}, the gas phase energy that is read from the energy-file, or – if no energy file is available – estimated from the compounds cosmo energy. A reference pressure for the reaction that is different from the default p_{ref} = 1 [bar] can be given by optional input keyword **pref=p**, where p is the reactions reference pressure in [mbar], It is also possible to give the reaction pressure in [Pa] (**pref_Pa=p** option), in [kPa] (**pref_kPa=p** option), in [psia] (**pref_psia=p** option), or in [bar] (**pref_bar=p** option).

Optionally "external" values for E_k^{QM} can be read from the input file. Thus it is possible to introduce high level quantum chemistry gas phase energies to be used in eqs. (5.4.5) and (5.4.6) to compute the compounds free energies and enthalpies and finally the reaction energy and enthalpy and equilibrium constant. The "external" quantum chemical energies of the reactant compounds can be given with the keyword **react_eqm={**E_{i1} E_{i2} …**}**, where E_{ik} are quantum chemical energies in atomic units, for the reactant compounds i_1, i_2, … . It is also possible to give the reaction energies in atomic units [Hartree] (**react_eqm={**E_{i1} E_{i2} …**}**, and **react_eqmH={**E_{i1} E_{i2} …**}** option), in [kcal/mol] (**react_eqmC={**E_{i1} E_{i2} …**}** option), in [kJ/mol] (**react_eqmJ={**E_{i1} E_{i2} …**}** option), or in [eV] (**react_eqmV={**E_{i1} E_{i2} …**}** option). Equivalently, the "external" quantum chemical energies of the product compounds can be given with the keyword **prod_eqm={**E_{j1} E_{j2} …**}**, or its variants for different energy units (**prod_eqmH={**E_{j1} E_{j2} …**}** [Hartree], **prod_eqmC={**E_{j1} E_{j2} …**}** [kcal/mol], **prod_eqmJ={**E_{j1} E_{j2} …**}** [kJ/mol], **prod_eqmV={**E_{j1} E_{j2} …**}** [eV]). Optionally the zero point vibrational energies as used in eqs. 5.4-5a and 5.4-6a can be given by the keyword **react_zpe={**EZP_{i1} EZP_{i2} …**}**, where EZP_{ik} are zero point vibrational energies in atomic units, for the reactant compounds i_1, i_2, … . It is also possible to give the zero point vibrational energies in atomic units [Hartree] (**react_zpe={**EZP_{i1} E_{i2} …**}**, and **react_zpeH={**EZP_{i1} EZP_{i2} …**}** option), in [kcal/mol] (**react_zpeC={**EZP_{i1} EZP_{i2} …**}** option), in [kJ/mol] (**react_zpeJ={**EZP_{i1} EZP_{i2} …**}** option), or in [eV] (**react_zpeV={**EZP_{i1} EZP_{i2} …**}** option). Equivalently, the zero point vibrational energies of the product compounds can be given with the keyword **prod_zpe={**EZP_{j1} EZP_{j2} …**}**, or its variants for different energy units (**prod_zpeH={**EZP_{j1} EZP_{j2} …**}** [Hartree], **prod_zpeC={**EZP_{j1} EZP_{j2} …**}** [kcal/mol], **prod_zpeJ={**EZP_{j1} EZP_{j2} …**}** [kJ/mol], **prod_zpeV={**EZP_{j1} EZP_{j2} …**}** [eV]). All of the `react_` / `prod_` options for the input of external QM or ZPE energies assume that the given energies are compound energies, which already are averaged over conformers (if there are any). Alternatively, the external QM or ZPE energies can be read individually for compounds conformers in the molecule input of the compound input section or from molecular vap-files (`EQM` and `EZP` option, see section 2.2 " Compound Input"). The energies thus read in during compound input will be used as E_k^{QM} and E_k^{ZPE} values in the reaction computation. If conformers are present, the conformer molecules of a compound will be weighted according to their external E_k^{QM} + E_k^{ZPE} energies (and not their cosmo-energies as is the default). Note that the `react_` / `prod_` input in the reaction mixture line overrides the `EQM` and `EZP` input of the compound input section. By default any `EQM` or `EZP` data read from the compound input line, the vap-file, or the energy-file is used in the reaction option. This automatic usage can be disabled locally by using the `EQM=off` and `EZP=off` options in the reaction property input line.

In addition to the "external" values for E_k^{QM} and E_k^{ZPE} it is also possible to provide "external" values for ΔG_k^{SOLV} and ΔH_k^{vap} to COSMO*therm*, which then will be used in eqs. 5.4-5 and 5.4-6 to compute the compounds free energies and enthalpies, overriding the COSMOtherm predictions for these properties. For the reactant compounds ΔG_{ik}^{SOLV} can be given with the keyword **`react_Gsol={`**G_{i1} G_{i2} …**`}`**, where G_{ik} are free energies of solvation in atomic units, for the reactant compounds i_1, i_2, … . It is possible to give ΔG_{ik}^{SOLV} in atomic units [Hartree] (**`react_Gsol={`**G_{i1} G_{i2} …**`}`**, and **`react_GsolH={`**G_{i1} G_{i2} …**`}`** option), in [kcal/mol] (**`react_GsolC={`**G_{i1} G_{i2} …**`}`** option), in [kJ/mol] (**`react_GsolJ={`**G_{i1} G_{i2} …**`}`** option), or in [eV] (**`react_GsolV={`**G_{i1} G_{i2} …**`}`** option). Equivalently, "external" ΔG_{jk}^{SOLV} for the product compounds can be given with the keyword **`prod_Gsol={`**G_{j1} G_{j2} …**`}`**, or its variants for different energy units (**`prod_GsolH={`**G_{j1} G_{j2} …**`}`** [Hartree], **`prod_GsolC={`**G_{j1} G_{j2} …**`}`** [kcal/mol], **`prod_GsolJ={`**G_{j1} G_{j2} …**`}`** [kJ/mol], **`prod_GsolV={`**G_{j1} G_{j2} …**`}`** [eV]). The reactants ΔH_{ik}^{vap} can be given with the keyword **`react_Hvap={`**H_{i1} H_{i2} …**`}`**, where H_{ik} are free energies of solvation in atomic units, for the reactant compounds i_1, i_2, … . It is possible to give ΔH_{ik}^{SOLV} in atomic units [Hartree] (**`react_Hvap`**, and **`react_HvapH`** option), in [kcal/mol] (**`react_HvapC`** option), in [kJ/mol] (**`react_HvapJ`** option), or in [eV] (**`react_HvapV`** option). Equivalently, "external" ΔH_{jk}^{vap} for the product compounds can be given with the keyword **`prod_Hvap={`**H_{j1} H_{j2} …**`}`**, or its variants for different energy units (**`prod_HvapH`** [Hartree], **`prod_HvapC`** [kcal/mol], **`prod_HvapJ`** [kJ/mol], **`prod_HvapV`** [eV]).

An example of the usage of the reaction option can be found in the input file `REACTION.inp` in the `EXAMPLE-INPUTS` directory of the COSMOtherm installation:

`Example 5.4-1: REACTION.inp`

```
ctd=BP_TZVP_C30_1701.ctd cdir=../CTDATA-FILES
fdir=../DATABASE-COSMO/BP-TZVP-COSMO wcmn efile
!! Computation of a reaction equilibrium constant in solution
f=5-methyl-4-nitrobenzofuroxane EQMH=-732.7436522 EZPC=79.68
f=7-methyl-4-nitrobenzofuroxane EQMH=-732.7493550 EZPC=79.64
f=h2o
f=propanone
f=hexane
tc=25 reaction=3 REACT={1} REACT_N={1} PROD={2} PROD_N={1}
tc=25 reaction=4 REACT={1} REACT_N={1} PROD={2} PROD_N={1}
tc=25 reaction=5 REACT={1} REACT_N={1} PROD={2} PROD_N={1}
tc=25 reaction=1 REACT={1} REACT_N={1} PROD={2} PROD_N={1}
```

In Example 5.4-1 the `EQMH`, and `EZPC` keywords are used to read in external high level quantum chemistry gas phase energies[100] and zero point vibrational energies to compute the reaction equilibrium constant K_{REACT} and reaction energy ΔG_{REACT} for the isomerization reaction (unimolecular Boulton-Katritzky rearrangement) of 5-methyl-4-nitrobenzofuroxane (compound 1, reactant ν_1=1) to 7-methyl-4-nitrobenzofuroxane (compound 2, product ν_2=1) in different solvents water, acetone and hexane, as well as in bulk reactant (which is assumed to be liquid).

The equilibrium constant `K(react)` that is computed by the `reaction` option according to equations 5.4-2 to 5.4-4 is the mole fraction based *apparent* equilibrium constant $\mathbf{K_x}$. The apparent equilibrium constant K_x varies with the chosen solvent, pressure and mixture conditions. The solvent-independent *activity*

---

[100] Eckert, F.; Rauhut, G.; Steele, P. J.; Katritzky, A. R.; *J. Am. Chem. Soc.* **121** (1999) 6700.

equilibrium constant $\mathbf{K_a}$ (`K(activity)`, also called *thermodynamic*, or *effective* equilibrium constant) is related to the apparent equilibrium constant K_x via the *nonideality factor* $\mathbf{K_\gamma}$ (`K(gamma)`, also called *activity coefficient* equilibrium constant):

$$K_a = K_x K_\gamma \tag{5.4-8}$$

The nonideality factor K_γ according to example reaction scheme 5.4-1 is defined as the stoichiometry-weighted ratio of the reactants and products activity coefficients γ_i:

$$K_\gamma = (\gamma_C^{\nu C} \gamma_D^{\nu D}) / (\gamma_A^{\nu A} \gamma_B^{\nu B}) \tag{5.4-9}$$

The computation of the activity equilibrium constant K_a is toggled by the keyword **`K_activity`**. If this keyword is given as a suboption of the `reaction` option, COSMO*therm* will compute the activity coefficents of all species in the given reaction and from these will compute K_γ and K_a, which will be written both to the COSMO*therm* table output and the output-file. In addition the activity coeffcients γ_i will be tabulated in additional column of the reaction table output.

If, for a given reaction, the activity equilibrium constant $\mathbf{K_a}$ is known experimentally, it can be passed into the `reaction` option of COSMO*therm* as a reference activity equilibrium constant $\mathbf{K_a^{Ref}}$. The solvent-independent reference equilibrium constant K_a^{Ref} then can be used to scale the solvent-dependent apparent equilibrium constant $\mathbf{K_x}$. This can be done using the option **`K_activity=`**$\mathbf{K_a^{Ref}}$. Using an experimental K_a^{Ref} the reaction system can be accounted for e.g. nonideality in the gas phase thus avoiding explicit use of fugacity coefficients (which are assumed to be unity in COSMO*therm*). If a K_a^{Ref} value is passed into the `reaction` option input via the `K_activity=`K_a^{Ref} option, the reference data K_a^{Ref} will be used to compute a scaling factor $f_a = K_a^{Ref}/K_a$, which in return is used to scale the apparent (=solvent dependent) equilibrium constant K_x.

Suboptions of the **reaction** option:

react={i_1 i_2 …} or nreact={name$_1$ name$_2$ …}	<u>Required</u> for reaction computations: Define the reactant composition from compound COSMO files. The input of the reactant composition is possible either via their compound numbers (react={i_1 i_2 …} command, where i_1, i_2 , … are the numbers of the compounds as given in the sequence of compounds in the compound input section. The arguments i_k are expected to be positive integer numbers) or via their compound name (nreact={name$_1$ name$_2$ …} command, where name$_i$, name$_j$,… are the names of the compounds as given in the compound input section).
react_n={ν_{i1} ν_{i2} …}	<u>Required</u> for reaction computations: Define the reactants stoichiometry. The input of the reactant stoichiometry factors ν_{i1}, ν_{i2} is possible via the react_n={ν_{i1} ν_{i2} …} command, where ν_{i1}, ν_{i2},… are the stoichiometry numbers of the reactant compounds defined by the compound numbers or names as given by the react={i_1 i_2 …} or nreact={name$_1$ name$_2$ …} commands. The stoichiometry factors ν_{ik} are expected to be positive integer numbers.
prod={j_1 j_2 …} or nprod={name$_1$ name$_2$ …}	<u>Required</u> for reaction computations: Define the product composition from compound COSMO files. The input of the product composition is possible either via their compound numbers (prod={j_1 j_2 …} command, where j_1, j_2 , … are the numbers of the compounds as given in the sequence of compounds in the compound input section. The arguments j_k are expected to be positive integer numbers) or via their compound name (nprod={name$_1$ name$_2$ …} command, where name$_i$, name$_j$,… are the names of the compounds as given in the compound input section).
prod_n={ν_{i1} ν_{i2} …}	<u>Required</u> for reaction computations: Define the products stoichiometry. The input of the product stoichiometry factors ν_{j1}, ν_{j2} is possible via the prod _n={ν_{j1} ν_{j2} …} command, where ν_{j1}, ν_{j2},… are the stoichiometry numbers of the reactant compounds defined by the compound numbers or names as given by the prod={j_1 j_2 …} or n prod ={name$_1$ name$_2$ …} commands. The stoichiometry factors ν_{jk} are expected to be positive integer numbers.

Suboptions of the **`reaction`** option (continued):

xr={x_1 x_2 …} or cr={c_1 c_2 …}	Optional for `reaction` computations: Give finite solvent mixture concentration at which the reaction equilibrium properties shall be computed. The input of the concentrations is possible either in mole fractions (`xr={}`) or mass fractions (`cr={}`) of the compounds of the mixture (as real numbers x_i and c_i). The arguments are expected as real numbers between zero and one in the same sequence of compounds as given in the second input area. If the values do not add up to one, COSMO*therm* will normalize them. If less mole fractions / concentrations than compounds are given, the missing ones will be assumed zero. If a negative number is given, the concentration for this compound will be inserted automatically using the normalization of the sum of mole fractions. Only one negative number is allowed per mixture input line.

`pref=p`
or
`pref_Pa=p`
or
`pref_kPa=p`
or
`pref_psia=p`
or
`pref_bar=p`
	Optional for `reaction` computations: use given reference pressure p for the computation of the reaction equilibrium. The input of the reference pressure `p` is possible via the `pref=p` keyword, where `p` is the pressure value that will be used to compute the reaction equilibrium. The pressure `p` is expected to be positive real number. For the keyword `pref` the given pressure are expected to be in [mbar], for the keyword `pref_Pa` it is expected to be in [Pa], for the keyword `pref_kPa` it is expected to be in [kPa], for the keyword `pref_psia` it is expected to be in [psia], and for the keyword `pref_bar` it is expected to be in [bar], respectively.

`K_activity`	Optional for reaction computations: compute the solvent-independent *activity* equilibrium constant K_a (`K(activity)`) and the nonideality factor K_γ (`K(gamma)`).
K_activity=K_a^{Ref}	Optional for reaction computations: input of a reference activity equilibrium constant K_a^{Ref}, for the purpose of scaling the apparent equilibrium constant K_x (`K(react)`).

Suboptions of the **reaction** option (continued):

react_eqm={E_{i1} E_{i2} …} or react_eqmH={E_{i1} E_{i2} …} or react_eqmC={E_{i1} E_{i2} …} or react_eqmJ={E_{i1} E_{i2} …} or react_eqmV={E_{i1} E_{i2} …}	Optional for reaction computations: provide external quantum chemistry energies for the reactant compounds. The input of the reactant energies E_{ik} is possible via the react_eqm={E_{i1} E_{i2} …} command, where E_{i1}, E_{i2}, … are the energies values that will be used for the reactant compounds defined by the compound numbers or names as given by the react={i_1 i_2 …} or nreact={$name_1$ $name_2$ …} commands. The energies E_{ik} are expected to be negative real numbers. For the keywords react_eqm and react_eqmH the given energies are expected to be in atomic units [Hartree], for the keyword react_eqmC energies are expected to be in [kcal/mol], for the keyword react_eqmJ energies are expected to be in [kJ/mol], and for the keyword react_eqmV energies are expected to be in [eV].
react_zpe={EZP_{i1} EZP_{i2} …} or react_zpeH={EZP_{i1} EZP_{i2}…} or react_zpeC={EZP_{i1} EZP_{i2}…} or react_zpeJ={EZP_{i1} EZP_{i2}…} or react_zpeV={EZP_{i1} EZP_{i2}…}	Optional for reaction computations: provide external zero point energies for the reactant compounds. The input of the reactant zero point energies EZP_{ik} is possible via the react_zpe={EZP_{i1} EZP_{i2} …} command, where EZP_{i1}, EZP_{i2}, … are the zero pontenergies values that will be used for the reactant compounds defined by the compound numbers or names as given by the react={i_1 i_2 …} or nreact={$name_1$ $name_2$ …} commands. The energies E_{ik} are expected to be real numbers. For the keywords react_zpe and react_zpeH the given energies are expected to be in atomic units [Hartree], for the keyword react_zpeC energies are expected to be in [kcal/mol], for the keyword react_zpeJ energies are expected to be in [kJ/mol], and for the keyword react_zpeV energies are expected to be in [eV].
eqm=off	Optional for reaction computations: disable automatic use of external QM energies read from the compound input, VAP-file, or energy-file.
ezp=off	Optional for reaction computations: disable automatic use of external zero-point vibrational energies read from the compound input, VAP-file, or energy-file.

Suboptions of the **reaction** option (continued):

prod_eqm={E_{j1} E_{j2} …} or prod_eqmH={E_{j1} E_{j2} …} or prod_eqmC={E_{j1} E_{j2} …} or prod_eqmJ={E_{j1} E_{j2} …} or prod_eqmV={E_{j1} E_{j2} …}	Optional for reaction computations: provide external quantum chemistry energies for the product compounds. The input of the product energies E_{jk} is possible via the prod_eqm={E_{j1} E_{j2} …} command, where E_{j1}, E_{j2}, … are the energies values that will be used for the reactant compounds defined by the compound numbers or names as given by the prod={j_1 j_2 …} or nprod={$name_1$ $name_2$ …} commands. The energies E_{jk} are expected to be negative real numbers. For the keywords prod_eqm and prod_eqmH the given energies are expected to be in atomic units [Hartree], for the keyword prod_eqmC energies are expected to be in [kcal/mol], for the keyword prod_eqmJ energies are expected to be in [kJ/mol], and for the keyword prod_eqmV energies are expected to be in [eV].
prod_zpe={EZP_{j1} EZP_{j2} …} or prod_zpeH={EZP_{j1} EZP_{j2} …} or prod_zpeC={EZP_{j1} EZP_{j2} …} or prod_zpeJ={EZP_{j1} EZP_{j2} …} or prod_zpeV={EZP_{j1} EZP_{j2} …}	Optional for reaction computations: provide external zero point energies for the product compounds. The input of the product zero point energies EZP_{jk} is possible via the prod_zpe={EZP_{j1} EZP_{j2} …} command, where EZP_{j1}, EZP_{j2}, … are the zero pontenergies values that will be used for the reactant compounds defined by the compound numbers or names as given by the prod={j_1 j_2 …} or nprod={$name_1$ $name_2$ …} commands. The energies E_{jk} are expected to be real numbers. For the keywords prod_zpe and prod_zpeH the given energies are expected to be in atomic units [Hartree], for the keyword prod_zpeC energies are expected to be in [kcal/mol], for the keyword prod_zpeJ energies are expected to be in [kJ/mol], and for the keyword prod_zpeV energies are expected to be in [eV].

Suboptions of the **`reaction`** option (continued):

`react_Gsol={`G_{i1} G_{i2} `…}`
or
`react_GsolH={`G_{i1} G_{i2} `…}`
or
`react_GsolC={`G_{i1} G_{i2} `…}`
or
`react_GsolJ={`G_{i1} G_{i2} `…}`
or
`react_GsolV={`G_{i1} G_{i2} `…}`
	Optional for `reaction` computations: provide external free energy of solvation energies for the reactant compounds. The input of the reactant ΔG_{ik}^{SOLV} G_{ik} is possible via the `react_Gsol={`G_{i1} G_{i2} `…}` command, where G_{i1}, G_{i2}, … are the free energy of solvation values that will be used for the reactant compounds defined by the compound numbers or names as given by the `react={`i_1 i_2 `…}` or `nreact={`$name_1$ $name_2$ `…}` commands. The arguments G_{ik} are expected to be real numbers. For the keywords `react_Gsol` and `react_GsolH` the given energies are expected to be in atomic units [Hartree], for the keyword `react_GsolC` energies are expected to be in [kcal/mol], for the keyword `react_GsolJ` energies are expected to be in [kJ/mol], and for the keyword `react_GsolV` energies are expected to be in [eV].

`react_Hvap={`H_{i1} H_{i2} `…}`
or
`react_HvapH={`H_{i1} H_{i2}`…}`
or
`react_HvapC={`H_{i1} H_{i2}`…}`
or
`react_HvapJ={`H_{i1} H_{i2}`…}`
or
`react_HvapV={`H_{i1} H_{i2}`…}`
	Optional for `reaction` computations: provide enthalpies of vaporization for the reactant compounds. The input of the reactant ΔH_{ik}^{vap} H_{ik} is possible via the `react_Hvap={`H_{i1} H_{i2} `…}` command, where H_{i1}, H_{i2}, … are the zero pontenergies values that will be used for the reactant compounds defined by the compound numbers or names as given by the `react={`i_1 i_2 `…}` or `nreact={`$name_1$ $name_2$ `…}` commands. The energies E_{ik} are expected to be real numbers. For the keywords `react_Hvap` and `react_HvapH` the given energies are expected to be in atomic units [Hartree], for the keyword `react_HvapC` energies are expected to be in [kcal/mol], for the keyword `react_HvapJ` energies are expected to be in [kJ/mol], and for the keyword `react_HvapV` energies are expected to be in [eV].

Suboptions of the **reaction** option (continued):

prod_Gsol={G_{j1} G_{j2} …} or prod_GsolH={G_{j1} G_{j2} …} or prod_GsolC={G_{j1} G_{j2} …} or prod_GsolJ={G_{j1} G_{j2} …} or prod_GsolV={G_{j1} G_{j2} …}	Optional for `reaction` computations: provide external free energy of solvation energies for the product compounds. The input of the product ΔG_{jk}^{SOLV} G_{jk} is possible via the prod_Gsol={G_{j1} G_{j2} …} command, where G_{j1}, G_{j2}, … are the free energy of solvation values that will be used for the product compounds defined by the compound numbers or names as given by the react={j_1 j_2 …} or nreact={$name_1$ $name_2$ …} commands. The arguments G_{jk} are expected to be real numbers. For the keywords `prod_Gsol` and `prod_GsolH` the given energies are expected to be in atomic units [Hartree], for the keyword `prod_GsolC` energies are expected to be in [kcal/mol], for the keyword `prod_GsolJ` energies are expected to be in [kJ/mol], and for the keyword `prod_GsolV` energies are expected to be in [eV].
prod_Hvap={H_{j1} H_{j2} …} or prod_HvapH={H_{j1} H_{j2}…} or prod_HvapC={H_{j1} H_{j2}…} or prod_HvapJ={H_{j1} H_{j2}…} or prod_HvapV={H_{j1} H_{j2}…}	Optional for `reaction` computations: provide enthalpies of vaporization for the product compounds. The input of the product ΔH_{jk}^{vap} H_{jk} is possible via the prod_Hvap={H_{j1} H_{j2} …} command, where H_{j1}, H_{j2}, … are the zero pontenergies values that will be used for the product compounds defined by the compound numbers or names as given by the react={j_1 j_2 …} or nreact={$name_1$ $name_2$ …} commands. The energies E_{jk} are expected to be real numbers. For the keywords `prod_Hvap` and `prod_HvapH` the given energies are expected to be in atomic units [Hartree], for the keyword `prod_HvapC` energies are expected to be in [kcal/mol], for the keyword `prod_HvapJ` energies are expected to be in [kJ/mol], and for the keyword `prod_HvapV` energies are expected to be in [eV].

4.5 Property Computation via σ-moment QSPR

4.5.1 σ-moments

As already noted in first COSMO-RS paper[4], the σ-potential (eq. 1.1-5) of a given liquid pure compound or mixture S can be represented by a Taylor-series with respect to σ:

$$\mu_S(\sigma)=\sum_{l}^{m} c_S^l M_l^X \quad \text{with} \quad M_l^X = \int p^X(\sigma)\sigma^l \, d\sigma \tag{5.5.1-1}$$

The c_S^l are σ-moment coefficients (SMC's) describing the liquid system S, and M_l^X are the σ-moments of solute X. The zero order σ-moment M_0^X is the molecular area of the compound or system, the first σ-moment M_1^X is the negative of the total charge of the compound or system, the second σ-moment M_2^X is correlated with screening charge of the compound/system, and the third second σ-moment M_3^X is a measure for the skewness of the sigma-profile of the compound. COSMO*therm* allows for a maximum number of $l_{max} = 6$, i.e. for a total of 7 SMC's. However, usually σ-moments of orders up to l=3 or 4 will be sufficient for a correct description of the σ-potential.

For compounds or systems that include Hydrogen Bonding donors and/or acceptors, Hydrogen Bonding moments $M_{1,HB}^X$ can be defined. In COSMO*therm* the first Hydrogen Bonding moment is defined as:

$$M_{1,HB}^X = \int p^X(\sigma) f_{1,HB}(\sigma) d\sigma \quad \text{with} \quad f_{1,HB}(\sigma)=\begin{cases} 0 & if \quad \pm\sigma \le \sigma_{HB} \\ \sigma \pm \sigma_{HB} & if \quad \pm\sigma > \sigma_{HB} \end{cases} \tag{5.5.1-2}$$

Where subscript ($_{HB}$) defines either a Hydrogen Bonding donor ($_{don}$), or an Hydrogen Bonding acceptor ($_{acc}$), and σ_{HB} is COSMO*therm* 's Hydrogen Bonding threshold (an adjustable parameter defined in equation 1.2 of section 1.1 of this manual). The higher Hydrogen Bonding moments are defined similarly, however using fixed threshold values:

$$M_{l,HB}^X = \int p^X(\sigma) f_{l,HB}(\sigma) d\sigma \quad \text{with} \quad f_{l,HB}(\sigma) = Abs\left[\begin{cases} 0 & if \pm\sigma \le (0.6+0.2l) \\ \sigma \pm (0.6+0.2l) & if \ \pm\sigma > (0.6+0.2l) \end{cases}\right] \tag{5.5.1-3}$$

Wherein l = 2, 3, 4. Note that via definitions 5.5.1-2 and 5.5.1-3, the first Hydrogen Bonding moments for HB acceptor and donors $M_{1,HB}^X$ have the sign of the Hydrogen Bonding screening charge values, while the higher Hydrogen Bonding moments $M_{l,HB}^X$ are positive absolute values.

4.5.2 σ-moments QSPR

The σ-moments M_i^X computed by COSMO*therm* can be used to compute certain molecular properties via a Quantitative Structure Property Relationship (QSPR) approach[32,101], i.e. COSMO*therms* σ-moments can be correlated with certain molecular properties such as lipophilicity, biological or environmental partition behavior like the octanol-water or soil-water partition, the partition of a compound between the blood-brain barrier or the CaCo2 cell permeability. The coefficients for a certain property can be determined from a multinlinear regression of the σ-moments with a sufficient number of experimental data. For a compound *X* a property *log(P)* is calculated via:

$$\begin{aligned}\log(P) = {} & c_1{*}M_0^X + c_2{*}M_1^X + c_3{*}M_2^X + c_4{*}M_3^X + c_5{*}M_4^X + c_6{*}M_5^X + c_7{*}M_6^X + \\ & c_8{*}M_{HBacc1}^X + c_9{*}M_{HBacc2}^X + c_{10}{*}M_{HBacc3}^X + c_{11}{*}M_{HBacc4}^X + \\ & c_{12}{*}M_{HBdon1}^X + c_{13}{*}M_{HBdon2}^X + c_{14}{*}M_{HBdon3}^X + c_{15}{*}M_{HBdon4}^X + \\ & c_{16}\end{aligned} \qquad (5.5.2\text{-}1)$$

where M_i^X is the *i* 'th σ-moment of compound *X* and M_{HBacci}^X and M_{HBdoni}^X are the *i* 'th hydrogen bonding acceptor and donor moments of compound *X*. Thus, 16 coefficients are required to do the σ-moment QSPR calculation of a molecular property. Such a calculation can be done automatically by COSMO*therm* if the coefficients c_i are given to the program. This is possible via the `QSPR=filename` or the `QSPR={c1 c2 … c18 name}` commands in the global command section of the COSMO*therm* input file or the `PROPQSPR=filename` or the `PROPQSPR={c1 c2 … c18 name}` commands in the mixture section of the COSMO*therm* input file. The `QSPR=filename` and `PROPQSPR=filename` commands attempt to read the coefficients from a file called `filename` while the `QSPR={c1 c2 … c18 name}` and `PROPQSPR={c1 c2 … c18 name}` read the coefficients directly from the input file. In the latter case, the coefficients given inside the curved brackets are expected as real number separated by blank spaces. The first 16 numbers are the QSPR coefficients $c_1 \dots c_{16}$ as used in the QSPR formula above with `c16` being a constant shift. The remaining two parameters are used only in connection with the `PWRL` option (see section 2.2) and give the minimum and maximum of the property color scale in the property VRML file. Values of c_{17}=-0.12 and c_{18}=0.03 usually are sufficient. The last (optional) entry is the name of the property to be computed. This name will be used in the output of the computed property, which by default is in the compound section of the COSMO*therm* output file. The structure of QSPR coefficient file read in with the `QSPR=filename` option is demonstrated in Example 5.5.2-1. The default search path for the QSPR-property files as given with the `QSPR=filename.prop` or `PROPQSPR=filename.prop` commands is the `/CTDATA-FILES/` directory as given by the global `cdir` command. If the QSPR-property file is not found in the `/CTDATA-FILES/` then as next step a `/PROP/` directory below the `/CTDATA-FILES/` directory will be searched for. This default can be overridden by the global `pdir` command: The `pdir=path` option given in the global commands section of the COSMO*therm* input file, allows to set a separate search path for the QSPR-property files, which then is independent from the `/CTDATA-FILES/` directory given by the `cdir` option.

---

[101] Klamt, A. and F. Eckert "COSMO-RS: A Novel Way from Quantum Chemistry to Free Energy, Solubility, and General QSAR-Descriptors for Partitioning" in *Rational Approaches to Drug Design*, Höltje, H.-D. and Sippl, W., Editors, (Prous Science, Barcelona, 2001) pages 195-205.

Example 5.5.2-1 shows the QSPR coefficient file `logPOW.prop`, which holds the QSPR coefficients from a multilinear regression of COSMO*therm* σ-moments with logarithmic octanol-water partition coefficients. The first two lines are comment lines (indicated by the hash character '#'), the following lines hold the coefficients c_1 … c_{16}, the property color ranges c_{17}, c_{18} (values of zero indicate that COSMO*therm* is setting the color range automatically) and the property name that is used to identify the QSPR property in the COSMO*therm* output file. It is possible to give several QSPR properties in one QSPR coefficient file. If such a multiproperty QSPR coefficient file is used COSMOtherm will print all given QSPR properties to the output and to the moments file(s).

For the `QSPR=filename` and the `QSPR={ c1 c2 … c18 name }` options the coefficients are assumed to be used with σ-moments in atomic units except for c_1 which is used with σ-moment M_0^X. M_0^X is equivalent to the molecular surface area and per default is given in [**Å²**]. If the QSPR coefficients have been determined for σ-moments in SI-units (i.e. M_0^X in [**nm²**] and all other M_i^X in atomic units) the `QSPR_SI=filename` and the `QSPR_SI={ c1 c2 … c18 name }` have to be used to read in the parameters.

`Example 5.5.2-1: logPOW.prop`

```
# logP(Octanol-Water) QSPR parameters for Turbomole BP-SVP-AM1 COSMO files
# Regression: 210 compounds, r^2=0.94, rmse=0.34
 0.02948233                          M_0 = area
 0                                   M_1 = -charge
-0.0432569                           M_2
-0.036411                            M_3
 0                                   M_4
 0                                   M_5
 0                                   M_6
 0                                   M_HBacc1
 0                                   M_HBacc2
 0.0578499                           M_HBacc3
 0                                   M_HBacc4
 0                                   M_HBdon1
 0                                   M_HBdon2
 0.00520617                          M_HBdon3
 0                                   M_HBdon4
-0.6681521                           Constant
 0                                   Minimum of property
 0                                   Maximum of property
log(Pow)                             Property name
```

Currently COSMO*therm* is shipped with QSPR coefficient file for the following properties:

-	`logPOW.prop`	Octanol-water partition coefficients logP$_{OW}$ for BP-SVP-AM1 COSMO.
-	`logKOC.prop`	Soil-Water partition coefficients logK$_{OC}$ for BP-SVP-AM1 COSMO[102].
-	`logKOC-BP-TZVP.prop`	Soil-Water partition coefficients logK$_{OC}$ for BP-TZVP-COSMO COSMO.
-	`logBB.prop`	Blood-Brain partition coefficients logP$_{BB}$ for BP-SVP-AM1 COSMO.
-	`logKIA.prop`	Intestinal absorption coefficients logK$_{IA}$ for BP-SVP-AM1 COSMO.
-	`logKHSA.prop`	Human Serum Albumin partition logK$_{HSA}$ for BP-SVP-AM1 COSMO.

---

[102] Klamt, A.; Eckert, F.; Diedenhofen, M. *Environmental Toxicology and Chemistry*, **21**, 2562-2566 (2002).

It is possible to give several QSPR properties in one QSPR coefficient file. If such a multiproperty QSPR coefficient file is used, COSMOtherm will print all given QSPR properties to the table, output and moments file. Currently COSMO*therm* is shipped with the the following multi-property QSPR coefficient files:

- `ADME.prop`: A collection of the three ADME properties $\log K_{IA}$, $\log P_{BB}$ and $\log K_{HSA}$ for BP-SVP-AM1 COSMO.
- `Abraham-BP-SVP-AM1.prop`: Abraham[103] linear free energy descriptors V (hydrophobicity), B (hydrogen bond acidity), S (polarity/polarizability), A (hydrogen bond acidity) and E (excess molar refraction) for BP-SVP-AM1 COSMO.
- `Abraham-BP-TZVP.prop`: Abraham[103] linear free energy descriptors V (hydrophobicity), B (hydrogen bond acidity), S (polarity/polarizability), A (hydrogen bond acidity) and E (excess molar refraction) for BP-TZVP COSMO.

The computation of the five Abraham[103] linear free energy descriptors V^X (hydrophobicity), B^X (hydrogen bond acidity), S^X (polarity/polarizability), A^X (hydrogen bond acidity) and E^X (excess molar refraction) for a compound X from the according QSPR property file, allows the additional possibility of directly calculating a thermodynamic property using the computed five descriptors and Abraham's linear free energy relationships for processes within condensed phases[103], equation 5.5.2-2:

$$\log(P_{XY}) = c_V{*}V^X + c_B{*}B^X + c_S{*}S^X + c_A{*}A^X + c_E{*}E^X + c_0 \tag{5.5.2-2}$$

Thus the solubility or partition property is defined by its five Abraham coefficients c_V, c_B, c_S, c_A, c_E and a shift parameter c_0. These six parameters can be read in by COSMOtherm from the QSPR property file if they are given with the command **ABRAHAM_COEFF={c_V c_B c_S c_A c_E c_0}** given in a comment line (i.e. following a '# ' character). The name of the property that is defined by these six coefficients can be read from the same line of the QSPR property file using the **`ABRAHAM_PROP=Name`** command. For example, in Abrahams formulation the decadic logarithm of the partition between water and wet octanol is given by eq. 5.5.2-2 using the coefficient values c_V=3.814, c_B=-3.460, c_S=-1.054, c_A=0.034, c_E=0.562 and c_0=0.088.

The input of these coefficients and the Abraham property name in a QSPR property files looks like this (example QSPR property file `BP-TZVP-Abraham-logP(Octanol-wet-Water).prop`):

```
# Turbomole BP-TZVP
#
# ABRAHAM_COEFF={3.814 -3.460 -1.054 0.034 0.562 0.088} ABRAHAM_PROP=logP(Octanol-wet-Water)
# Coefficient:   c_V    c_B    c_S    c_A    c_E    c_0
# Abraham coefficients for the computation of the decadic logarithm of the partition
# coefficient between solvents water and 1-octanol(wet)
# …
```

---

[103] Zissimos, A. M.; Abraham, M. H.; Klamt, A.: Eckert, F.: Wood, J. *J. Chem. Inf. Comput. Sci.* **42**, 1320 (2002).

When reading a QSPR property file, COSMO*therm* automatically detects if Abraham coefficients are given. If that is the case, COSMO*therm* automatically will compute the thermodynamic property defined by the given Abraham coefficients and the computed Abraham descriptors using eq. 5.5.2-2. For each compound and molecule the computed property value will be written to an additional column of the COSMO*therm* table and/or moments file and to an additional line of the COSMO*therm* output file. By default all Abraham parameter coefficients computed will be written to the table file together with the optional Abraham property as last column. If the additional keyword `noaptab` is given in the `PROPQSPR` mixture line, the computed Abraham parameter coefficients will not be printed to the table file. Instead, only the final property computed from the Abraham coefficients will be listed as results column in the `PROPQSPR` table.

Currently COSMO*therm* is shipped with a number of QSPR property files holding the QSPR coefficients for the five Abraham parameters and the definition of a thermodynamic partition property via the six Abraham coefficients, for both computational COSMO levels BP-TZVP and BP-SVP-AM1. More Abraham coefficients for a vast number of solvent or physiological partitions and related free energy properties can be found in the literature[104].

Abraham parameter QSPR property files for BP-TZVP-COSMO level:

QSPR property file (for BP-TZVP-COSMO level)	Partition coefficient
BP-TZVP-Abraham-logP(Octanol-wet-Water).prop:	wet 1-octanol – water
BP-TZVP-Abraham-logP(Octanol-dry-Water).prop:	dry 1-octanol – water
BP-TZVP-Abraham-logP(OleylAlcohol-Water).prop:	oleyl alcohol – water
BP-TZVP-Abraham-logP(Acetone-Water).prop:	acetone – water
BP-TZVP-Abraham-logP(PGDP-Water).prop:	propylene glycol dipelargonate – water
BP-TZVP-Abraham-logP(Diethylether-wet-Water).prop:	wet diethylether – water
BP-TZVP-Abraham-logP(CH2Cl2-Water).prop:	methylenechloride – water
BP-TZVP-Abraham-logP(CHCl3-Water).prop:	chloroform – water
BP-TZVP-Abraham-logP(CCl4-Water).prop:	tetrachloromethane – water
BP-TZVP-Abraham-logP(CS2-Water).prop:	carbon disulfide – water
BP-TZVP-Abraham-logP(Benzene-Water).prop:	benzene – water
BP-TZVP-Abraham-logP(Toluene-Water).prop:	toluene – water
BP-TZVP-Abraham-logP(Cyclohexane-Water).prop:	cyclohexane – water
BP-TZVP-Abraham-logP(Hexane-Water).prop:	n-hexane – water
BP-TZVP-Abraham-logP(Heptane-Water).prop:	n-heptane – water
BP-TZVP-Abraham-logP(Isooctane-Water).prop:	2,2,4-trimethylpentane – water
BP-TZVP-Abraham-logP(Hexadecane-Water).prop:	n-hexadecane – water

---

[104] M.H. Abraham, *Chem. Soc. Rev.* **22** (1993) 73; M.H. Abraham, H.S. Chadha, G.S.Whiting, R.C. Mitchell, *J. Pharm. Sci.* **83** (1994) 1085; M.H. Abraham, H.S. Chadha, J. Dixon, A.J. Leo, *J. Phys. Org. Chem.* **7** (1994) 712; M.H. Abraham, G.S. Whiting, W.J. Shuely, R.M. Doherty, *Can. J. Chem.* **76** (1998) 703;M.H. Abraham, G.S. Whiting, P.W. Carr, H. Ouyang, *J. Chem. Soc., Perkin Trans. 2* (1998) 1385; M.H. Abraham, J.A. Platts, A. Hersey, A.J. Leo, R.W. Taft, *J. Pharm. Sci.* **88** (1999) 670; M.H. Abraham, J. Andonian-Haftvan, J.P. Osei-Owusu, P. Sakellariou, J.S. Urieta, M.C. Lopez, R. Fuchs, *J. Chem. Soc., Perkin Trans. 2* (1993) 299; M.H. Abraham, F. Martins, R.C. Mitchell, C.J. Salter, *J. Pharm. Sci.* **88** (1999) 241; M.H. Abraham, J. Le, W.E. Acree Jr., *Collect. Czechoslov. Chem. Commun.* **64** (1999) 1748; M.H. Abraham, J. Le, W.E. Acree Jr., P.W. Carr, *J. Phys. Org. Chem.* **12** (1999) 675; M.H. Abraham, A.M. Zissimos, W.E. Acree Jr., *New J. Chem.* **27** (2003) 1041; M.H. Abraham, A.M. Zissimos, W.E. Acree Jr., *Phys. Chem. Chem. Phys.* **3** (2001) 3732.

Abraham parameter QSPR property files for BP-TZVP-COSMO level (continued):

QSPR property file (for BP-TZVP-COSMO level)	Partition coefficient
BP-TZVP-Abraham-logP(OliveOil-Water).prop:	olive oil – water
BP-TZVP-Abraham-logP(Gas-Water).prop:	gas – water
BP-TZVP-Abraham-logP(Cell-Water).prop:	water – cell permeation
BP-TZVP-Abraham-logP(Skin-Water).prop:	water – skin permeation at 37°C
BP-TZVP-Abraham-logP(Blood-Brain).prop:	blood –brain (logBB)
BP-TZVP-Abraham-logP(PlantCuticle-Water).prop:	water – plant cuticle
BP-TZVP-Abraham-logP(Tadpole-Narcosis).prop:	tadpole narcosis (log(1/C))
BP-TZVP-Abraham-Intestinal-Absorption.prop:	intestinal absorption (in %)
BP-TZVP-Abraham-Hsolv(Water).prop:	enthalpy of solvation in water (kj/mol)
BP-TZVP-Abraham-logS(Water).prop:	aqueous solubility log(x_S)

Abraham parameter QSPR property files for BP-SVP-AM1 level:

QSPR property file (for BP-SVP-AM1-COSMO level)	Partition coefficient
BP-SVP-AM1-Abraham-logP(Octanol-wet-Water).prop:	wet 1-octanol – water
BP-SVP-AM1-Abraham-logP(Octanol-dry-Water).prop:	dry 1-octanol – water
BP-SVP-AM1-Abraham-logP(OleylAlcohol-Water).prop:	oleyl alcohol – water
BP-SVP-AM1-Abraham-logP(Acetone-Water).prop:	acetone – water
BP-SVP-AM1-Abraham-logP(PGDP-Water).prop:	propylene glycol dipelargonate – water
BP-SVP-AM1-Abraham-logP(Diethylether-wet-Water).prop:	wet diethylether – water
BP-SVP-AM1-Abraham-logP(CH2Cl2-Water).prop:	methylenechloride – water
BP-SVP-AM1-Abraham-logP(CHCl3-Water).prop:	chloroform – water
BP-SVP-AM1-Abraham-logP(CCl4-Water).prop:	tetrachloromethane – water
BP-SVP-AM1-Abraham-logP(CS2-Water).prop:	carbon disulfide – water
BP-SVP-AM1-Abraham-logP(Benzene-Water).prop:	benzene – water
BP-SVP-AM1-Abraham-logP(Toluene-Water).prop:	toluene – water
BP-SVP-AM1-Abraham-logP(Cyclohexane-Water).prop:	cyclohexane – water
BP-SVP-AM1-Abraham-logP(Hexane-Water).prop:	n-hexane – water
BP-SVP-AM1-Abraham-logP(Heptane-Water).prop:	n-heptane – water
BP-SVP-AM1-Abraham-logP(Isooctane-Water).prop:	2,2,4-trimethylpentane – water
BP-SVP-AM1-Abraham-logP(Hexadecane-Water).prop:	n-hexadecane – water
BP-SVP-AM1-Abraham-logP(OliveOil-Water).prop:	olive oil – water
BP-SVP-AM1-Abraham-logP(Gas-Water).prop:	gas – water
BP-SVP-AM1-Abraham-logP(Cell-Water).prop:	water – cell permeation
BP-SVP-AM1-Abraham-logP(Skin-Water).prop:	water – skin permeation at 37°C
BP-SVP-AM1-Abraham-logP(Blood-Brain).prop:	blood –brain (logBB)
BP-SVP-AM1-Abraham-logP(PlantCuticle-Water).prop:	water – plant cuticle
BP-SVP-AM1-Abraham-logP(Tadpole-Narcosis).prop:	tadpole narcosis (log(1/C))
BP-SVP-AM1-Abraham-Intestinal-Absorption.prop:	intestinal absorption (in %)
BP-SVP-AM1-Abraham-Hsolv(Water).prop:	enthalpy of solvation in water (kj/mol)
BP-SVP-AM1-Abraham-logS(Water).prop:	aqueous solubility log(x_S)

The global `QSPR` option will write the computed property of each molecule to the compound output section of the COSMO*therm* output file. If the `smom[=file.mom]` option is used to create an additional tabulated file with the molecular σ-moments, the computed QSPR property will also be written to an additional column of that file. The mixture option `PROPQSPR` will write the computed property of each molecule to the compound output section of the COSMO*therm* output file and to the COSMO*therm* table file.

If no temperature and no mixture concentrations are given in the mixture input line, the `PROPQSPR` command will toggle the computation of the given QSPR property for all "molecules". Conformers are treated as individual molecules in this case. If however, a temperature and a mixture concentration are given, the `PROPQSPR` command will toggle the computation of the given QSPR property for all "compounds". This means that if several conformers are present for a given compound, the computed values of QSPR properties of the conformers will be averaged due to the Boltzmann distribution of the conformer's total free energy at the given temperature and mixture conditions. The averaged compound QSPR properties thus computed are written to the COSMO*therm* output and table files. If the keyword `smomc[=file.momc]` option is used in combination with the `PROPQSPR` mixture option, an additional tabulated file with the averaged "compound" QSPR properties and the averaged σ-moments will be written to an additional compound-σ-moments file, which is identified by the file extension `.momc`. The filename of the `.momc` file can be given as argument of the `smomc` keyword. If no argument is given, the `.momc` file will obtain the name of the current COSMO*therm* input-file.

By default the `PROPQSPR` mixture option only writes the molecule or compound QSPR properties to the COSMO*therm* table file. With keyword `pr_mom` an additional five QSPR descriptors will be printed to the COSMO*therm* table file. The five descriptors printed are the ones that have proven to be most significant for σ-moment QSPR applications. They are the molecular surface area A^X, the second and third σ-moment M_2^X, and M_3^X, as well as the third hydrogen bonding acceptor and donor moments $M_{\mathrm{HBacc}3}^X$, and $M_{\mathrm{HBdon}3}^X$. With keyword `pr_allmom` all COSMO-RS derived QSPR descriptors available will be printed to the COSMO*therm* table files. In addition to the 15 σ-moment descriptors described above some other molecule/compound properties are written to the COSMO*therm* table file if the `pr_allmom` option is used in combination with `PROPQSPR`. In agreement with the descriptors printed to the `.mom` and/or `.momc` files the extra properties are molecular Volume V^X, molecular weight MW^X, dielectric energy E_{diel}^X, the energy gain resulting from the averaging of the molecular surface charges dE_{av}^X, the molecular van der Waals energy E_{vdW}^X, the topological descriptor E_{ring}^X, and the chemical potential of the molecule/compound in the gas phase μ_{Gas}^X. Some of these supplementary descriptors may be useful for the QSPR of complicated or weakly defined thermodynamic properties, where the σ-moments alone are not sufficient for a good fit. It is important to note that two of these supplementary descriptors (E_{vdW}^X and μ_{Gas}^X) may differ considerably if they are computed as properties of "compounds" (i.e. computed in a mixture at given temperature and concentration conditions) compared to them being computed as properties of "molecules". The reason for this is that in the case of a "compound" `PROPQSPR` calculation in a given mixture, the compound's actual van der Waals energy E_{vdW}^X and chemical potential in the gas phase μ_{Gas}^X as computed by COSMO-RS thermodynamics at the given temperature/mixture conditions are used as descriptors, while in the case of a "molecular" `PROPQSPR` calculation (without given temperature and concentrations) approximate, temperature- and mixture-independent estimates are used for these two descriptors/properties.

4.5.3 σ-moment correction

As noted above, the σ-potentials of pure compounds and liquid mixtures (and other more complex, disordered matrices) can be represented by a Taylor-series with respect to σ. In the same sense we may assume that for a given solvent system *S* there is a correction function $\delta\mu_S(\sigma)$ which corrects the σ-potential of μ_S^X as calculated by COSMO*therm*, and that this function can be expressed by a Taylor series of low order.

In such situation the chemical potential of a solute *X* in solvent *S* as calculated by COSMO*therm* has to be corrected by a term

$$\delta\mu_S^X = \sum_l^m c_S^l M_l^X \tag{5.5.3-1}$$

where the c_S^l are σ-moment coefficients (SMC's) describing the specific corrections required for matrix S. The M_l^X are the σ-moments of solute *X*. The program allows for $m = l_{max} = 6$, i.e. for a total of 7 SMC's. However, usually σ-moments of orders up to l=3 or 4 will be sufficient for a correct description of chemical potentials.

Although in principle acting on solvents, SMC's are considered as compound specific in COSMO*therm*. SMC's for mixtures are handled by weighting SMC's of the compounds by their relative contribution to the total σ-profile of the mixture.

By default, the SMC's of all compounds are set to zero, unless specified otherwise. SMC's can be specified individually for each molecule in the compound/option section of the COSMO*therm* input file (see section 2.2).

In general, SMC's may be used in two situations:

- If you want to obtain a very good fit for a certain logarithmic partition coefficient, for which you have many experimental data. In this case you may first take the direct COSMO*therm* calculations of the logarithmic partition coefficients for a large set of compounds and then consider the deviations from experimental data as goal property in a linear regression analysis, which uses the σ-moments up to the desired order as descriptors. If you divide the coefficients yielded by the regression by RTln10 (i.e. at room temperature by 1.365 kcal/mol), you get the required SMCs, which you have to use as an input for one of the 2 solvents of your partition problem. Please take care about the correct sign! Leave out the first σ-moment in this case in the regression, because as long as you only have neutral compounds the first moment is zero and hence the coefficient is not defined.
- If you want to describe a solvent or a matrix which cannot be expected to be properly described by standard COSMO*therm*, either due to unknown impurities, ionization, amorphous state or for other reasons, and if you have good partition data available for that matrix, you may just proceed as in (1). An application of the SMC method to the prediction of thermal adsorption equilibria is given in ref. 105.

---

[105] Mehler, C.; Klamt, A. and Peukert, W., *AICHE Journal*. **48**, 1093 (2002).

4.6 Chemical Potential Gradients and Partial Enthalpy

4.6.1 Chemical Potential Gradients

Composition and temperature derivatives of the chemical potential or activity coefficient are important properties for process simulations[106]. COSMO*therm* allows the computation of temperature and composition derivatives of the chemical potential. The computation of the gradients is triggered by the **`grad`** option either in the global command section or the temperature/mixture section of the COSMO*therm* input file. In the first case, the temperature and composition gradients of all compounds are printed into the output file for all mixtures that are computed. In the latter case the gradients are printed only for the mixture where the `grad` command is given. Example 5.6-1 shows the output of the analytic temperature and composition gradients of propanone and water in an equimolar mixture of the two compounds at room temperature:

Example 5.6-1:

```
Compound:   1  (h2o)
 Chemical potential of the compound in the mixture :        -1.98266 kcal/mol
 Chemical potential gradient (dmu/RT(  1)/dT)      :         0.03775 1/K
 Chemical potential gradient (dmu/RT(  1)/dn(  1)) :        -0.43629
 Chemical potential gradient (dmu/RT(  1)/dn(  2)) :        -0.29694
 Chemical potential gradient (dmu/RT(  1)/dn(  3)) :         0.73323
…
 Compound:  2  (ethanol)
 Chemical potential of the compound in the mixture :        -2.68992 kcal/mol
 Chemical potential gradient (dmu/RT(  2)/dT)      :         0.04050 1/K
 Chemical potential gradient (dmu/RT(  2)/dn(  1)) :        -0.29848
 Chemical potential gradient (dmu/RT(  2)/dn(  2)) :         0.05891
 Chemical potential gradient (dmu/RT(  2)/dn(  3)) :         0.23957
…
 Compound:  3  (1-octanol)
 Chemical potential of the compound in the mixture :        -5.91792 kcal/mol
 Chemical potential gradient (dmu/RT(  3)/dT)      :         0.06926 1/K
 Chemical potential gradient (dmu/RT(  3)/dn(  1)) :         0.72901
 Chemical potential gradient (dmu/RT(  3)/dn(  2)) :         0.22867
 Chemical potential gradient (dmu/RT(  3)/dn(  3)) :        -0.95768
…
```

In addition to the COSMO*therm* output file, the gradients will be written to the table of mixture calculations that are using the `ctab` command to create a table output for the mixture calculation.

Example 5.6-2:

```
 Property  job 10 : Mixture - compound properties ;
 Settings  job 10 : T= 298.15 K ; x(1)= 0.3333 x(2)= 0.3333 x(3)= 0.3333 ;
 Units     job 10 : Concentrations in mole fractions ; Energies in kcal/mol ; Pressure in mbar ; Area in A^2 ; Volume in A^3

  Nr Compound                        mu     …     dmu/RT/dT dmu/RT/dn(1) dmu/RT/dn(2) dmu/RT/dn(3)
   1 h2o                       -1.98266     …       0.03775     -0.43629     -0.29694      0.73323
   2 ethanol                   -2.68992     …       0.04050     -0.29848      0.05891      0.23957
   3 1-octanol                 -5.91792     …       0.06926      0.72901      0.22867     -0.95768
```

For the actual utilization of the COSMO*therm* computed gradients it is important to be aware of two specific characteristics of these properties:

1) The computed gradients are on output not the derivatives of the chemical potential μ_i, but the derivatives of the <u>dimensionless</u> chemical potential μ_i ***/ RT***. This dimensionless chemical potential has the advantage of being an equivalent of a logarithmic activity coefficient $\ln(\gamma_i)$ with respect to its usage in chemical thermodynamics, it is commonly used in thermodynamics and process simulations[106].

---

[106] Taylor, R. and Kooijman, H. A., *Chem. Eng. Comm.*, **102**, 87 (1991).

2) COSMO*therm* computes the unconstrained derivatives of the dimensionless chemical potential μ_i / *RT* with regard to the temperature *T* (i.e. $(\partial(\mu_i/RT)/\partial T)_{nj}$) and composition number n_k (i.e. $(\partial(\mu_i/RT)/\partial n_k)_{T,nj}$).
    a. To obtain the constrained and normalized derivatives of activity coefficient ln(γ_i) with respect to the temperature *T*, the unconstrained temperature derivatives at composition x_i have to be subtracted from the unconstrained temperature derivatives of the pure compounds at the given temperature:

$$\left(\frac{\partial ln(\gamma_i)}{\partial T}\right)_{x_k} = \left(\frac{\partial \mu_i(T,\underline{x})/RT}{\partial T}\right)_{n_k} - \left(\frac{\partial \mu_i(T,x_i=1)/RT}{\partial T}\right)_{n_k} \qquad (5.6\text{-}1)$$

    b. To obtain the normalized derivatives of ln(γ_i) with respect to the composition x_i the unconstrained composition derivatives with respect to n_k have to be subtracted from the unconstrained composition derivative with respect to n_i:

$$\left(\frac{\partial ln(\gamma_i)}{\partial x_j}\right)_{T,x_{k\neq i}} = \left(\frac{\partial \mu_j(T,\underline{x})/RT}{\partial n_i}\right)_{T,n_j} - \sum_{k\neq i}\left(\frac{\partial \mu_j(T,\underline{x})/RT}{\partial T}\right)_{n_k} \qquad (5.6\text{-}2)$$

If the `grad` command is given globally or in a property input line, COSMO*therm* will try compute the requested gradients analytically, following the procedure outlined for the COSMOSPACE equations[107]. The analytic computation of the derivatives is possible for all classic COSMO*therm* parameterizations (i.e. all BP-TZVP, BP-SVP-AM1, and DMOL3 parameter sets) and also for most other new parameterizations. For a few parameterizations, which include the newest developments, theoretical and methodical improvements (e.g. current BP-TZVPD-FINE level parameter set `BP_TZVPD_FINE_C30_1701.ctd`) the analytic gradients may not yet be available. In this case COSMO*therm* automatically switches to numerical derivation to compute the derivatives. Finite differences in temperature *T* and mole fraction *x* are used in this case:

a. Numerical derivatives with respect to the temperature:

$$\left(\frac{\partial \mu_i/RT}{\partial T}\right)_{n_k} = \frac{\mu_i(T_2)/RT_2 - \mu_i(T_1)/RT_1}{T_2 - T_1} \qquad (5.6\text{-}1a)$$

a. Numerical derivatives with respect to the composition:

$$\left(\frac{\partial \mu_i/RT}{\partial n_j}\right)_{T,n_k} = (1-x_i)\frac{\mu_j(T,\underline{x}_2)/RT - \mu_j(T,\underline{x}_1)/RT}{x_{i,2} - x_{i,1}} \qquad (5.6\text{-}2a)$$

The numerical gradients by default are computed from central differences, i.e. two additional state points are calculated at $T_1 = T + \Delta T$ and $T_2 = T - \Delta T$, and $x_{i,1} = x_i + \Delta x$ and $x_{i,2} = x_i - \Delta x$, respectively. If the central derivation is not possible (e.g. if $x_i = 0$ or $x_i = 1$), forward differences, i.e. one additional state point is calculated at $T_2 = T + \Delta T$, and $x_{i,2} = x_i + \Delta x$, while $T_1 = T$ and $x_{i,1} = x_i$, respectively. The state point calculations of the finite differences are calculated by COSMO*therm* on the fly. They will not be printed to the COSMO*therm* output- or table-file.

The usage of numerical derivatives can be forced with the keyword `num_grad`. The defaults for the forced numerical differentiation are as described above. If possible, central derivatives will be used, otherwise

---

[107] Klamt, A.; Krooshof, G.J.P.; Taylor, R., *AIChE Journal*, **48**, 2332 (2002).

forward derivatives are calculated. The usage of forward derivatives can be forced with the keyword `grad_2point`. Although the forced use of forward derivatives will save considerable amounts of calculation time during the numerical differentiation, it should be used with care, as the quality of the gradients thus computed somewhat decreases and numerical noise increases. The stepsize of the finite diffences by default is taken to be ΔT = 0.5 K, and Δx = 0.001. These values can be changed with the keywords `numgrad_dT=ΔT` and `numgrad_dx=Δx`, respectively.

4.6.2 Partial Enthalpy

In COSMO*therm* the enthalpy ***H*** is computed as expectation value over the microscopic interaction energies E_i of contacting surface pieces:

$$H = \sum_i E_i = \langle E \rangle \qquad (5.6\text{-}3)$$

This formulation has the advantage that the enthalpy can be calculated without additional computational effort. Moreover, it can be split up into the different interaction energy types of electrostatic misfit H_{MF}, hydrogen bonding H_{HB}, and dispersion H_{vdW}, all of which enter the COSMO-RS energy (see section 1.2). The definition of enthalpy as expectation value has the disadvantage however, that the *H* of equation 5.6-3 is an observable only as net enthalpy of the whole system. As the summation of the interaction energies E_i to the expectation value <*E*> in equation 5.6-3 is based upon the contact statistics of the complete ensemble of interacting surface pieces (i.e. on the contact statistics of the total system, which may be a pure solvent, or a given mixture), it is not possible to obtain the observable partial enthalpies $\boldsymbol{H_i}$ of the indvidual compounds from the expectation value. The enthalpies of equation 5.6-3 are observables only for computations of pure compounds and for mixtures where the sum of all contributions is considered. I.e. the total enthalpy of the mixture $H^{tot} = \sum_i x_i H_i$ and similar system-wide properties like excess enthalpy $H^E = \sum_i \left(x_i H_i - x_i H_i^0\right)$ are observables and can be compared with experimental data. The individual $\boldsymbol{H_i}$ values of a mixture however are not and thus they can not be compared to partial enthalpies derived from experiment.

Partial enthalpies H_i that serve as observables can be computed as partial derivatives of the system enthalpy with respect to mole numbers n_i:

$$H_i = \left(\frac{\partial nG}{\partial n_i}\right) - T\left(\frac{\partial^2 nG}{\partial n_i \partial T}\right) = \mu_i - T\left(\frac{\partial \mu_i}{\partial T}\right) \qquad (5.6\text{-}4)$$

I.e. the partial enthalpy H_i of compound *i* can be calculated with the help of the temperature gradient of the chemical potential $(\partial\mu_i/\partial T)$ as described in the previous section. If no analytic gradients are available, the partial enthalpies H_i can also be determined numerically, through finite temperature derivation of the chemical potential, or via finite concentration derivation of the overall enthalpy H:

$$H_i = \frac{\mu_i(T_2)/T_2 - \mu_i(T_1)/T_1}{1/T_2 - 1/T_1} \qquad (5.6\text{-}4a)$$

$$H_i = \left(\frac{\partial nH}{\partial n_i}\right) = H + (1 - x_i)\left(\frac{\partial H}{\partial x_i}\right) \qquad (5.6\text{-}4b)$$

The calculation of partial enthalpies is activated by the keyword **`H_partial`**. Unlike the `grad` option, this keyword will not create additional output in the output or table files of not COSMO*therm*. Instead, all occurences of enthalpies in the output files will be replaced with the thermodynamically correct observable H_i as computed by eq. 5.6-4, as is demonstrated in example 5.6-3.

Example 5.6-3:

```
Compound:  1  (h2o)
 Chemical potential of the compound in the mixture :        -1.98266 kcal/mol
…
 Total mean interaction energy in the mix (H_int)  :        -6.66766 kcal/mol
 Misfit interaction energy in the mix (H_MF)       :         0.31562 kcal/mol
 H-Bond interaction energy in the mix (H_HB)       :        -6.09658 kcal/mol
 VdW interaction energy in the mix (H_vdW)         :        -0.88671 kcal/mol
…
 Compound:  2  (ethanol)
 Chemical potential of the compound in the mixture :        -2.68992 kcal/mol
…
 Total mean interaction energy in the mix (H_int)  :        -7.15514 kcal/mol
 Misfit interaction energy in the mix (H_MF)       :         1.26116 kcal/mol
 H-Bond interaction energy in the mix (H_HB)       :        -4.44115 kcal/mol
 VdW interaction energy in the mix (H_vdW)         :        -3.97515 kcal/mol
…
 Compound:  3  (1-octanol)
 Chemical potential of the compound in the mixture :        -5.91792 kcal/mol
…
 Total mean interaction energy in the mix (H_int)  :       -12.23900 kcal/mol
 Misfit interaction energy in the mix (H_MF)       :         1.64653 kcal/mol
 H-Bond interaction energy in the mix (H_HB)       :        -3.88109 kcal/mol
 VdW interaction energy in the mix (H_vdW)         :       -10.00444 kcal/mol
…
```

In addition to the COSMO*therm* output file, the enthalpies will be replaced in the table of mixture calculations that are using the `ctab` command to create a table output for the mixture calculation.

Example 5.6-4:

```
 Property  job 2 : Mixture - compound properties ;
 Settings  job 2 : T= 298.15 K ; x(1)= 0.3333 x(2)= 0.3333 x(3)= 0.3333 ;
 Units     job 2 : Concentrations in mole fractions ; Energies in kcal/mol ; Pressure in mbar ; Area in A^2 ; Volume in A^3

  Nr Compound                          mu      …      H_int         H_MF         H_HB         H_vdW …
   1 h2o                         -1.98266      …   -6.66766      0.31562     -6.09658      -0.88671 …
   2 ethanol                     -2.68992      …   -7.15514      1.26116     -4.44115      -3.97515 …
   3 1-octanol                   -5.91792      …  -12.23900      1.64653     -3.88109     -10.00444 …
```

Similar to the `grad` option, the `H_partial` option will compute the required chemical potential derivatives silently in the background. If possible, analytic gradients will be used. Otherwise numerical derivatives are computed. The usage of numerical derivatives can be forced with the keyword `num_grad`. The defaults for the forced numerical differentiation are as described above. If possible, central derivatives will be used, otherwise forward derivatives are calculated. The usage of forward derivatives can be forced with the keyword `grad_2point`. The stepsize of the finite diffences by default is taken to be ΔT = 0.5 K, and Δx = 0.001. These values can be changed with the keywords `numgrad_dT=ΔT` and `numgrad_dx=Δx`, respectively.

Property options related to derivatives of the chemical potential

`grad`	Toggle computation of chemical potential gradients with respect to the temperature and mixture composition.
`H_partial`	Toggle computation of observable partial enthalpies.

Suboptions of the `grad` and `H_partial` options:

`num_grad`	Optional: Force numerical computation of chemical potential gradients and partial enthalpies.
`grad_2point`	Optional: Use forward differences in the numerical computation of chemical potential gradients and partial enthalpies.
`numgrad_dT=ΔT`	Optional: Change temperature stepsize of the numerical computation of chemical potential gradients and partial enthalpies. Argument `ΔT` is expected to be a positive real number temperature step in [K].
`numgrad_dx=Δx`	Optional: Change temperature stepsize of the numerical computation of chemical potential gradients and partial enthalpies. Argument `Δx` is expected to be a positive real number mole fraction concentration step.

4.7 Surface Contact Statistics

4.7.1 Computation of Contact Statistics

COSMO*therm* allows the computation of the contact probability of molecules and molecule surface segments in arbitrary mixtures via the `contact` and the `segment_contact` options, respectively. The net contact probability P_{AB} of molecule A with molecule B is given as

$$P_{AB} = \frac{x_B \sum_{i \in A} \sum_{j \in B} A_i A_j \gamma_i \gamma_j e^{-E_{ij}/kT}}{A^{A}_{total} A_{total}} \tag{5.7-1}$$

Where *i, j* are indices of the surface segments of molecule A and B, respectively ($i=1,\ldots,n_A$), x_B is the mole fraction of molecule B in the mixture, A_i , A_j are the segment surface area for surface segments of molecule A and B, respectively, $A^A{}_{total}$ is the total surface area of molecule A, A_{total} is the total surface area of all molecules in the mixture, γ_i , γ_j are the segment activity coefficients for surface segments of molecule A and B, A_i , A_j are the segment surface areas for surface segments of molecule A and B, respectively and E_{ij} is the surface segment interaction energy. Note that P_{AB} is not necessarily equal to P_{BA}.

If the **`contact`** option is given in the global command section of the COSMO*therm* input file, the contact interactions P_{AB} of all possible combinations of the given molecules will be computed for all mixtures given in the mixture section of the COSMO*therm* input file. The values of P_{AB} will be written to the COSMO*therm* output file. If given in a temperature/mixture line of the COSMO*therm* input file, the `contact` option is active only for the temperature / mixture line where it is given. If the option `contact` is given without argument, P_{AB} is printed for all compounds in the mixture. If the option `contact={i`$_1$ `i`$_2$`}` is given, the contact statistics is printed *only* for compounds `i`$_j$, where `i` is the number of the compound in the order that is given in the compound section of the COSMO*therm* input file. If the option `ncontact={name`$_1$ `name`$_2$`}` is given, the contact statistics is printed *only* for compounds `name`$_1$, `name`$_2$ ..., where `name`$_i$ is the name given in the compound section of the COSMO*therm* input file. If conformers of compounds are used, the contact statistics will be averaged over conformer weights as computed for the given temperature and mixture condition. If detailed conformer information output is requested by the global or mixture section keyword `wconf` (see section 2.1) the individual contact probabilities of all given conformers with nonzero concentration in the given mixture will be printed to the molecule/conformer section in the output file, in addition to.the conformer properties and weight factors.

It is also possible to calculate a more detailed contact interaction statistics of all segments of molecule A with the molecule B. This can be achieved with the command **`segment_contact={i`$_1$ `i`$_2$ `…}`** in a temperature / mixture line of the COSMO*therm* input file. This command is active only in the temperature / mixture line where it is given. If this option is given, a detailed contact statistics is computed, which gives the contact interaction probability of all individual segments of compound A with the complete molecule B. In addition, the contact interaction probabilities of all atoms of compound A with the complete molecule B are computed. The detailed contact statistics will be written to an additional output file, the "*contact statistics map file*" called **`name.contact`**, where `name` is the name of the COSMO*therm* input file. The detailed contact statistics file can be renamed by input option **`namcont=yourname.contact`**, which overrides the default `name.contact`. The contact statistics are written *only* for compounds `i`$_j$, where `i` is the number of the compound in the order that is given in the compound section of the COSMO*therm* input file. If the **`segment_contact`** option is given without argument, the detailed contact statistics map will be written for *all* possible contacts of *all* compounds that are present in the given mixture at a finite concentration. If certain compounds consist of several conformers, by default only *one* contact statistics map per compound and contact is printed to the contact statistics map file. If no other input is given, the first conformer as given in the input section is used for the contact statistics map. If the additional keyword **`wlconf`** is given, the contact statistics maps of the conformer with the lowest free energy in the given mixture (or, equivalently the conformer with the highest conformer weight) is used to compute the contact statistics map of the given compound. If however, the detailed conformer information output is requested by the global or mixture section keyword `wconf` (see section 2.1) then the individual contact probabilities of *all* given conformers with nonzero concentration in the given mixture will be computed and written to the contact statistics map file.

In addition to the default of the `segment_contact` option, where the contacts of the segments of a molecule A with the complete molecule B are computed, it is also possible to compute site-specific contact probabilities, i.e. the contacts of the segments of a molecule A with specific sites on molecule B. The sites on molecule B can be defined as single surface segments, single atoms, or groups of atoms. Using the additional command **`1seg={`i_{mol} i_{seg}`}`** in combination with the `segment_contact={`i_1 i_2 …`}` command, the detailed contact interaction statistics of all individual segments of compound A (i.e. molecule i_j as given in the `segment_contact={`i_1 i_2 …`}` command) with one single segment **i_{seg}** of molecule B (molecule **i_{mol}** as given in the `1seg={`i_{mol} i_{seg}`}` option) are computed. The resulting contact interaction map thus quantitatively describes the probabilities that segments j_{seg} of molecule A will have contact with the given segment i_{seg} of molecule B. Using the additional command **`1atm={`i_{mol} i_{atom}`}`** in combination with the `segment_contact={`i_1 i_2 …`}` command, the detailed contact interaction statistics of all individual segments of compound A (i.e. molecule i_j as given in the `segment_contact={`i_1 i_2 …`}` command) with one given atom **i_{atom}** of molecule B (molecule **i_{mol}** as given in the `1atm={`i_{mol} i_{atom}`}` option) are computed. The resulting contact interaction map thus quantitatively describes the probabilities that segments j_{seg} of molecule A will have contact with the all of the segment that are associated with atom i_{atom} of molecule B. If you substitute the atom number i_{atom} by the string "`ALL`", (e.g. `1atm={`i_{mol} `ALL}`) then the contact statitics with all atoms of the given molecule i_{mol} is computed and printed to the contact-file. In addition, an atom-atom contact matrix is printed to COSMO*therm* table file. Using the additional command **`natm={`i_{mol} i_{atom1} i_{atom2} i_{atom3} …`}`** in combination with the `segment_contact={`i_1 i_2 …`}` command, the detailed contact interaction statistics of all individual segments of compound A (i.e. molecule i_j as given in the `segment_contact={`i_1 i_2 …`}` command) with the functional group defined by the atoms **i_{atom1},i_{atom2},i_{atom3},…** of molecule B (molecule **i_{mol}** as given in the `natm={`i_{mol} i_{atom1} i_{atom2} …`}` option) are computed. The resulting contact interaction map thus quantitatively describes the probabilities that segments j_{seg} of molecule A will have contact with the all of the segment that are associated with the functional group defined by the given atoms i_{atomj} of molecule B.

Suboptions of the `segment_contact={`i_1 i_2 …`}` option:

`1seg={`i_{mol} i_{seg}`}`	Optional: Compute statistics of the surface segment contacts between all segments of molecule i_j as given in the command `segment_contact={`i_j …`}` with segment i_{seg} of molecule i_{mol}. This option is active only for the temperature / mixture line where it is given. The contact statistics is printed *only* for compound i_{mol}, where `i` is the number of the compound in the order that is given in the compound section of the COSMO*therm* input file. The molecular surface contacts for all segments of the given compounds are written to the contact statistics file `name.contact`, where `name` is the name of the COSMO*therm* input file.

`1atm={`i_{mol} i_{atom}`}`
or
`1atm={`i_{mol} `ALL}`
	Optional: Compute statistics of the surface segment contacts between all segments of molecule i_j as given in the command `segment_contact={`i_j …`}` with all of the segments that are associated with atom i_{atom} of molecule i_{mol}, or, if the string "`ALL`" is given instead of i_{atom}, with all atoms of molecule i_{mol}. This option is active only for the temperature / mixture line where it is given. The contact statistics is printed only for compound i_{mol}, where `i` is the number of the compound in the order that is given in the compound section of the COSMOtherm input file. The molecular surface contacts for all segments of the given compounds are written to the contact statistics file `name.contact`, where `name` is the name of the COSMOtherm input file.

`natm={`i_{mol} i_{atom1} i_{atom2} i_{atom3} …`}`	Optional: Compute statistics of the surface segment contacts between all segments of molecule i_j as given in the command `segment_contact={`i_j …`}` with all of the segments that are associated with the group of atoms i_{atomk} of molecule i_{mol}. This option is active only for the temperature / mixture line where it is given. The contact statistics is printed only for compound i_{mol}, where `i` is the number of the compound in the order that is given in the compound section of the COSMOtherm input file. The molecular surface contacts for all segments of the given compounds are written to the contact statistics file `name.contact`, where `name` is the name of the COSMOtherm input file.

4.7.2 Visualization of Contact Statistics

COSMO*therm* is able to visualize the detailed segment contact interaction maps `name.contact` that were created by the `segment_contact={i₁ i₂ …}` option as a VRML file[25]. COSMO*therm* can read in a contact statistics map file `name.contact` that has been created with the `segment_contact` option via the command **`wrlmap=name.contact`** in the compound input section of the COSMO*therm* input file. Via the `wrlmap` command COSMO*therm* will create a VRML file `molec_map.wrl` that visualizes the contact interaction probability as given in file `name.contact` on the molecular COSMO surface of molecule `molec`. The filename of the VRML file to be created can be given by the **`namwrl=mapname.wrl`** command, which overrides the default name `molec_map.wrl`. If the `name.contact` file contains more than one segment contact map, a cascade of VRML files will be created for each of segment map entries in the `name.contact` file. The VRML files created will be named by additional numbers according to the entries in the `name.contact` file: `molec_map1.wrl`, `molec_map2.wrl`, `molec_map3.wrl`, … will be created for the first, second third,… entry in the `name.contact` file. If the output name of the VRML files id given with the `namwrl=mapname.wrl` command, then the VRML files will be numbered on the basis of the given name: `mapname1.wrl`, `mapname2.wrl`, `mapname3.wrl`, … will be created for the first, second third,… entry in the `name.contact` file. The visualization of a contact map file can be done either in the compound input section or in the mixture input section (either in the same input line where the contact file was created or in a subsequent mixture line). Please see section 2.2 for details on the VRML file creation procedure. Please note that the contact probabilities visualized as VRML surfaces are scaled by the apparent surface area, i.e. not the absolute contact numbers, as given in the contact file are visualized, but the relative contact strengths per surface area (contact/area), which is in direct analogy to the visualization of a COSMO-file surface charges in terms in terms of surface σ (charge/area). It is possible to force the visualization of the area-dependent contact probabilities with the keyword `absconwrl`.

The color scheme of the contacts visualized, by default is determined from the minimum and maximum values of the given contacts. This automatic assignment of the color scheme according to the property minimum and maximum has the advantage of building a stable framework for the coloring of the VRML file, which always remains within the typical color scheme that is used for surface charge visualization. It has the disadvantage that the contact probabilities of different molecules can not be compared visually, because the color scheme of the contact surface visualized is determined individually for each molecule's contact surface due to it's contacts minimum and maximum values. Moreover, this kind of color designation may lead to visual artifacts if the absolute values of the scaled contacts are very similar, i.e. if the difference between the minimum and maximum contact/area is small. In such a case the color scheme may be blown out of proportion and exaggerate minimal differences, or even numerical noise.

To avoid the kind of ambiguity that is caused by a min/max value based coloring scheme, COSMO*therm* offers two additional color schemes for the visualization of contacts, namely the options `absconscale` and `relconscale`. Both options provide an "absolute" color scheme that is independent of the individual numerical distribution of the contact values in a contact file, thus allowing the direct visual comparison of the contact surfaces of different molecules or conformers in a qualitative to semi-quantitative manner. For both of the "absolute" color schemes, colors are assigned in a way that a typical color distribution in direct analogy to the surface σ (charge/area) color scheme is achieved. I.e. the average contact is green, while stronger-than-average contacts with increasing contact strength will settle in the color range of yellow, orange, red, and brown. Weaker-than-average contacts with decreasing contact strength will settle in the color range between turquoise, blue, and purple. In each case the color reference is the contact map of the compound with itself (i.e. a surface that is completely green). The two absolute color schemes differ in the way the color distribution is spread with respect to the contact strength relative to the reference state: for option `absconscale` the contact's color distribution is scaled by the "ideal" contact strength, which is just the surface concentration ratio of the contacting molecules, whereas for option `relconscale` it is scaled by the "apparent" contact strength, which is the sum of the contacts in this molecule. Thus typically, the `absconscale` option will show a somewhat weaker color spread than the `relconscale` option. Due to their "absolute" nature, which is calibrated to best visualize typical contact strengths of neutral molecules, both color schemes may overshoot if extreme contact strengths are present in a molecule. Please also note that the `absconwrl`, `absconscale`, and `relconscale` options are mutually exclusive. All of the contact coloring options (including the default min/max scheme) may be overridden by the explicit input of the minimum and maximum of the color scheme by the `wrl_min` and `wrl_max` keywords.

Control options for the visualization of surface contact statistics map files:

`wrlmap=mapfile`	Optional: Create a VRML file[25] of the molecular COSMO surface property map that is given in file `mapfile`. The `wrlmap` command is active only for the actual compound and the actual property map `mapfile`. The property map file `mapfile` is expected to be in the same format as the surface contact statistics map file `name.contact` that can be created by COSMO*therm* via command `segment_contact`. The `mapfile` is expected to be found in the directory where the COSMO*therm* input file is.
`namwrl=name.wrl`	Optional (Suboption of the `wrlmap=mapfile` command): Give the name of the VRML visualization file of the property map given by the `wrlmap=mapfile` command. `name.wrl` overrides the default VRML visualization file name `molec_map.wrl`.
`map_column=`i_{col}	Optional (Suboption of the `wrlmap=mapfile` command): Choose column i_{col} of the COSMO surface property map `mapfile` that shall be visualized on the COSMO surface. By default the 6th real number column of `mapfile` is expected to hold the property that should be visualized. If property map file `mapfile` is a surface contact statistics file that was created by COSMO*therm* via command `segment_contact`, the 6th real number column holds the contact interaction statistics (see section 0). The `map_column` command allows the visualization of other properties in file `mapfile`. For example, the surface potential on the COSMO surface that is given in the 7th column of a surface contact statistics file created by COSMO*therm* can be visualized via `map_column=7`.

Control options for the visualization of surface contact statistics map files (continued):

`absconwrl`	Optional (suboption of the `wrlmap=mapfile` command): Visualize contact statistics that are not scaled by the apparent surface area. This option is only valid if segment contact statistics are visualized from a contact-file, i.e. if the default visualization is used, or if `map_column=6` is given.
`absconscale`	Optional (suboption of the `wrlmap=mapfile` command): Use "absolute" coloring scheme scaled by surface concentration ratio of the contacting molecules. This option is only valid if segment contact statistics are visualized from a contact-file, i.e. if the default visualization is used, or if `map_column=6` is given.
`relconscale`	Optional (suboption of the `wrlmap=mapfile` command): Use "absolute" coloring scheme scaled by the apparent overall contact probability of the contacting molecules. This option is only valid if segment contact statistics are visualized from a contact-file, i.e. if the default visualization is used, or if `map_column=6` is given.
`wrl_min=min_val`	Optional (suboption of the `wrlmap=mapfile` command): Set the minimum value `min_val` for the property visualization of a COSMO surface property map mapfile. By default this value is determined automatically from the smallest value of the property given in the property column of mapfile. If the `wrl_min` command is used, the color management for the COSMO surface property map is adjusted according to the given `min_val`.
`wrl_max=max_val`	Optional (suboption of the `wrlmap=mapfile` command): Set the maximum value `max_val` for the property visualization of a COSMO surface property map mapfile. By default this value is determined automatically from the largest value of the property given in the property column of mapfile. If the `wrl_max` command is used, the color management for the COSMO surface property map is adjusted according to the given `max_val`.

4.7.3 Creation of Cluster and Ion Geometries based on Contact Statistics

Using the **`SSC_PROBABILITY`** or **`SSC_STRENGTH`** option it is possible to create geometry files of the most probable surface-segment-contacts (SSC) of any two molecules in a given mixture. These options may be helpful to visualize and thus better understand the actual contact interactions between any two molecules that are present in a given mixture. Moreover, these options may be used to find the most probable structures of the molecular complexes or clusters built (or not built) by two given compounds (e.g. binary complexes bound by intermolecular hydrogen bonds or by charge-transfer complexation) and provide reasonable initial geometries of such complexes, which in the following may be optimized by a quantum chemical method. Only complexes that are not hindered sterically or by van-der-Waals interactions will be created. Thus by default not all possible complexes actually will be created. You may force the creation of additional complexes using the keyword `SSC_WEAK`, which forces the creation of complexes with weak contacts. Option `SSC_PROBABILITY` will create geometries of complexes based on the criterion of highest contact probabilities between two molecules, while option `SSC_STRENGTH` will create geometries of complexes based on the criterion of lowest interaction energy between the surface segments of two molecules. Please note that the `SSC_PROBABILITY` segment contact criterion is somewhat more strict and tends to create a lower number of complex geometry files (only complexes with strong interactions, such as intermolecular hydrogen bonds will be created), while the `SSC_STRENGTH` interaction energy criterion is less tight and will also create complexes of unpolar molecules, which can not build intermolecular hydrogen bonds or charge-transfer complexes. For the complex formation options `SSC_PROBABILITY` and `SSC_STRENGTH` it is possible to force the contact of two molecules at specific sites within the two complex-forming molecules. This can be achieved by adding an argument bracket to SSC option, i.e. SSC_PROBABILITY={C n_1 n_2 …} or SSC_STRENGTH={C n_1 n_2 …} (see suboptions table below). These special SSC options may be helpful if a specific cluster-complex geometry is sought that is not the geometry of the most probable contact or lowest interaction energy obtained from the `SSC_PROBABILITY` and `SSC_STRENGTH` options (as given without argument).

In addition to searching the segments on the molecular surfaces of the two compounds that have the most probable contact or the lowest interaction energy, the SSC options will probe several orientations of the two molecules in the complex and create the complex geometries of the most favourable orientations. By default the dihedral angle between the two contacting molecules in the cluster is varied with a stepsize of 45°. This stepsize can be changed using the keyword `ssc_ang=angle`.

Apart from cluster-complex geometries it is possible to create ionic structures using the **`SSC_IONS`** suboption of the `SSC_PROBABILITY` or `SSC_STRENGTH` options. If this keyword is given, COSMO*therm* will create two additional geometry xyz-files for each cluster. The ions are built are created by shifting the hydrogen atom that is doing the contact interaction from one cluster compound to the other one, thus simulating the dissociation reaction of the interacting compounds. The `SSC` options, i.e. the contact cluster complex and ion geometry generation tools by create produce geometry files that are indexed by the molecule numbers in the order as given in the input. Sometimes it is more convenient to have the created geometry files named by the actual molecule names. This can be done with the **`SSC_NAME`** keyword.

Suboptions of the `contact={i`$_1$ `i`$_2$ `…}` option (creation of cluster and ion geometry files):

`ssc_probability`	Optional: Create geometry files of the most probable surface-segment-contacts (SSC) for all compounds given in the `contact` option with all other compounds in the mixture. Molecular surface contact complex geometry files will be written for the contacts of all conformers of the compounds that are given in the `contact={i`$_1$ `i`$_2$ `…}` command with all conformers of the other compounds `j` that are present in the mixture. No complex geometry file will be written for molecules that are present at infinite dilution only. Moreover, no complex geometry file will be written if the contact between the two molecules is hindered by unfavourable van-der-Waals interaction. The surface contact complex geometry files will be written to the current working directory in Cartesian Xmol (XYZ) format[23]. The surface contact complex geometry files will be named `complex_i_j_angle.xyz`, where `i` is the number of the molecule/conformer in the order that is given in the compound section of the COSMO*therm* input file, `j` is the number of the partner molecule/conformer in the order that is given in the compound section of the COSMO*therm* input file and `angle` is the dihedral angle orientation of two molecules in the complex. Please note that the surface contact complex XYZ-geometry files of subsequent `contact` runs will be overwritten.
`ssc_strength`	Optional: Create geometry files of the lowest energy surface-segment-contacts (SSC) for all compounds given in the `contact` option with all other compounds in the mixture. Molecular surface contact complex geometry files will be written for the contacts of all conformers of the compounds that are given in the `contact={i`$_1$ `i`$_2$ `…}` command with all conformers of the other compounds `j` that are present in the mixture. No complex geometry file will be written for molecules that are present at infinite dilution only. Moreover, no complex geometry file will be written if the contact between the two molecules is hindered by unfavourable van-der-Waals interaction. The surface contact complex geometry files will be written to the current working directory in Cartesian Xmol (XYZ) format[23]. The surface contact complex geometry files will be named `complex_i_j_angle.xyz`, where `i` is the number of the molecule/conformer in the order that is given in the compound section of the COSMO*therm* input file, `j` is the number of the partner molecule/conformer in the order that is given in the compound section of the COSMO*therm* input file and `angle` is the dihedral angle orientation of two molecules in the complex. Note that the complex XYZ-geometry files of subsequent `contact` runs will be overwritten.

Suboptions of the `contact=`$\{i_1\ i_2 \ldots\}$ option (creation of cluster and ion geometry files, continued):

`ssc_probability=` $\{C\ n_1\ n_2\ C\ n_1\ n_2 \ldots\}$	Optional: Create geometry files of the most probable surface-segment-contacts (SSC) for all compounds given in the `contact` option with all other compounds in the mixture. This option behaves similar to the `ssc_probability` option described above, with the difference that generation of the interaction complexes is restricted to the boundary conditions given by the argument group "$C\ n_1\ n_2$", where `C` is a single character that may be "`A`" or "`E`" and n_1 and n_2 are integer numbers or element symbols. It is possible to give several "$C\ n_1\ n_2$" argument groups in row. There are two possible usages of the option: 1) If the character "`A`" is given, the integer numbers n_1 and n_2 will be interpreted as atom numbers of the two compounds. I.e. the generation of the interaction complexes is restricted to the contacts between atom n_1 of molecule 1 and atom n_2 of molecule 2 only. Thus via the `ssc_probability=`$\{A\ n_1\ n_2\}$ option complexes with specific atom-atom interaction patterns can be created. 2) If the character "`E`" is given, the integer numbers n_1 and n_2 will be interpreted as element numbers (e.g. 1 for hydrogen, 8 for oxygen). Alternatively, the element symbols can be given (e.g. "`A H Cl`"). Element numbers and symbols can also be mixed. Either way, the generation of the interaction complexes is restricted to the contacts between elements n_1 and n_2 only. Thus via the `ssc_probability=`$\{E\ n_1\ n_2\}$ option complexes with specific element-element interaction patterns (e.g. hydrogen to oxygen) can be created.

Suboptions of the contact={i_1 i_2 …}option (creation of cluster and ion geometry files, continued):

ssc_strength= {C n_1 n_2 C n_1 n_2 …}	Optional: Create geometry files of the lowest energy surface-segment-contacts (SSC) for all compounds given in the contact option with all other compounds in the mixture. This option behaves similar to the ssc_strength option described above, with the difference that generation of the interaction complexes is restricted to the boundary conditions given by the argument group "C n_1 n_2", where C is a single character that may be "A" or "E" and n_1 and n_2 are integer numbers. It is possible to give several "C n_1 n_2" argument groups in row. There are two possible usages of the option: 1) If the character "A" is given, the integer numbers n_1 and n_2 will be interpreted as atom numbers of the two compounds. I.e. the generation of the interaction complexes is restricted to the contacts between atom n_1 of molecule 1 and atom n_2 of molecule 2 only. Thus via the ssc_probability={A n_1 n_2} option complexes with specific atom-atom interaction patterns can be created. 2) If the character "E" is given, the integer numbers n_1 and n_2 will be interpreted as element numbers (e.g. 1 for hydrogen, 8 for oxygen). Alternatively, the element symbols can be given (e.g. "A H Cl"). Element numbers and symbols can also be mixed. Either way, the generation of the interaction complexes is restricted to the contacts between elements n_1 and n_2 only. This way, complexes with specific element-element interaction patterns (e.g. hydrogen to oxygen) can be created.

Suboptions of the `contact={i`$_1$ `i`$_2$ `…}`option (creation of cluster and ion geometry files, continued):

`ssc_ions`	Optional: Create molecular structures of ions by "dissociating" a cluster geometry created by the `ssc_strength` or `ssc_probability` option. The ion geometry files will be written to the current working directory in Cartesian Xmol (XYZ) format[23]. The surface contact complex geometry files will be named `ionA_i_j.xyz`, and `ionB_i_j.xyz`, where `i` is the number of the molecule/conformer in the order that is given in the compound section of the COSMO*therm* input file, `j` is the number of the partner molecule/conformer in the order that is given in the compound section of the COSMO*therm* input file and `angle` is the dihedral angle orientation of two molecules in the complex. Please note that the surface contact complex XYZ-geometry files of subsequent `contact` runs will be overwritten.
`ssc_name`	Optional: Write named files with molecular contact cluster or ion geometry created by the `ssc_strength`, `ssc_probability` or `ssc_ions` option: The geometry files will be written to the current working directory in Cartesian Xmol (XYZ) format[23]. The surface contact complex geometry files will be named `complex_name`$_i$`_name`$_j$`_angle.xyz`, where `name`$_i$ and `name`$_j$ are the molecule/conformer names of the contacting compounds and `angle` is the dihedral angle orientation of two molecules in the complex. The ion geometry files created by the `ssc_ions` option will be named `cation_name_i_j.xyz`, and `anion_name_i_j.xyz`, where `name` is the molecule/conformer name of the anion and cation created, respectively, while `i` and `j` are the number of the molecule/conformer contacts. `i` and `j` are given in the order that is given in the compound section of the COSMO*therm* input file. Please note that the surface contact complex XYZ-geometry files of subsequent `contact` runs will be overwritten.
`ssc_weak`	Optional for the `ssc_strength`, `ssc_probability` or `ssc_ions` option: Force the creation of additional, weak surface contact complex XYZ-geometry files.
`ssc_ang=angle`	Optional: Give the stepsize of the dihedral angle used in the creation of cluster geometries by the `ssc_strength` or `ssc_probability` options. Thus for a given dihedral angle stepsize "`angle`", which has to be given as integer degree between 1 and 359 (default: `angle`=45°), surface contact complex geometry files `complex_i_j_angle.xyz` (or `complex_namei_namej_angle.xyz` if the `ssc_name` option is used) will be written for the range of dihedral angles between 0 and 360° with the given stepsize `angle`.

4.8 Concentration Dependent Reactions (COSMO-RS-DARE)

The prediction of thermodynamic mixture properties with COSMO*therm* sometimes involves the additional complication that in solution and dependent on the solute concentration, reactions between solute and solvent or solute and solute occur. Examples for such reactions are the concentration dependent dimerization of small organic acids (solute-solute reaction) or the formation of charge-transfer complexes between solute and solvent. In experimental property measurements, normally the influence of such a reaction on the measured property is not separated from the other effect of the solutes interactions (polar interaction, hydrogen bonding). Figure 5.8-1 shows the activity coefficient plot of the binary system octane (1) – acetic acid (2) at 343 K. This binary system shows a concentration dependent dimerization of acetic acid. At high concentrations of acetic acid, mostly the acids dimer can be expected whereas at low concentration the monomer is more probable. Clearly, the steep ascent of the activity coefficient of acetic acid on the side of low acid concentration in Figure 5.8-1 is due to the fact that at low acid concentration the formation of the dimer is unlikely. Thus the activity coefficient is dominated by the monomeric acetic acid, which is much more polar than the dimer and thus also its activity coefficient in the unpolar octane is larger. Thus the overall activity coefficient of acetic acid in the mixture with octane rises strongly if the concentration dependent dimerization reaction is pushed to the side of the monomer (which is the case at low acid concentrations, if x_1(octane) > 0.95).

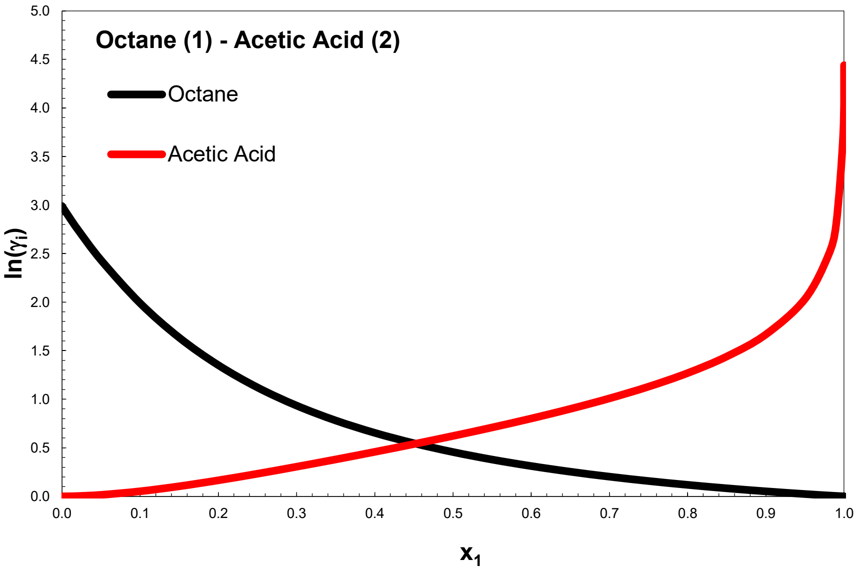


Figure 5.8-1: Activity coefficients of the binary system octane (1) – acetic acid (2) at T=343 K.

COSMO*therm* is able to treat such concentration dependent reactions with the concept of Interaction Energy Indices (**IEI** numbers). The theory has been published as "COSMO-RS with Dimerization, Aggregation, and Reaction Extension" (**COSMO-RS-DARE**)[108]. The reacting compounds have to be treated as pseudo-conformers in the compound input section COSMO*therm* input file. For an equilibrium reaction A + B ↔ A-B, the first conformer of compound A is the free compound A. The second conformer for

---

[108] Sachsenhause, T.; Rehfeld, S.; Klamt, A.; Eckert, F., *Fluid Phase Equilibria*, **382**, 89 (2014).

compound A is a COSMO metafile of the reaction product A-B wherein all atoms of the reaction partner B are set to zero by atomic weights (see section 5.3 for COSMO meta-files and the handling of atomic weights). Thus the second conformer describes the behaviour of compound A after the reaction (not the behaviour of the reaction product A-B). Compound B is constructed similarly (i.e. one conformer is pure compound B and the second conformer is a meta-file of A-B with all atomic weights of A set to zero). The conformer built from the COSMO meta-file has to be identified by a unique interaction energy index (IEI number). This is done with the **`IEI=i`** command in the same line where the compound is given. It is also possible to assign more than one IEI number to a compound using the command **`IEI={`**i_1 i_2 i_3**`}`** (the latter is required only if the compound is able to undergo several reactions with different products). Figure 5.8-1 shows the simple case of the dimerization reaction of acetic acid in a binary mixture with octane (i.e. the equilibrium A + A ↔ A-A). One conformer of the reactive species is monomeric acetic acid and the second conformer is the acetic acid dimer, where one half of the dimer is set to zero in the COSMO meta-file.

The second step in the description of a reaction is the assignment of the interaction energy in the mixture part of the COSMO*therm* input file. This is done with the **`en_IEI={`**i_1 i_2 **`c_H c_S}`** option. i_1 and i_2 are the IEI numbers of the reacting compound as given in the compound input section. The two parameters `c_H` and `c_S` are parameters for the enthalpic and entropic contribution to the interaction energy of the two IEI numbered compounds. `c_H` and `c_S` are [kcal/mol] and [kcal/mol K], respectively. The interaction Gibbs free energy between the two IEI numbers is calculated from equation 5.8-1:

$$G\,(i_1,\ i_2) = -2\,\Delta^0 + \mathbf{c\_H} - T\,\mathbf{c\_S} \tag{5.8-1}$$

Δ^0 is the energy different between the first conformer (A) and the second conformer (meta-file A-B). The value of Δ^0 automatically is calculated by COSMO*therm*. It is possible to give several `en_IEI` entries in one compound input line (for the treatment of several different reactions). Figure 5.8-1 shows the according `en_IEI` entry for the dimerization reaction of acetic acid. The only interaction that has to be taken into account for this reaction is that of IEI number 1 with itself, because only species A (acetic acid with IEI number 1) are involved in the reaction. This is the simplest case of a concentration dependent reaction in solution. The interaction energy of this reaction is determined from eq. 5.8-1 using `c_H` = –3 kcal/mol while `c_S` is zero.

`Example 5.8-1:` COSMO*therm* input

```
ctd=BP_TZVP_C30_1701.ctd cdir=../CTDATA-FILES
fdir=../DATABASE-COSMO/BP-TZVP-COSMO efile vpfile
!! Isothermal binary phase diagram of Octane - Acetic Acid
f = octane.cosmo
[ f = aceticacid.cosmo
  f = aceticacid_dimer_half.mcos ] IEI=1
tc=70.0 binary EN_IEI={1 1 -3.0 0.0}
```

`Example 5.8-2:` **Acetic acid dimer COSMO meta-file** `(aceticacid_dimer_half.mcos)`

```
f=aceticacid_dimer.cosmo W={1111111100000000}
```

4.9 Ionic Liquids

The prediction of thermodynamic properties of ionic liquid (IL) solutions with COSMO*therm* involves a complication that results from the fact that the ionic liquid, which in experiment is seen as one species, in COSMO*therm* should be treated as two individual ions. In general, any salt (liquid or solid) can be described via two different definition of the mole fraction. The salt either can be described as "*one substance*", or, as a "*sum of anion and cation*" in the ratio of their stoichiometric coefficients. While the "*one substance*" approach is quite natural for experimental purposes, the "*sum of ions*" view is more convenient for modeling strategies that take into account the different particle interactions in the mixture. One can imagine three approaches for the calculation of properties of ionic liquid mixtures: I.) Use the two ions as separate molecules with the equal mole fractions. II.) Combine the two ions in a meta file. III.) Use a cosmo file of an ion pair. The third approach does not describe a molten salt very well, because the relative orientation of anion and cation is fixed in this picture. The system consists of ion pairs with the same structure only. The second approach combines the surface charge σ, the area and the volume of the two ions and takes into account interactions with two distinct ions. But the entropic term of the whole IL is used, thus leading to errors in the chemical potential if the IL is present at a finite concentration in the mixture. We recommend the first approach, because the picture of two distinct ions should be closer to reality. However approach I requires a careful look at the definitions and reference states of the computed thermodynamic property: In the COSMO*therm* input file the two IL components are given as individual compounds, see Example 5.9-1:

`Example 5.9-1:` COSMOtherm input for Ionic Liquid

```
ctd=BP_TZVP_C30_1701.ctd cdir=../CTDATA-FILES
fdir=../DATABASE-COSMO/BP-TZVP-COSMO
! Compute activity coefficient of hexane in an IL
f = 4-methyl-n-butylpyridinium.cosmo   # IL cation
f = BF4.cosmo                          # IL anion
f = hexane.cosmo                       # Solute
tk=314 gamma=1 xg={0.5 0.5}            # Activity coefficient computation
```

In example 5.9-1, the ionic liquid is used as a solvent for a third compound present in infinite dilution. For the calculation one has to take care that anion and cation have the same mole fraction. In this example, the activity coefficient of all substances in a mixture of 50 % 4-methyl-N-butylpyridinium-cation and 50 % borontetrafluoride-anion is calculated. Other properties can be calculated in the same way.

Some care has to be taken in the interpretation of the COSMO*therm* results for the computed thermodynamic properties. One has to be aware of the different definitions of the mole fraction in an IL system. Considering example 5.9-1, the COSMO*therm* calculation in the "*sum of ions*" view is a *ternary* mixture calculation: cation, anion and solute *i* with the boundary condition that the molar amount of anion equals the molar amount of cation in the mixture ($n_{ion} = n_{anion} = n_{cation}$):

$$x_S^{ternary} = \frac{n_i}{n_i + 2n_{ion}} \tag{5.9-1}$$

On the other hand, the experimental determination of IL thermodynamic properties in the "*one substance*" view is based on the assumption of a *binary* system consisting of the IL and the solute:

$$x_S^{binary} = \frac{n_i}{n_i + n_{IL}}$$

These two definitions are just different views on the same system. Therefore any calculated value corresponds to both definitions of the mole fraction. The calculated chemical potentials are dependent on only the temperature and the composition of the system on the molecular level. Nevertheless, the two definitions may lead to different values if the mole fraction is used explicitly for the determination of experimental ("*real world*" or "*laboratory binary*" [109]) data. Example: In a given ionic liquid mixture x_i^{binary} and $x_i^{ternary}$ belong to same composition. To obtain the activity coefficient from experimental data one can apply both definitions of the mole fraction.

$$\frac{p_i}{p_i^0} = x_i \gamma_i \qquad (5.9\text{-}2)$$

Because the pressure in eq. 5.9-2 is an observable (a measurable value), the experimentally determined activity coefficient is dependent on the view on the system, i.e. the definition of the mole fraction. The calculated value is dependent on the interactions in the system only. Because both definitions describe the same system (on the molecular level), it is independent of the definition of the mole fraction. If $x_i^{ternary}$ is used to obtain the experimental data, the resulting activity coefficients can be compared directly with COSMO*therm* predictions of γ_i. However, if the x_i^{binary} definition is used for the determination of the experimental data one has to apply a factor to the COSMO*therm* predictions to account for the different macroscopic reference systems (binary and ternary system). The activity coefficient γ_i for a system that is defined as binary system experimentally but calculated as a ternary system in COSMO*therm* (as done in Example 5.9-1) is defined as:

$$x_i^{ternary} \gamma_i^{ternary} = \frac{x_i^{ternary}}{x_i^{ternary} + x_{ion}^{ternary}} \gamma_i^{binary} \qquad (5.9\text{-}3)$$

Eq. (5.9.4) leads to conversion equation 5.9-4 that translates the ternary (COSMO*therm* calculated) activity coefficients to binary (experimentally measured) activity coefficients:

$$\gamma_i^{binary} = \gamma_i^{ternary}\left(x_i^{ternary} + x_{ion}^{ternary}\right) \qquad (5.9\text{-}4)$$

For the calculation of an activity coefficient in infinite dilution in an IL this reduces to:

$$\gamma_i^{binary} = \frac{1}{2}\gamma_i^{ternary} \qquad (5.9\text{-}5)$$

Because now we have the same reference framework of the IL composition for the experimental and calculated system, the COSMO*therm* prediction of the vapor pressures according to eq. 5.9-2 directly can be compared with the experimental data. If however activity coefficients γ_i and properties derived from

---

[109] Historically this term has been coined for mixtures of one IL and one solute. Although it might be a bit misleading, we will continue with this expression even for mixtures with more than three components.

activity coefficients are considered it is crucial to stay within the same definition. Thus typically activity coefficients computed with COSMO*therm* in the "*sum of ions*" or "*ternary*"[109] framework need to be converted to the "*one substance*" or "*laboratory binary*"[109] framework to be comparable to values derived from experiment. A general expression for the conversion between the two frameworks is given below.

In COSMO*therm* three types of mole fraction definitions are used. For ILs with stoichiometry ν_{anion} and ν_{cation} and $\nu_{IL}^{tot} = \Sigma \nu_{ion} = \nu_{anion} + \nu_{cation}$ the mole numbers are the same for all definitions:

$$n_{ion} = n_{cation} = n_{anion} = n_{IL} \tag{5.9-6}$$

In the "*one substance*" or "*laboratory binary*"[109] the mole fraction of a solute *i* is given as

$$x_i^{bin} = \frac{n_i}{\sum_i n_i + n_{IL}} \tag{5.9-7}$$

In the "*sum of ions*" or "*ternary*"[109] framework the mole fraction of a solute *i* is given as

$$x_i^{tern} = \frac{n_i}{\sum_i n_i + \nu_{IL}^{tot} n_{ion}} \tag{5.9-8}$$

In the "*pseudo-binary*"[109] definition that is used in the COSMO*therm* table output, the mole fraction of the IL is expressed as a sum over the ion mole fractions, while the solute *i* mole fractions are the "*ternary*" ones x_i^{tern}. This definition is used in the table bodies of the COSMO*therm* table output. It was introduced to preserve the column structure of the table files:

$$x_i^{pbin} = x_i^{tern}; \quad x_{IL}^{pbin} = \sum_{IL} x_{ion}^{tern} \tag{5.9-9}$$

Using eq. 5.9-9 the conversion of the "*sum of ions*" or "*ternary*" mole fraction x_i^{tern} to the "*one substance*" or "*laboratory binary*" x_i^{bin} definition reads:

$$x_i^{bin} = \frac{x_i^{pbin} / \nu_{IL}^{i}}{\sum_k x_k^{pbin} / \nu_{IL}^{k}} \tag{5.9-10}$$

In eq. 5.9-10 the ν_{IL}^{j} values are equal to ν_{IL}^{tot} (the sum of the anions and cations stoichiometries of the given IL) if compound *j* is the IL, and equal to one if compound *j* is a neutral solute compound. X_j^{pbin} is the "*pseudo-binary*" definition of the mole fraction according to eq. 5.9-9. I.e. x_j^{pbin} is equal to x_j^{tern} for neutral solute compounds. If compound *j* is the IL, x_j^{pbin} is equal to the sum of the the "*ternary*" mole fractions of all components of the IL.

4.10 FlatSurf: Surface Activity

COSMO-RS theory originally was developed to calculate the chemical potential of solutes in homogeneous liquid phases, where the residual part of the chemical potential μ_i^S of solute compound *i* in phase *S* is calculated as

$$\mu_i^{S,res} = \sum_{v \in i} a_v \mu_S(\sigma) \tag{5.10-1}$$

Here $\mu_S(\sigma)$ is the σ-potential of phase S. Sometimes it is of interest to estimate the free energy gain of a molecule at a surface or interface between two different liquid phases *S* and *S'*. Under the idealized assumption of a flat interface we can calculate this energy gain (relative to the free energy in phase *S*) in the following way: For a given position, i.e. distance *z* of the solute center from the interface, and orientation Γ (described by the orientation of a fixed solute axis with respect to the surface normal direction) of the solute relative to the interface a certain part of the surface segments will be imbedded in phase *S* and the rest in phase *S'*. Thus, for fixed *z* and Γ we can calculate the chemical potential as

$$\mu_i^{S,S',res}(z,\Gamma) = \sum_{v \in iinS} a_v \mu_S(\sigma) + \sum_{v \in iinS'} a_v \mu_{S'}(\sigma) \tag{5.10-2}$$

By sampling all relevant positions and orientations we can find the minimum of the free energy of the solute *i* at the flat interface of S and S' (see Figure 5.9-1). Taking into account the free energy difference between different conformations of *i*, the search for the optimal association of *i* at the interface can be extended to conformationally flexible molecules, if required.

Assuming that the free combinatorial part of the free energy stays essentially unchanged, and subtracting the residual chemical potential in bulk phase S we yield the maximum free energy gain of *i* at the S,S' interface. In addition to the maximum free energy gain we can try to calculate a total free energy gain from the interface partition sum:

$$G_i^{S,S'} = -RT \ln\left\{\sum_{z,\Gamma} \exp\left[-\frac{\mu_i^{S,S',res}(z,\Gamma) - \mu_i^{S,res}}{RT}\right]\right\} \tag{5.10-2}$$

The minimum of the free energy of the solute *i* at the flat interface of *S* and *S'*, $\mu_i^{S,S',res}$ of eq. 5.10-2 and the total free energy of the solute *i* at the flat interface of *S* and *S'*, $G_i^{S,S'}$ of eq. 5.10-3 both can be used as significant and thermodynamically rooted descriptors for the determination of surface activity in a solution.

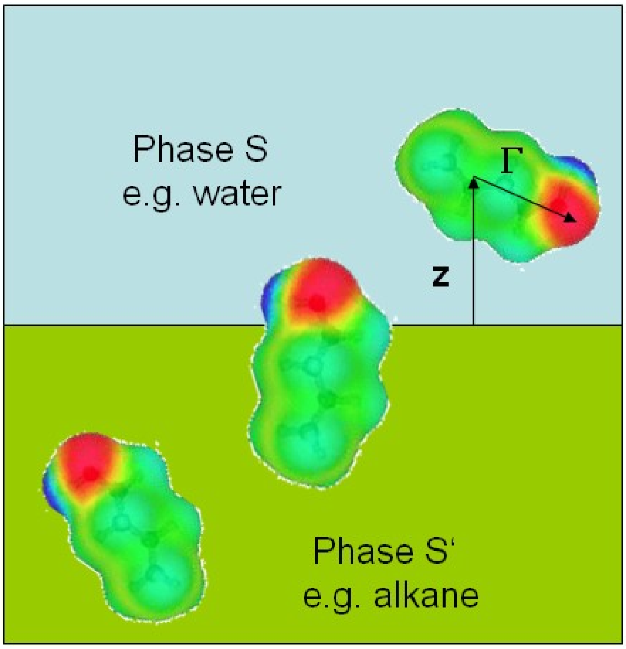


Figure 5.9-1: Schematic illustration of the FlatSurf option.

In COSMO*therm,* the calculation of the flat surface interaction energy between two solvents is toggled via the `flatsurf={i j}` option (`i` and `j` being the compound numbers in the order of the compounds given in the compound input section) or the `nflatsurf={nam_i nam_j}` option (`nam_i` and `nam_j` being the compound names as given in the compound input section of the COSMO*therm* input file). Using either of these automatic calculation options, COSMO*therm* will compute the surface interaction energy at the surface interface of the two solvents for all compounds that are given in the compound input section of the COSMO*therm* input file. The flat surf option will provide the following descriptors for each compound *i* and for each of the two phases and write them to the COSMO*therm* output file and in tabulated from to the COSMO*therm* table file: $\mu_i^{S,S',res}$ (`Gmin`) the minimum of the free energy of the solute *i* at the flat interface of *S* and *S'*; $G_i^{S,S'}$ (`Gtot`) the total free energy of the solute *i* at the flat interface of *S* and *S'*; $\tilde{a}_i^{S,S'}$ (`Across,mean`) the cross section area of the solute *i* at the flat interface; $\bar{a}_i^{S,S'}$ (`Asurf,mean`) the mean expulsion area of the solute *i* at the flat interface; $a_i^{S,S'}$ (`Asurf,min`) the contact area of the solute *i* with phase *S* at the free energy minimum; a_i (`Acomp`) the initial area of the COSMO-surface of solute *I*; $\bar{V}_i^{S,S'}$ (`Vsurf,mean`) the mean volume of the solute *i* at the flat interface; $V_i^{S,S'}$ (`Vsurf,min`) the contact volume of the solute *i* with phase *S* at the free energy minimum; V_i (`Vcomp`) the initial volume of the COSMO-file of solute *I*; z_i^{min} (`Depth`) the distance of the center of mass of the solute *i* from the interface at the free energy minimum; z_i^z (`Depth_z`) the maximum z-coordinate of the distance of the center of solute *i* from the interface at the free energy minimum. If several conformers were used to compute a compounds surface interaction energy, COSMO*therm* will always write the name of the specific conformer to the table output, which was able to achieve the lowest value of $\mu_i^{S,S',res}$ (`Gmin`). Thus from the list of all conformers of a given compound, the one with lowest of all minimum free energy values at the flat interface of *S* and *S'* will be listed. In contrast, $G_i^{S,S'}$ (`Gtot`) the total free energy gain of the solute *i* at the flat interface always is the thermodynamic average according to the interface partition sum of all conformers, if several conformers are present in one compound.

In addition to FlatSurf in pure solvent phases, as described above, it is possible to compute the FlatSurf properties in phases of mixed solvents (e.g. compute the surface chemical potential of a solute between one phase water and another phase consisting of an alkane mixture). For this the concentrations of the compounds in the two phases have to be given in the input. This is possible via the `xf1={`x_1 x_2 `…}` or `cf1={`c_1 c_2 `…}` commands (for the first solvent phase) and the `xf2={`x_1 x_2 `…}…` or `cf2={`c_1 c_2 `…}…` commands (for the second solvent phase). If the phase concentration is not known in advance it is possible to compute the equilibrated phases with the suboption `eq_phases`. This option toggels an upfront phase equilibration calculation using the multi-component - multi-phase equilibrium technique described in section 2.3.12. The pure phases given as arguments of the `logp` command, or the phase concentration input via options `xf1`, `cf1`, `xf2`, and `cf2` will be used as starting concentrations for the phase equilibration. If the `eq_phases` keyword is given, the computation of the chemical potentials that enter the FlatSurf calculation will be done in the phases thus equilibrated.

COSMO*therm* can use the experimental interfacial tension of the two solvent phases to improve the computed FlatSurf energies. This is possible with the `IFT=value` keyword. The value of the interfacial tension is expected to be in [mN/m] = [dyne/cm]. Values for interfacial tensions of various solvent-solvent or air-solvent combinations can be found e.g. in the CRC Handbook of Chemistry and Physics[110]. Please note that the IFT option considerably increases the computational time of a FlatSurf calculation.

The immersion and geometric partition of a solute in the two phases can be visualized with the `Fwrl` command. This keyword either can be given in the global command section or in the mixture input section. It is only active if a FlatSurf calculation is done. The `Fwrl` command provides a graphical representation of the immersion depth *z* of a solute between the two solvent phases. The immersion in the phase is shown on the charge surface in the form of a black and white ring, where the black part of the ring points towards FlatSurf solvent phase 1 and the smaller white part of the ring point towards FlatSurf solvent phase 2. Thus the ring indicates how the solute molecule is immersed in each of the two phases.

Molecules adsorbed to liquid-gas surfaces can be approximately handled by `FlatSurf` representing the gas phase by the *vacuum* pseudo-liquid in terms of the `vacuum_c0.cosmo` file. The use of a vacuum pseudo-liquid however is not an exhaustive representation of the gas phase, as the chemical potential of the molecule in vacuum is not identical with the chemical potential in the gas phase. To obtain a better representation of the gas phase, $G_i^{S,S'}$ (`Gtot`), the total free energy of the solute *i* at the flat interface of the liquid phase *S* and the vacuum phase *V* can be complemented with the liquid-gas transition in terms of ΔG^S_{solv}, the free energy of solvation of solvent *S*. This correction term can be toggled with the option `flat_gas`. Note that the `flat_gas` option will only be active if vacuum is the first of the two phases of the interface.

---

[110] Lide, R. D. (Ed.), *CRC Handbook of Chemistry and Physics*, CRC Press, (2000).

Suboptions of the `FlatSurf` or `nFlatSurf` commands are:

`xf1={`x_1 x_2 `…}`
or
`cf1={`c_1 c_2 `…}`
and/or
`xf2={`x_1 x_2 `…}`
or
`cf2={`c_1 c_2 `…}`
	Optional for `FlatSurf` computations: Give finite mixture concentrations for the two phases between which the FlatSurf properties shall be computed. The input of the concentrations is possible either in mole fractions (`xf1={…}`, `xf2={…}`) or mass fractions (`cf1={…}`, `cf2={…}`) of the compounds of the mixture (as real numbers x_i and c_i). The arguments are expected as real numbers between zero and one in the same sequence of compounds as given in the second input area. If the values do not add up to one, COSMO*therm* will normalize them. If less mole fractions / concentrations than compounds are given, the missing ones will be assumed zero. If a negative number is given, the concentration for this compound will be inserted automatically using the normalization of the sum of mole fractions. Only one negative number is allowed per mixture input line.

`eq_phases`	Optional for `FlatSurf` computations: Compute the phase equilibrium between the given two phases in which the FlatSurf properties shall be computed.
`ift=value`	Optional for `FlatSurf` computations: Give the interfacial tension (`IFT`) of the two phases between which the FlatSurf properties shall be computed. The argument `value` is expected to be in [**mN/m**] = [**dyne/cm**].
`flat_gas`	Optional for `FlatSurf` computations with *vacuum* phase: toggle a more realistic representation of the gas phase by using the free energy of solvation of solvent *S* in the computation of the total free energy at the flat interface between vacuum and solvent.
`Fwrl`	Optional for `FlatSurf` computations: Create VRML files[25] of the molecular COSMO surface charges at the FlatSurf interface. A VRML file `name_flat.wrl` will be created for each molecule `name.cosmo` that is given in the compound input section. The usage of the `Fwrl` command and the created VRML files are identical to the sigma-charge surface VRML files created by the `Swrl` command (cf. the `Swrl` keyword in the compound input section of the COSMO*therm* input file, section 2.2 of this manual), with the small difference that the `Fwrl` command is only possible in combination with a FlatSurf calculation and that the FlatSurf immersion depth between the two phases is shown on the charge surface in the form of a black and white ring, where the black part of the ring points towards FlatSurf solvent phase 1 and the smaller white part of the ring point towards FlatSurf solvent phase 2. I.e. the ring shows how the solute molecule is immersed in the two phases.

4.11 Fine Tuning of COSMO-RS parameters

It is possible to adjust the COSMO-RS method to achieve better predictions for a specific system. This is decribed further in the article "Prediction, fine tuning, and temperature extrapolation of a vapor liquid equilibrium using COSMO*therm*" *Fluid Phase Equilibria* **260** (2007) 183[111]. There are two principal approaches to the task of trying to obtain optimal results with respect to given reference data for a specific molecule system and a specific prediction property. Both approaches are somewhat complimentary and can be used together in COSMO*therm*:

I) from the viewpoint of the molecule or compound, it is possible to modify the COSMO charge surface of a molecule, thus accounting for the deficits of the quantum chemical DFT-COSMO calculations for the specific molecule used. In practice, such a modification of COSMO charges can be done with the technique of "**difference charges**", which is described in section 4.3.1 of this manual.

II) from the viewpoint of the thermodynamic interaction energy contribution (i.e. from the viewpoint of the interacting system of compounds) it is possible to modify the microscopic COSMO-RS interaction energy contributions in the COSMO*therm* program to adjust the COSMO*therm* predictions to match certain given thermodynamic property data. However, before changing any COSMO-RS parameters it is important to be aware of the sometimes severe ramifications of the parameter modification. All parameters in the COSMO-RS model are dependent upon each other in a non-linear way. This means that:

any modification reduces the "predictive" capabilities of the COSMO-RS as a whole. E.g. if the parameters are modified in a way that alkanes are predicted well, then inevitably polar molecules like ethanol will be predicted much much worse. Thus any "modified-parameter" COSMO-RS will not be general any more. It will not be applicable any more to compounds with a different chemical functionality.

any modification of a single parameter indirectly (via the computed segment-contact statistics that results from the solution of the COSMO-RS equation) affects all other parameters. Thus if one predicted property may get better by modifying a parameter, other may get worse.

it does not make sense to change absolute numbers to a given value, e.g. from a different published parameter set, because the interdependence with the other parameters is nonlinear and ultimately unknown. Any strong modification of single parameters may destroy the method as a whole.

With the objections raised above it is possible to modify the parameters to get better results for a given compound system. It is possible to modify all the microscopic energy contributions E_misfit, E_HB and E_vdw as well as the gas phase energy contribution to the vapor pressure. All of the parameters involved can be scaled by a factor given in the COSMO*therm* input. In the COSMO*therm* input file these options

---

[111] A. Klamt, F. Eckert, *Fluid Phase Equilibria* **260** (2007) 183-189 (doi:10.1016/j.fluid.2007.07.055).

have to be given in the global command section (i.e. first or second line of the input file). The following fine tuning scaling parameters can be given:

1) The electrostatic misfit energy contribution to COSMO-RS is given in equation 5.11-1. Therein, σ and σ' are surface charge (sigma) values, a_{eff} is the effective contact area, and α', the misfit prefactor, is an adjustable parameter.

$$E_{MF}(\sigma,\sigma') = a_{eff}\frac{\alpha'}{2}(\sigma+\sigma')^2 \tag{5.11-1}$$

The electrostatic misfit term, can be modified by its prefactor α' via the global `CMFSET=factor` command. This will scale the α' coefficient in the COSMO*therm* parameter set by the given value of `factor`. Please note that any change of the α' misfit prefactor using the `CMFSET` keyword will strongly affect the overall outcome of COSMO*therm*. This option should be used with greatest care!

2) The hydrogen bonding energy contribution to COSMO-RS is given in equation 5.11-2. Therein a_{eff} is the effective contact area; σ_{donor} and $\sigma_{acceptor}$ are surface charge values of hydrogen bonding donor and acceptor sites, respectively, while c_{HB} and σ_{HB} are adjustable parameters.

$$E_{HB}(\sigma,\sigma') = a_{eff}c_{HB}\min\left(0;\min\left(0;\sigma_{donor}+\sigma_{HB}\right)\max\left(0;\sigma_{acceptor}-\sigma_{HB}\right)\right) \tag{5.11-2}$$

The hydrogen bonding energy contribution can be modified in four different ways: first, the general hydrogen bonding prefactor c_{HB} can be scaled via option `CHBSET=factor` command. The scaling of the c_{HB} coefficient by the given `factor` value globally will affect the hydrogen bonding energy. All microscopic hydrogen bonding energy contributions will be scaled by this factor. Second, it is possible to scale the hydrogen bonding prefactor c_{HB} in an element specific way using the `XHBSET={i`$_{\text{element}}$ `factor}` command, where `i`$_{\text{element}}$ is the element symbol or element number of the atom to which the HB-donor-hydrogen atom is attached. This means that the `XHBSET` option allows for the individual scaling of hydrogen bonds of different bonding type (e.g. C-H···Acceptor type hydrogen bonds, which typically are weaker than O-H···Acceptor type hydrogen bonds, can be scaled independently using the `XHBSET` option). It is possible to provide several `XHBSET={i`$_{\text{element}}$ `factor}` commands to the global command input lines at the same time, thus accounting for several different hydrogen bonding types. Third, it is possible to scale the hydrogen bonding threshold σ_{HB} in an element specific way using the `SHBSET={i`$_{\text{element}}$ `factor}` command, where `i`$_{\text{element}}$ is the element symbol or element number of either the atom to which the hydrogen bonding donor hydrogen atom is attached, or the element of the hydrogen bonding acceptor atom. This means that the `SHBSET` option allows for the individual enabling/disabling of hydrogen bonds of different Donor-H···Acceptor bonding types. It is possible to provide several `SHBSET={i`$_{\text{element}}$ `factor}` commands to the global command input lines at the same time. Fourth, it is possible to scale the hydrogen bonding prefactor c_{HB} of atomic compounds (i.e. compounds such as the chloride-anion Cl^-) in an element specific way using the `AHBSET={i`$_{\text{element}}$ `factor}` command, where `i`$_{\text{element}}$ is the element symbol or element number of the atomic compound. It is possible to provide several `AHBSET={i`$_{\text{element}}$ `factor}` commands to the global command input lines at the same time, thus accounting for several different atomic compounds. Please note that currently no other element than hydrogen is allowed to be a donor in a hydrogen bonding interaction. However sometimes atomic cations (i.e. compounds such as the sodium-cation Na$^+$) are seen as hydrogen-like donors as well. The usage of atomic cations as hydrogen bond like donor molecule can be enabled with the global `SPHCATION` command. If thus enabled, the hydrogen bonding donor strength of the atomic cations can be scaled by the `AHBSET={i`$_{\mathtt{element}}$ `factor}` command. In COSMO*therm* the hydrogen bonding prefactor c_{HB} is considered to be temperature dependent[112]. The temperature dependence of c_{HB} is accounted for in the functional form of eq. 5.11-3.

$$c_{HB}(T) = c_{HB}\,\frac{T\ln\left[\left(w_{HB}+\exp\left(-d_{HB}/RT\right)\right)/\left(w_{HB}+1\right)\right]}{T_{Room}\left[\left(w_{HB}+\exp\left(-d_{HB}/RT_{Room}\right)\right)/\left(w_{HB}+1\right)\right]} \tag{5.11-3}$$

This functional form is derived from a plausible physical assumption about the energy gain and the entropy loss during the formation of a hydrogen bond[112]. It holds two parameters w_{HB} and d_{HB} which can be scaled via the `WHBSET=factor` and `DHBSET=factor` commands.

3) The van der Waals (vdW) energy contribution to COSMO-RS is given in equation 5.11-4. Therein a_{eff} is the effective contact area, c_{vdW} and τ_{vdW} are element specific adjustable parameters.

$$E_{vdW}(\sigma_i,\sigma_j) = a_{eff}c_{vdW}\left(\sigma_i,\sigma_j\right) = a_{eff}\left(\tau_{vdW}(i)+\tau_{vdW}(j)\right) \tag{5.11-4}$$

The vdW energy is dependent only on the element type of the atoms that are involved in surface contact. The vdW energy contribution can be modified in terms of scaling the element-element interaction vdW energy parameter c_{vdW}. This is possible with the `VDWSET={i`$_{\mathtt{element}}$ `j`$_{\mathtt{element}}$ `factor}` command. where `i`$_{\mathtt{element}}$ and `j`$_{\mathtt{element}}$ are the element symbols or element numbers of the atoms that are having a vdW contact. This means that the `VDWSET` option allows for the individual scaling of the interaction energy of specific element-element vdW contacts. It is possible to provide several `VDWSET={i`$_{\mathtt{element}}$ `j`$_{\mathtt{element}}$ `factor}` commands to the global command input lines at the same time, thus accounting for several different vdW interaction types. In COSMO*therm* the van der Waals parameter c_{vdW} is considered to be temperature dependent, too[112]. The temperature dependence of c_{vdW} is accounted for in the functional form of eq. 5.11-5.

$$c_{vdW}(T) = c_{vdW}\,\frac{T\ln\left[\left(w_{vdW}+\exp\left(-d_{vdW}/RT\right)\right)/\left(w_{vdW}+1\right)\right]}{T_{Room}\left[\left(w_{vdW}+\exp\left(-d_{vdW}/RT_{Room}\right)\right)/\left(w_{vdW}+1\right)\right]} \tag{5.11-5}$$

This functional form is derived from a plausible physical assumption about the energy gain and the entropy loss during the formation of a van der Waals contact[112]. It holds two parameters w_{vdw} and d_{vdW} which can be scaled via the `WVDWSET=factor` and `DVDWSET=factor` commands.

---

[112] A. Klamt, "*COSMO-RS, From Quantum Chemistry to Fluid Phase Thermodynamics and Drug Design*", Elsevier Science Ltd., Amsterdam, The Netherlands (2005), ISBN: 0-444-51994-7.

4) The chemical potential of the ideal gas phase μ_{Gas}, as used in the vapor pressure related property predictions of COSMO*therm* is given in equation 5.11.6, where E^i_{Gas} and E^i_{COSMO} are the quantum chemical energies of the molecule in the gas phase and in the COSMO conductor. The remaining contributions consist of a correction term for ring shaped molecules with n^i_{Ring} being the number of ring atoms in the molecule and ω_{Ring} an adjustable parameter as well as parameter η_{Gas} providing the link between the reference states of the system's free energy in the gas phase and in the liquid thus describing the entropy loss that occurs in the transition from gas to liquid state.

$$\mu_i^{Gas} = E^i_{Gas} - E^i_{COSMO} - \omega_{Ring} n^i_{Ring} + \eta_{Gas} \tag{5.11-6}$$

The gas phase chemical potential related parameters can be modified in terms of scaling the general gas phase parameter η_{Gas}. This is possible with the `ETASET=factor` command, which globally scales η_{Gas}.by the `factor` value. In addition, the entropic ring contribution parameter ω_{Ring}.can be modified with the `OMRSET=factor` command, which globally scales ω_{Ring}.by the `factor` value. Note that modification of η_{Gas} or ω_{Ring} only affect the gas phase related properties, such as vapor pressure, Henry law constant, and heat of solvation, but not the liquid state properties such as activity coefficient. The temperature dependence of of η_{Gas} and ω_{Ring} is accounted for in the simple functional form of eq. 5.11.7.

$$c_{Gas}(T) = c_{Gas} - \tau_{c_{Gas}} + \tau_{c_{Gas}} T / T_{Room} \tag{5.11.7}$$

The temperature dependency parameters τ_η and τ_ω can be scaled via the `TETSET=factor` and `TORSET=factor` commands.

5) Miscellaneous options: The thresholds of the symmetry detection, as used in the conformer equilibration and the `musym` option can be modified with the `SYMCSET=factor` command for the symmetry detection of the geometries read from COSMO files and with the `SYMGSET=factor` command for the symmetry detection of the geometries read from gas phase energy files. Please note that the symmetry scaling factors affect the symmetry detection thresholds in a general way, i.e. all the different thresholds present in the symmetry detection are scaled by the same given factor.

Input options for the fine tuning of COSMO*therm* parameters:

vdwset= {$i_{element}$ $j_{element}$ factor}	Global input option: element specifically scale van der Waals interaction parameter c_{vdW} with factor. The argument $i_{element}$ and $j_{element}$ are the element symbols or element numbers of the atoms undergoing van der Waals interaction. They are expected to be integer numbers. Argument factor is expected to be a real number.
wvdwset=factor	Global input option: scale van der Waals temperature dependency parameter w_{vdw} with factor. The argument factor is expected to be a real number.
dvdwset=factor	Global input option: scale van der Waals temperature dependency parameter d_{vdW} with factor. The argument factor is expected to be a real number.
etaset=factor	Global input option: scale general gas phase parameter η_{Gas}. with factor. Argument factor is expected to be a real number.
omrset=factor	Global input option: scale gas phase ring correction parameter ω_{Ring}. with factor. Argument factor is expected to be a real number.
tetset=factor	Global input option: scale general gas phase temperature dependency parameter τ_{η}. with factor. Argument factor is expected to be a real number.
torset=factor	Global input option: scale gas phase ring correction temperature dependency parameter τ_{ω} with factor. Argument factor is expected to be a real number.

Input options for the fine tuning of COSMO*therm* parameters, continued:

cmfset=factor	Global input option: scale misfit prefactor α' with factor. The argument factor is expected to be a real number.
chbset=factor	Global input option: scale hydrogen bonding global prefactor c_{HB} with factor. The argument factor is expected to be a real number.
xhbset={$i_{element}$ factor}	Global input option: element specifically scale hydrogen bonding global prefactor c_{HB} with factor. The argument $i_{element}$ is the element symbol or element number of the atom to which the hydrogen bonding donor hydrogen atom is attached. It is expected to be an integer number. Argument factor is expected to be a real number.
shbset={$i_{element}$ factor}	Global input option: element specifically scale hydrogen bonding threshold parameter σ_{HB} with factor. The argument $i_{element}$ is the element symbol or element number of either the atom to which the hydrogen bonding donor hydrogen atom is attached, or of the hydrogen bond acceptor atom. It is expected to be an integer number. Argument factor is expected to be a real number.
ahbset={$i_{element}$ factor}	Global input option: scale the hydrogen bonding prefactor c_{HB} of atomic compounds. The argument $i_{element}$ is the element symbol or element number of the atomic compound. It is expected to be an integer number. Argument factor is expected to be a real number.
sphcation	Global input option: Allow hydrogen bonding donor capacity for atomic cation compounds.
whbset=factor	Global input option: scale hydrogen bonding temperature dependency parameter w_{HB} with factor. The argument factor is expected to be a real number.
dhbset=factor	Global input option: scale hydrogen bonding temperature dependency parameter d_{HB} with factor. The argument factor is expected to be a real number.
symcset=factor	Global input option: scale COSMO file symmetry detection thresholds with factor. The argument factor is expected to be a real number larger than zero.
symgset=factor	Global input option: scale gas phase energy file symmetry detection thresholds with factor. The argument factor is expected to be a real number larger than zero.

4.12 COSMO*mic*

COSMO*mic* is a plugin option of COSMO*therm* and its graphical user interface COSMO*therm*X. COSMO*mic* models surfactant micelles or bio-membranes as inhomogeneous, layered liquids, allowing the calculation of membrane partition coefficients and free energy profiles of solutes in a micelle or a membrane. Details of the methodology are described in Ref[113]: Further particulars of the methodology as well as as some practical guidelines for the use of COSMO*mic* can be found in the COSMO*mic* documentation that is available in the COSMO*therm*X graphical user interface. Please note that the COSMO*mic* plugin is not available in default COSMO*therm*. It needs to be activated via the COSMO*therm* license key `license.txt`.

Within a COSMO*therm* input file the COSMO*mic* plugin has to be called in two steps. First, a micelle definition file needs to be read in the global input section via command `rmic=name.mic` (see section 2.1 of this manual). The micelle definition file (file extension `.mic`) holds all relevant information about the micelle, i.e. the compound information (cosmo-filenames and file paths) of the components that the micelle consists of and, tabulated grid information about the layers that form the micelle. Note that only one micelle definition file can be read in a COSMO*therm* input file, i.e. only one micelle definition can be used per input file. Micelle definition file can be created by the COSMO*therm*X graphical user interface. A recipe for the micelle import procedure in COSMO*therm*X is given in the COSMO*mic* documentation. Because all information about the micelle - including cosmo-file and search path information of the components the micelle is built of – is in the micelle definition file, all additional compounds that are read in the compound input section of the COSMO*therm* input file are considered as solutes in the framework of the COSMO*mic* calculation. The second piece of input required for a COSMO*mic* calculation is the actual call of the micelle as a pseudo-solvent for the given solutes. This call is done with the `x_pure=MICELLE` keyword in the temperature/mixture input lines of the COSMO*therm* input file (see section 2.3 of this manual). Thus a COSMO*mic* input closely resembles a simple mixture input with a pure compound solvent (only temperature `tc`, `tk`, or `tf` and `x_pure` keywords are required), with the difference that the argument of `x_pure` does not point to a given solvent but to the `MICELLE` definition that was read in by the `rmic=name.mic` option. It is possible to run several COSMO*mic* jobs in one COSMO*therm* input file (e.g. COSMO*mic* jobs at different temperatures). It is also possible to run other COSMO*therm* jobs in the same COSMO*therm* input file where the COSMO*mic* input is given. As noted above the only restriction is that only one micelle definition per input file can be used. An example input for COSMO*mic* is given in Example 5.12-1, keywords relevant to COSMO*mic* are highlighted. Further information on this topic can be found in the separate COSMO*mic* manual.

---

[113] A. Klamt, U. Huniar, S. Spycher, and J. Keldenich, *J. Phys. Chem. B* **112** (2008) 12148-12157.

`Example 5.12-1:` COSMOmic input with DMPC micelle and solute methanol

```
ctd=BP_TZVP_C30_1701.ctd cdir=../CTDATA-FILES
ldir=../../licensefiles RMIC=COSMOmic-dmpc.mic
! COSMOmic with DMPC micelle
f = methanol_c0  fdir=../DATABASE-COSMO/BP-TZVP-COSMO # Compound input: methanol
tc=25.00 x_pure=1 CTAB    # Mixture computation in solvent compound 1 (methanol)
tc=25.00 x_pure=MICELLE   # COSMOmic computation with solvent = DMPC micelle
```

5 Appendix A: Publications

5.1 A.1 COSMO-RS / COSMO*therm* related publications

Review Articles[114]

"COSMO-RS: From Quantum Chemistry to Fluid Phase Thermodynamics and Drug Design", Andreas Klamt, Elsevier Science Ltd., Amsterdam, The Netherlands (2005), ISBN: 0-444-51994-7.

- This book provides an extensive and detailed overview over the COSMO-RS/COSMO*therm* methodology and its various fields of application.

"Fast Solvent Screening via Quantum Chemistry: The COSMO-RS approach" Frank Eckert and and Andreas Klamt, *AIChE Journal*, **48**, 369-385 (2002).

- This article aimed especially at the chemical engineering audience presents an introduction to the COSMO-RS / COSMO*therm* methodology and compares it to other methods used in that field (i.e. group contribution methods such as UNIFAC). In addition a number of typical applications of industrial relevance is presented. This review article is possibly the best starting point for interested readers from the chemical engineering / industrial chemistry community.

"COSMO-RS: A novel way from Quantum Chemistry to Free Energy, Solubility and General QSAR-Descriptors for Partitioning" Andreas Klamt and Frank Eckert, book chapter in *Rational Approaches to Drug Design*, H.-D. Höltje and W. Sippl, Editors (Prous Science, S.A., Barcelona, 2001), pp. 195 - 205.

- This article aimed especially at the life science and molecular modeling audience presents an introduction to the COSMO-RS / COSMO*therm* methodology and compares it to other methods used in that field (i.e. QSAR / QSPR, MM / MD-methods). The article concentrates on the prediction of properties relevant to industrial life science applications such as solubility and other QSAR / QSPR descriptors. This review article is possibly the best starting point for interested readers from the biochemistry / life science or environmental chemistry community.

---

[114] An updated list of publications related to COSMO and COSMO*therm* can be found on the COSMO*logic* website at `http://www.cosmologic.de/theory/scientific-papers.html`.

COSMO-RS / COSMO*therm* Theory and Application[114]

"Prediction of acidity in acetonitrile solution with COSMO-RS", Frank Eckert, Ivo Leito, Ivari Kaljurand, Agnes Kütt, Andreas Klamt, Michael Diedenhofen, *Journal of Computational Chemistry* **30**, 799-810 (2009).

- Presents COSMOtherm methodology for the prediction of acid pK_a dissociation constants in nonaqueous solvent acetonitrile. Article-DOI: 10.1002/jcc.21103.

A Klamt, BJ Smith in "Molecular Drug Properties: Measurement and Prediction; Methods and Principles in Medicinal Chemistry", R. Mannhold, H. Kubinyi , G. Folkers (Series Editors), Wiley, 2008

- This book chapter describes the theory and the inherent problems of drug solubility prediction. It is shown that COSMO-RS has a systematic advantage in predicting the non-linearity in drug solubility, but that other aspects, as the unresolved problem of the prediction of ΔGfus , are giving rise to a substantial noise level.

"Prediction, fine tuning, and temperature extrapolation of a vapor liquid equilibrium using COSMOtherm", Andreas Klamt, and Frank Eckert, *Fluid Phase Equilibria* **260**, 183-189 (2007).

- Presents a novel approach for fine-tuning and adjustment of COSMOtherm predictions to experimental data. Article-DOI: 10.1016/j.fluid.2007.07.055.

"Prediction of Solubility with COSMO-RS", Frank Eckert, in *Developments and Applications in Solubility*, Trevor Letcher, (Ed.), The Royal Society of Chemistry, UK (2006), ISBN: 0-85404-372-1, ISBN-13 978 0 85404 372 9.

- This book chapter presents the COSMOtherm methodology for and application of the prediction of the solubility of neutral compounds and organic salts in water and non-aqueous solvents.

"Prediction of Blood-Brain Partitioning and Human Serum Albumin Binding Based on COSMO-RS ?-Moments", Karin Wichmann, Michael Diedenhofen, and Andreas Klamt, *Journal of Chemical Information and Modeling* **47**, 228 - 233 (2007).

- Presents QSPR models for blood-brain partitioning (logBB) and human serum albumin binding (logK-(HSA)) of neutral molecules on the basis of COSMO-RS sigma-moment descriptors. Article-DOI: 10.1021/ci600385w.

"Accurate prediction of basicity in aqueous solution with COSMO-RS", Frank Eckert and Andreas Klamt, *Journal of Computational Chemistry* **27**, 11-19 (2006).

- Presents COSMOtherm methodology for and application of the prediction of pK_a basicity. Abstract, Article-DOI: 10.1002/jcc.20309.

"Prediction of Halocarbon Thermodynamics with COSMO-RS" Frank Eckert and Andreas Klamt, *Fluid Phase Equilibria*, **210**, 117-141 (2003). DOI: 10.1016/S0378-3812(03)00166-3.

- This article presents COSMOtherm applications to the prediction of thermodynamic properties of hydrofluoro- hydrochloro- and mixed halocarbon compounds.

"Prediction of Infinite Dilution Activity Coefficients Using COSMO-RS" R. Putnam, R. Taylor, A. Klamt, F. Eckert and M. Schiller, *Industrial and Engineering Chemistry Research* **42**, 3635-3641 (2003). DOI: 10.1021/ie020974v.

- This article presents COSMOtherm prediction of infinite dilution activity coefficients of organic compounds in variuos solvents and compares the results to UNIFAC predictions. A detailed analysis of the COSMO-RS and UNIFAC results is given.

"Prediction of Infinite Dilution Activity Coefficients of Organic Compounds in Ionic Liquids Using COSMO-RS" Michael Diedenhofen, Frank Eckert and Andreas Klamt, *Journal of Chemical and Engineering Data*, **48**, 475-479 (2003). DOI: 10.1021/je025626e.

- This article presents COSMOtherm applications to the prediction of organic compound properties in ionic liquid solvents.

"Prediction of the mutual solubility of hydrocarbons and water with COSMO-RS" Andreas Klamt, *Fluid Phase Equilibria*, **206**, 223-235 (2003). DOI: 10.1016/S0378-3812(02)00322-9.

- This article presents COSMOtherm applications to the prediction and analysis of hydrocarbon - water solubility and activity coefficients.

"Prediction Of Soil Sorption Coefficients With A Conductor-Like Screening Model For Real Solvents" Andreas Klamt, Frank Eckert and Michael Diedenhofen, *Environmental Toxicology and Chemistry*, **21**, 2562-2566 (2002).

- This article presents the application of COSMO*therm* to the prediction of the soil sorption coefficients $\log K_{OC}$ of over 500 compounds of varying chemical functionality. COSMO*therm* results are compared to correlations of $\log K_{OC}$ with experimental octanol – water partition coefficients.

"Prediction of Aqueous Solubility of Drugs and Pesticides with COSMO-RS" Andreas Klamt, Frank Eckert, Martin Hornig, Michael E. Beck and Thorsten Bürger, *Journal of Computational Chemistry*, **23**, 275-281 (2002). DOI: 10.1002 / jcc.1168.

- This article presents the application of COSMO*therm* to the prediction of the water solubility of drug-like substances and pesticides. COSMO*therm* results are compared to predictions from the HQSAR method.

"Use of COSMO-RS for the Prediction of Adsorption Equilibria" Carsten Mehler, Andreas Klamt and Wolfgang Peukert *AICHE Journal*. **48**, 1093-1099 (2002).

- This article presents COSMO*therm* applications to the correlation and prediction of adsorption equilibria.

" Was darf der Verfahrensingenieur von COSMO-RS erwarten?" Robert Franke, Jörg Krissmann and Ralf Janowsky *Chemie Ingenieur Technik*. **74**, 85-89 (2002).

- This article presents a critical evaluation and validation of the COSMO*therm* method by a research group independent from COSMO*logic* GmbH & Co KG. Applications to vapor-liquid equilibria and liquid-liquid equilibria of industrial relevance are presented. Note: Article is in german language.

"Validation of the COSMO-RS method: Six Binary Systems" Frank Eckert and Andreas Klamt, *Industrial and Engineering Chemistry Research*, **40**, 2371-2378 (2001). DOI: 10.1021 / ie0009132.

- This article presents COSMO*therm* applications to vapor-liquid equilibria (VLE) and liquid-liquid equilibria (LLE) of industrial relevance and compares them to the results obtained with the group contribution method UNIFAC.

"COSMO-RS: a novel view to physiological solvation and partition questions" Andreas Klamt, Frank Eckert and Martin Hornig, *Journal of Computer-Aided Molecular Design*, **15**, 355-365 (2001).

- Presents applications of COSMO-RS to problems of biochemistry / life science.

"COSMO-RS: a novel and efficient method for the a priori prediction of thermophysical data of liquids" Andreas Klamt, and Frank Eckert, *Fluid Phase Equilibria* **172**, 43-72 (2000).

- This article presents COSMO-RS theory and does a methodological comparison of the COSMO-RS theory to group contribution method UNIFAC. In addition some applications are presented.

"Refinement and Parameterization of COSMO-RS" Andreas Klamt, Volker Jonas, Thorsten Bürger and John C. W. Lohrenz *J. Phys. Chem. A*, **102**, 5074-5085 (1998).

- Presents an older version of COSMO-RS theory and shows some applications.

"COSMO and COSMO-RS" Andreas Klamt in *Encyclopedia of Computational Chemistry*, P. v. R. Schleyer and L. Allinger Editors, (Wiley, New York, 1998) pp. 604-615.

- This book chapter in the critically acclaimed "Encyclopedia of Computational Chemistry" presents an overview of the COSMO and COSMO-RS / COSMO*therm* methods and an older implementation of the method into a quantum chemistry program.

"Conductor-like Screening Model for Real Solvents: A New Approach to the Quantitative Calculation of Solvation Phenomena" Andreas Klamt *J. Phys. Chem.*, **99**, 2224-2235 (1995).

- Presents the original COSMO-RS approach and gives a first glimpse of its accuracy and wide applicability to chemical problems in solution.

5.2 A.2 COSMO related publications

COSMO Theory[114]

"COSMO Implementation in TURBOMOLE: Extension of an efficient quantum chemical code towards liquid systems" Ansgar Schäfer, Andreas Klamt Diana Sattel, John C. W. Lohrenz and Frank Eckert *Phys. Chem. Chem. Phys.*, **2**, 2187-2193 (2000).

- Presents the implementation of the refined COSMO into the TURBOMOLE *ab initio* quantum chemistry program and demonstrates its capabilities on some applications.

"First principles implementation of solvent effects without outlying charge error" Kim Baldridge and Andreas Klamt *J. Chem. Phys.*, **106**, 66622-66633 (1997).

- Presents the implementation of the refined COSMO into the *ab initio* quantum chemistry program GAMESS-US.

"Treatment of outlying charge in continuum solvation models" Andreas Klamt and Volker Jonas" *J. Chem. Phys.*, **105**, 9972-9980 (1996).

- Presents some methodological refinements of COSMO, increasing the accuracy and reliability of this model.

"Incorporation of solvent effects into density functional calculations of molecular energies and geometries" Jan Andzelm, Christoph Kölmel and Andreas Klamt *J. Chem. Phys.*, **103**, 9312-9320 (1995).

- Implementation of COSMO into the Density Functional program DMOL3.

"COSMO: A New Approach to Dielectric Screening in Solvents with Explicit Expressions for the Screening Energy and its Gradient" A. Klamt and G. Schüürmann *J. Chem. Soc. Perkin Trans. II*, 799-805 (1993).

- The original presentation of the Conductor-like Screening Model COSMO and its first implementation into the semiempirical quantumchemistry program package MOPAC.

COSMO Applications and Related Articles[114]

"Calculation of UV/Vis Spectra in Solution" Andreas Klamt *J. Phys. Chem.*, **100**, 3349-3353 (1996).

- Application of the COSMO-MOPAC model to the theoretical calculation of UV-Vis spectra.

"Prediction of Phase Equilibria for Binary Mixtures by Molecular Modeling" Maurizio Fermeglia and Sabrina Pricl, *AIChE Journal*, **47**, 2371-2382 (2001).

- A combination of the COSMO model with an equation of state (EOS) is applied to the prediction of binary phase diagrams. Comparisons with results from COSMO*therm* calculations are also given.

C. J. Cramer and D. G. Truhlar, *Chemical Reviews.* **99**, 2161-2200 (1999).

- Review article on continuum solvation models including COSMO. A methodological comparison as well as a large number of applications are presented.

C. J. Cramer and D. G. Truhlar in *Reviews in Computational Chemistry*, Volume IV, K. B. Lipkowitz and D. Boyd Editors. (VCH, New York, 1995) pp. 1-72.

- Older review article on continuum solvation models including COSMO. A methodological comparison as well as a some applications are presented.

6 Index