# ONETEP solvent and electrolyte model

Jacek Dziedzic, James C. Womack, Arihant Bhandari & Gabriel Bramley

July 2021

**This manual pertains to ONETEP versions v6.0.0 and later**.
For older versions, see separate documentation on the ONETEP website.

**Major changes relative to v6.0.0:**

- **Soft-sphere model added in v6.1.1.8**
- **Surface Accessible Volume added in v6.1.3.0**
- **Conjugate gradient solver added in v6.1.3.6**

**WARNING to users of v6.1.3.0 and later.**
The method used to calculate the surface area of the dielectric cavity was changed in version 6.1.3.0. The surface area is used to calculate the $\Delta G_{\mathrm{npol}}$ component of the solvation. The new method is more mathematically consistent, but gives approximately 20% smaller values for the surface area. By default, we use the new method, which means the value of $\Delta G_{\mathrm{solv}}$ and may not agree with earlier versions. If you need full compatibility with versions before 6.1.3, set `is_apolar_sasa_definition` to `isodensity`.

# Contents

# 1 Overview of capabilities

ONETEP implements the Minimal Parameter Solvent Model (MPSM), first presented in Ref. 1. A more detailed description, including guidelines on the choice of parameters, is given in Ref. 2. MPSM offers a very accurate treatment of the polar (electrostatic) solvation contribution, and a rather simple, yet still accurate, treatment of the apolar terms: cavitation, dispersion and repulsion. MPSM is based on the Fattebert and Gygi model (later extended by Scherlis *et al.*) [3].

Implicit solvation calculations in ONETEP can be performed in open boundary conditions (OBC) and in periodic boundary conditions (PBC) alike, with OBC assumed by default. The solute can be immersed in pure solvent (necessitating the solution of the Poisson equation (PE)) or in solvent with electrolyte (leading to the Poisson-Boltzmann equation (PBE)). Solvation energies can be calculated by running a calculation in vacuum first, followed by a calculation in solvent. This can be done either automatically ("auto solvation") or manually. Apart from energy terms due to solvation, ONETEP calculates solvation contributions to forces. It is thus possible to perform in-solvent geometry optimisation and molecular dynamics (with difficulty). Additional solvent exclusion regions can be specified to keep the solvent from predefined regions of the simulation cell.

# 2 The model

ONETEP includes solvation effects by defining a smooth dielectric cavity around the solute ("solute cavity"). In contrast to PCM-based approaches, the transition from a dielectric permittivity of 1 (in the immediate vicinity of the solute) to the bulk value of the solvent is smooth, rather than discontinuous. Thus, there is no "cavity surface", strictly speaking, but rather a thin region of space where the transition takes place. The cavity and the transition are defined by a simple function relating the dielectric permittivity at a point, $\epsilon(\vec{r})$, to the electronic density there, $n(\vec{r})$, yielding

an isodensity model. ONETEP offers two modes of operation – one, where $n(\vec{r})$ is the *current* electronic density ("the self-consistently updated cavity mode"), and another one, where $n(\vec{r})$ is *fixed*, usually to the converged in-vacuum density ("the fixed cavity mode").

The dielectric function $\epsilon(\cdot)$, defined in Ref. 3, Eq. 7, and in Ref. 1, Eq. 1 depends on three parameters: $\epsilon_\infty$, the bulk permittivity of the solvent; $\rho_0$, the electronic density threshold, where the transition takes place; and $\beta$, which controls the steepness of the change in $\epsilon$. The Poisson equation is then solved to obtain the potential due to the molecular density in the nonhomogeneous dielectric. A more general case, where electrolyte ions can be added to the solvent (specified via concentrations), requires solving the Poisson-Boltzmann equation.

## 2.1 Solute cavity

In the MPSM model the dielectric cavity is determined wholly by the electronic density, freeing the model of any per-species parameters (such as van der Waals radii). Thus, the cavity will change shape every time the electronic density changes. From the physical point of view this is good, since it means the density can respond self-consistently to the polarisation of the dielectric and vice versa. From the computational point of view this is rather inconvenient, because it requires extra terms in the energy gradients (see e.g. Ref. 3, Eqs. 5 and 14). Because these terms tend to vary rapidly over very localised regions of space, their accurate calculation usually requires unreasonably fine grids and becomes prohibitively difficult for larger molecules.

### 2.1.1 Fixed cavity

One workaround for the above problem, which is straightforward, but introduces an approximation, consists in *fixing* the cavity and not allowing it to change shape. This is realised by performing an in-vacuum calculation first, then restarting the solvated calculation from the converged in-vacuum density, and using this density to generate the cavity that then remains fixed for the duration of the solvated calculation. Both calculations are self-consistent (in the DFT sense), only the cavity is *not* self-consistently updated in the in-solvent calculation.

How good is this approximation? From experience, it yields solvation energies within several percent of the accurate, self-consistent calculation, cf. Ref. 1. More specifically, the error in solvation energy is expected to be less than 3-4% percent for charged species and less than 1% for neutral species.

If you can spare the computational resources, it would be good to test it on a representative molecule, by comparing the solvation energy against a calculation with a self-consistently updated cavity.

As the cavity remains fixed, the difficult extra terms no longer need to be calculated, and the memory and CPU requirements are significantly reduced (because the grid does not need to be made finer). It is thus the recommended solution. The fixed cavity mode is activated by `is_dielectric_model FIX_INITIAL`, which is the default setting.

### 2.1.2 Self-consistently updated cavity

If one insists on performing calculations with the solute cavity self-consistently responding to changes in density (as in Ref. 3), this can be achieved by `is_dielectric_model SELF_CONSISTENT`. As mentioned earlier, this is costly, because it almost always requires grids that are finer than the default. The relevant grid ("fine grid") can be made finer by `fine_grid_scale` ***n***, with $n > 2$ (which is the default). Typically one would use 3, you might be able to get away with 2.5, you might need 3.5 or even more. The memory and CPU cost increase with the *cube* of this value, so, for instance, when using `fine_grid_scale 3.5` one would expect the computational cost to increase by a factor of $(3.5/2)^3 \approx 5.36$.

Even when using much finer grids, the additional gradient term due to the self-consistently updated cavity poses numerical difficulties. This is especially true if the changes in the density are rapid. For this reason, even if it is technically possible to run a calculation in solvent *without* a preceding calculation in vacuum, it is not recommended to do so – the initial, dramatic changes in the density will likely prove problematic. It will be much easier to run an in-vacuum calculation to convergence, and to restart a calculation in solvent from there. The auto solvation functionality (see section 3.3) makes this easy.

### 2.1.3 Soft Sphere Cavity Model

In addition to MPSM, the soft sphere cavity model of *Fisicaro et al.* (Ref. 4) has been implemented to provide a dielectric cavity function closer to the standard per-species parametrisation models. This feature is especially useful when the system under study requires significantly different solvation radii for its constituent species. This contrasts with MPSM, which applies the parameter controlling the dielectric cavity shape (the isodensity contour) globally, which leads to the dielectric cavity being too large/small for particular species for a single input isodensity value.

The dielectric cavity within the soft sphere model is composed of a set of interlocking, atom-centered spheres with radii assigned to each atom. Much like MPSM, the dielectric function for each atom varies smoothly from vacuum to bulk permitivity. The dielectric functions themselves are defined by: i) the soft sphere radius set by default by Alverez's database of van der Waals' radii (Ref. 5) or manually set in the `is_soft_sphere_radii` block. The default radii can be uniformly scaled using `is_soft_sphere_scale`. ii) The steepness of the transition from vacuum to bulk

permitivity is controlled by `is_soft_sphere_delta`. To activate the soft sphere cavity model, set `is_dielectric_function` to 'soft_sphere'. By default, `is_soft_sphere_scale` is set to 1.33 and `is_soft_sphere_delta` to 0.5, as determined by minimizing the error of the solvation free energy against empirical data for a set of small neutral, organic molecules. These cavity radii may not give accurate solvation energies for heavier elements/system types, and it is encouraged to perform further parametrization to minimize error with respect to selected experimental data.

Forces for the soft sphere cavity model are not yet implemented, so only single point energy calculations are supported.

## 2.2 Apolar terms: cavitation energy

MPSM includes the apolar cavitation term in the solvent-accessible surface-area (SASA) approximation, thus assuming the cavitation energy to be proportional to the surface area of the cavity, the constant of proportionality being the (actual physical) surface tension of the solvent, $\gamma$, and the constant term being zero. The cavitation energy term is calculated and added automatically, unless `is_include_apolar F` is explicitly stated. Surface tension of the solvent has to be specified (otherwise the default for water near room temperature (about 0.074 N/m) will be used). This can be done using `is_solvent_surf_tension`. Keep in mind that the apolar term is *scaled by default* to account for dispersion and repulsion (see section 2.3). The scaling is controlled by `is_apolar_scaling_factor`, and the default is *not* unity.

## 2.3 Apolar terms: dispersion-repulsion energy

ONETEP's model allows for the solute-solvent dispersion-repulsion apolar energy term to be modeled approximately. This greatly improves the quality of obtained solvation energies for uncharged molecules, particularly so if they are large. This term is reasonably approximated with the same SASA approach that is used for cavitation, albeit with a smaller, and negative, prefactor. In practice this is most easily achieved by simply scaling the cavitation term down by a constant multiplicative factor. A good scaling factor is 0.281705, which is what our model uses by default (see Ref. 1 for justification). The keyword controlling this parameter is `is_apolar_scaling_factor` (with the above default), and its argument is a unitless value.

## 2.4 Apolar terms: solvent sccessible volume (SAV)

The accuracy of the implicit solvent model can be further improved by adding the surface-accessible volume (SAV) to the nonpolar energy term:

$$\Delta G_{npol} = \tau * \gamma S + pV \tag{1}$$

where $\gamma$ is the physical surface tension (Section 2.2), $\tau$ is the apolar scaling factor tuned by `is_apolar_scaling_factor` in the SASA model (Section 2.3), and $p$ is the solvent pressure. This method is activated by setting `is_apolar_method` to 'SAV'. We note that the scaling factors $\tau$, $p$, and (in the case of soft sphere) $f$, must be tuned to give accurate free energies of solvation compared to the original SASA model. By minimising the mean absolute error (MAE) of $\Delta G_{solv}$ with respect to experiment for a small set of neutral molecules, we found the optimum scaling factors for water ($\gamma = 0.07415\ Nm^{-1}$) are:

- Soft Sphere: $f = 1.20$, $\tau = 0.813$ and $p = -0.35\ GPa$
- MPSM: $\rho_0$=0.00035, $\tau = 0.684$ and $p = -0.35\ GPa$

Currently, these values are only fully optimised for the soft sphere implicit solvent model, but the values provided for the MPSM provide a starting estimate. We note that in this simple model, $p$ does not correspond to the physical pressure of the solvent and acts as a fitting parameter to give optimum values of $\Delta G_{solv}$, meaning it can assume negative values.

Furthermore, if fixed PAOs are used in place of optimised NGWFs, the optimum parameters change significantly. For the QZP basis, best parameters for the soft sphere model and MPSM are:

- Soft Sphere: $f = 1.21$, $\tau = 0.861$ and $p = -0.35\ GPa$
- MPSM: $\rho_0 = 0.00035$, $\tau = 0.785$ and $p = -0.35\ GPa$

Larger or smaller fixed PAO basis sets may require slightly different optimal parameters given the above values were calculated with the QZP basis only.

In summary:

- **polar, cavitation, dispersion and repulstion terms (SASA)**:
  `is_include_apolar T` (default)
  `is_apolar_method SASA` (default)
  `is_apolar_scaling_factor 0.281705` (default)
- **polar, cavitation, dispersion and repulstion terms (SAV)**:
  `is_include_apolar T` (default)
  `is_apolar_method SAV`
  `is_soft_sphere_scaling_factor 1.20` (optimised for soft sphere)
  `is_apolar_scaling_factor 0.813` (optimised for soft sphere)
  `is_solvent_pressure -0.35 GPa` (optimised for soft sphere)
- **polar and cavitation terms only**:
  `is_include_apolar T` (default)
  `is_apolar_scaling_factor 1.0`
- **polar term only**:
  `is_include_apolar F`

# 3 Practicalities

## 3.1 DL_MG solver

ONETEP uses a multigrid solver to solve the Poisson or Poisson-Boltzmann equation. Currently this is done by interfacing to a solver called DL_MG [6, 7]. DL_MG is distributed with ONETEP and is compiled in by default. If your version does not include DL_MG your calculation will stop with a descriptive error message.

Solving the P(B)E is a memory- and time-consuming process, and you should expect solvation calculations to take about 2-3 times longer compared to standard ONETEP (also remembering that you will likely have to run two calculations per result – one in vacuum, and one in solvent). The memory requirement of the solver grows linearly with the volume of the system, meaning that padding with vacuum or with bulk solvent is not free, in contrast to calculations not employing the multigrid solver.

The solver uses a multigrid approach to solve the P(B)E to second order. To ensure the high-order accuracy necessary for solvation calculations, the solver then applies a high-order defect correction technique, which iteratively corrects the initial solution to a higher order. Consult Ref. 2, 7 for more information on the defect correction approach used in DL_MG.

## 3.2 Grid sizes

### 3.2.1 Under OBC

One limitation of DL_MG is that the grid sizes it uses are not created equal. Good grid sizes are divisible many times into grids twice as small. For example a grid with 161 points (and so 160 grid-edges in between them) is an excellent choice, since it divides into two grids with 81 points (160 splits into two 80's), these divide into two grids with 41 points, which in turn divide into two grids with 21 points, which divide into two grids with 11 points and so on. This lets the solver use many multigrid levels, increasing efficiency. For contrast, consider a grid with 174 points (and so 173 grid-edges). 173 is prime, and this grid cannot be subdivided at all, making it a poor choice.

Knowing about these limitations, ONETEP will sometimes slightly reduce (truncate) your fine grid dimensions when passing data to and from the multigrid solver. This truncation always affects the right-hand side of the grid, and by default between 1 and 7 grid lengths will be truncated, to give DL_MG enough flexibility. This is done automatically, and you will be informed about the details like this:

```
ONETEP fine grid is 126 x 126 x 126 gridpoints, 29.0000 x 29.0000 x 29.0000 bohr.
FD multigrid is     121 x 121 x 121 gridpoints, 27.8492 x 27.8492 x 27.8492 bohr.
```

Here, ONETEP discarded three slabs, each just over 1 $a_0$ thick, from your system, at the highest values of $x$, $y$, and $z$.

Even though this is done automatically, it is your responsibility to ensure that nothing of significance (read: any charge density) is in the margin that is thrown away. If any of your NGWFs extend into the margin, your calculation will be meaningless (and will likely stop with an error). Due to *Fourier ringing*, tails of very small, but nonzero charge density extend in all Cartesian directions from your system, even outside the localisation spheres of the NGWFs. It is thus good practice to pad your system with a little vacuum in all directions, say 10 $a_0$. This is in addition to the margin lost due to truncation.

### 3.2.2 Under PBC

Under PBC, the grid used by the multigrid solver must have the same dimensions as the simulation cell. This is necessary to ensure that the solution from the solver has the correct periodicity. The approach of truncating the grid (section 3.2.1) to obtain a grid which satisfies the grid size constraints of the multigrid solver cannot therefore be used in periodic BCs. Instead, an appropriately sized grid for use by the multigrid solver is obtained by *scaling* ONETEP's fine grid, changing the number and spacing of grid points, while maintaining the same physical dimensions.

This corresponds to slightly increasing the scale factor for the fine grid (used, among other things, for multigrid operations) with respect to the standard grid (determined by the kinetic energy cutoff) along each coordinate direction to ensure that the dimensions of the fine grid satisfy the requirements of the solver (see Ref. 6, 7 for details about these requirements).

This is done automatically, and you will be informed about the details like this:

```
Grid scale modified to satisfy multigrid solver grid constraints
         Grid scale values after modification:         2.09  2.09  2.00
*******************************************************************************
[...]
ONETEP fine grid is 136 x 136 x 240 gridpoints, 32.4219 x 32.7579 x 60.0000 bohr.
FD multigrid is     136 x 136 x 240 gridpoints, 32.4219 x 32.7579 x 60.0000 bohr.
```

Here, the grid was scaled by a factor of 2.09 along the $x$ and $y$ coordinates, and no scaling was necessary for the $z$ coordinate. The two grids (ONETEP fine grid and the grid seen by DL_MG) are identical.

Changing the fine grid scale factor causes ONETEP to use the modified fine grid throughout the calculation (not only when invoking the multigrid solver). This has the unfortunate consequence that ONETEP must perform additional work to interpolate and filter between the fine grid and a slightly smaller "double grid", which is used during other parts of a ONETEP calculation. Normally this is avoided by making the fine and double grids the same size, but is no longer possible when the fine grid is modified for multigrid operations in PBCs.

## 3.3 Auto solvation

An in-solvent calculation is almost universally preceded by a calculation in vacuum. In the fixed cavity mode this is necessary to generate the cavity from a converged in-vacuum calculation. In the self-consistently updated cavity mode this helps mitigate stability issues associated with the cavity updates (cf. 2.1.2). To make the procedure easier for users, ONETEP provides what is known as "auto solvation" – a mode of operation, where the two calculations (in vacuum and in solvent) are automatically run in sequence.

To enable auto solvation (which is *off* by default), use `is_auto_solvation T`. This will automatically run an in-vacuum calculation, followed by a calculation in solvent. Some input parameters might have to be adjusted along the way, but this will happen automatically and you will always be informed when this happens. Once the calculation in solvent completes, a detailed breakdown of the energies will be printed. It will look something like this:

```
Individual components of total energy in solvent:     hartree           kcal/mol
- Usual non-electrostatic DFT terms:       -26.28930636174560      -16496.788451
- Electrostatic fixed charge energy:         3.05443104938460        1916.684380
- Apolar cavitation energy:                  0.02080496905999          13.055315
- Apolar dispersion-repulsion energy:       -0.01495721238146          -9.385792
 ------------------------------------------------------------------------------
- Total energy in solvent:                 -23.22902755568246      -14576.434548
```

The above shows a breakdown of the total energy in solvent into the usual DFT terms (except for electrostatic energy), the electrostatic energy, the apolar cavitation energy and the apolar dispersion-repulsion energy.

```
Components of total energy in solvent:                hartree           kcal/mol
- Usual non-electrostatic DFT terms:       -26.28930636174560      -16496.788451
- Electrostatic energy:                      3.05443104938460        1916.684380
- Apolar energy terms:                       0.00584775667854           3.669523
 ------------------------------------------------------------------------------
- Total energy in solvent:                 -23.22902755568246      -14576.434548
```

In the above all the apolar terms have been summed together for convenience.

```
Calculation of free energy of solvation:              hartree           kcal/mol
- Total energy in solvent:          (+)     -23.22902755568246      -14576.434548
- Total energy in vacuum:           (-)     -23.20990671966879      -14564.436043
 ------------------------------------------------------------------------------
- Total free energy of solvation:           -0.01912083601367         -11.998505
```

The above is a direct calculation of the free energy of solvation as a difference of the in-solvent and in-vacuum energies.

```
Components of polar term in f.e. of solvation:        hartree           kcal/mol
- Electrostatic:                            -0.06759943752720         -42.419287
- Change in nonelectrostatic DFT terms:      0.04263084483500          26.751258
 ------------------------------------------------------------------------------
- Polar term in f.e. of solvation:          -0.02496859269221         -15.668028
```

The above is the calculation of the polar term to solvation, as a sum of the change in electrostatic energy between in-solvent and in-vacuum and the change in the remaining DFT terms.

```
Components of free energy of solvation:                hartree           kcal/mol
- Polar term in f.e. of solvation: (+)       -0.02496859269221         -15.668028
- Apolar (cavitation, dis., rep.): (+)        0.00584775667854           3.669523
 ------------------------------------------------------------------------------
- Total free energy of solvation:            -0.01912083601367         -11.998505
```

Finally, the total free energy of solvation is calculated as the sum of the polar and apolar terms calculated earlier. This is usually what you are after.

Auto solvation relies on restart files to achieve a seamless transition from the calculation in vacuum to the calculation in solvent. A `.vacuum_dkn` and a `.vacuum_tightbox_ngwfs` file will be written to disk once the calculation in vacuum is completed (and also earlier, if you used `write_denskern T` and/or `write_tightbox_ngwfs T`). These files are then read at the beginning of the calculation in solvent. This makes restarting in-solvent geometry optimisation and molecular dynamics runs very tricky – this is not recommended in practice. Please ensure such calculations run to completion without manual restarts.

## 3.4 Manual solvation and restarts

Occasionally you might want to run a calculation in solvent without automatically running a calculation in vacuum first. Perhaps you already have the calculation in vacuum and you prefer to manually restart it in solvent. This is known as "manual solvation". To activate it, use `is_implicit_solvent T` (the default is `F`), and make sure to have `is_auto_solvation F` (which is the default).

Make sure you know how the solute cavity is generated in this case. If this is a fresh calculation (not a restart), the cavity will be generated from the initial guess density. This is probably not what you want. In the fixed cavity mode, this will mean that you will be stuck with a cavity that is not very realistic (coming from a guess). In the self-consistently updated cavity mode, the cavity will adapt to the subsequent changes to the density, but the initial, dramatic changes might make this numerically unstable. Therefore, restarting from a converged in-vacuum run is recommended instead.

If you ran an in-vacuum calculation to convergence earlier and you have the requisite restart files, you can add `read_denskern T` and `read_tightbox_ngwfs T` to your input to effect a restart. ONETEP will look for a `.dkn` and a `.tightbox_ngwfs` file. The cavity will be constructed from the density generated from these files, and the calculation will also proceed from this DKN and NGWFs. If the in-vacuum calculation you ran earlier was a part of an auto-solvation calculation, you will need to rename or link the `.vacuum_dkn` and `.vacuum_tightbox_ngwfs` files to their `.dkn` and `.tightbox_ngwfs` counterparts.

Finally, if you want to restart an in-solvent calculation from an unfinished in-solvent calculation, you have to be careful. This is because you want the calculation to continue from the partially-converged in-solvent density, while still constructing the cavity from the converged in-vacuum density. To do this, use the `is_separate_restart_files` keyword. Setting it to `T` (the default is `F`) will instruct ONETEP to construct the solute cavity from the `.vacuum_dkn` and `.vacuum_tightbox_ngwfs` files, while the density for continuing the calculation will be generated from the `.dkn` and `.tightbox_ngwfs` files.

## 3.5 Solvation in PBC

Implicit solvation operates under OBC by default. However, ONETEP allows solvation calculations in PBC, with some caveats. Only fully periodic BCs are supported, i.e. where the system is periodic along all simulation cell directions. Support for mixed BCs (where OBC are applied along some directions and PBC along others) is planned for the future, but is not currently supported. If you intend to solvate slabs, surfaces or wires, you would probably be best off using PBC and suitable padding.

Boundary conditions can be specified individually for the multigrid solver, local pseudopotential, ion-ion interaction and the smeared ion representation using the following keywords:

- `multigrid_bc`,
- `pspot_bc`,
- `ion_ion_bc`,
- `smeared_ion_bc`.

Each of these keywords accepts a string which should contain three characters (which may be separated by spaces), specifying the BCs along the $x$, $y$ and $z$ directions of the simulation cell. For `multigrid_bc` the characters may be `O`, `P` or `Z`, corresponding to open (Coulombic), periodic and zero BCs, respectively. "Zero" BCs are open BCs, but with the potential set to zero at the boundary, rather than approximately computed. For `pspot_bc`, `ion_ion_bc` and `smeared_ion_bc`, the values can be `O` or `P`, defined as for `multigrid_bc`.

These keywords allow for flexible selection of mixtures of open and periodic BCs, but currently only fully open and fully periodic BCs are supported, corresponding to values of `O O O` and `P P P` (and `Z Z Z` to use zero BCs in the multigrid solver).

### 3.5.1 Key points on using the BC keywords

- If `multigrid_bc` is set in an input file, but the implicit solvent model is not activated (e.g. if other solvent model keywords are not used) then the multigrid solver is used to compute the Hartree potential in vacuum, without the smeared ion representation.
- Setting `smeared_ion_bc` is insufficient to activate the smeared ion representation—you must also set `is_smeared_ion_rep` (or use the full solvent model, e.g. via `is_implicit_solvent`).
- If BCs are not explicitly set using `multigrid_bc` and the multigrid solver is activated (for example, by setting `is_implicit_solvent: T`), then the BCs for the multigrid solver default to fully open BCs.
- If BCs are not explicitly set using `pspot_bc` then the BCs for the local pseudopotential are determined by the type of calculation being performed and should respect previous defaults. Setting `openbc_pspot: T` will set fully open BCs, as will setting `is_implicit_solvent: T` or `is_smeared_ion_rep: T`. In vacuum (without smeared ions) the local pseudopotential defaults to fully periodic BCs, unless the cutoff Coulomb approach is used, in which case the BCs are determined by the value of the `coulomb_cutoff_type` keyword.
- If BCs are not explicitly set using `ion_ion_bc` then the BCs for the ion-ion interaction are determined by the type of calculation being performed and should respect previous defaults. Setting `openbc_ion_ion: T` will set fully open BCs, as will setting `is_implicit_solvent: T` or `is_smeared_ion_rep: T`. In vacuum (without smeared ions) the ion-ion interaction defaults to fully periodic BCs, but this can be changed (as normal) by using the cutoff Coulomb or Martyna-Tuckerman approaches.
- If `smeared_ion_bc` is not explicitly set, then the BCs used for smeared ions are the same as those used for the multigrid solver (with the exception that zero BCs for the multigrid solver are converted to open BCs for smeared ions).
- It is possible to specify inconsistent BCs for different interaction terms. A warning should be output if this is detected, but care is necessary to avoid unphysical results.

In short: the boundary conditions selected by default in previous versions of ONETEP should be respected if the new keywords are not explicitly set. If the keywords are set, then care must be taken to ensure that they are set consistently in order to obtain physically realistic results. An effort has been made to prevent inconsistencies between the setting of the new keywords for controlling BCs and earlier keywords (such as `openbc_hartree`, `openbc_pspot` and `openbc_ion_ion`), but this has not been extensively tested.

## 3.6 Smeared ions

The P(B)E is almost always solved for the molecular (total) density, because we are interested in how the solvent polarises in response to the total (valence electronic + core) charge density. The solution is the molecular potential, and not the electronic potential. To reconcile this with the usual DFT way of thinking in terms of valence-electronic and core densities and potentials separately (which is needed e.g. in the calculation of the NGWF gradient), a numerical trick known as the smeared-ion formalism is used. In this formalism ionic cores are modelled by narrow positive Gaussian distributions and the usual energy terms are re-cast (cf. Ref. 2, Appendix):

- the usual Hartree energy is now replaced by the "molecular Hartree energy" (also called electrostatic energy), that is, the electrostatic energy of the molecule's total charge distribution in the potential this charge distribution generates, in the presence of dielectric;
- the local pseudopotential energy is corrected by an extra term that takes the smeared-ion nature of the cores into account;
- a self-interaction correction term is added to the total energy to account for the added Gaussian distributions (each of them self-interacts). This term does not depend on the electronic degrees of freedom, but depends on the ionic positions;
- a non-self-interaction correction term is added to the total energy to account for the added Gaussian distributions (they interact with each other). This term does not depend on the electronic degrees of freedom, but depends on the ionic positions.

In principle, the total energy of the system is unchanged by the application of the smeared-ion formalism, however, due to minor numerical inaccuracies some discrepancies may be observed. These cancel out when calculating energy differences between solvated and *in vacuo* systems, **provided the smeared-ion formalism is used for the vacuum calculation as well**. There is one parameter to the smeared-ion formalism, $\sigma$, which controls the width of the Gaussians placed on the ions. See Ref. 2 for more details on the choice of this parameter. The default value is almost always OK.

The key takeaway message here is that you need to use smeared ions in **both** the in-vacuum calculation and the in-solvent calculation to ensure the energy expressions are comparable. To do that, add `is_smeared_ion_rep T` to your input file(s). If you forget about this in a solvation calculation (`is_implicit_solvent T`) or if you do auto solvation (`is_auto_solvation T`) it will be added automatically for you, but a warning will be produced. However, if you run manual solvation, you need to remember to include `is_smeared_ion_rep T` in the in-vacuum calculation – ONETEP has no way of knowing you will follow this with an in-solvent calculation.

## 3.7 Forces

If you ask ONETEP to calculate forces, the force terms due to implicit solvent will be automatically calculated and included. The formulas employed are exact (to numerical accuracy) when a self-consistently updated cavity is used. For the case of a fixed cavity, they are approximate. The approximation is very good, but initial tests suggest that you might not be able to converge geometries to typical thresholds – although the noise in the forces will be small, it might be enough close to equilibrium to throw off the geometry optimiser. Keep this in mind.

You should be able to do geometry optimisation and molecular dynamics without any problems with implicit solvent, provided that you use `is_auto_solvation T`. Note that restarting these might be tricky if they are interrupted during the in-solvent stage – you will need to ensure the correct restart files (the vacuum restart files) are used to generate the solvent cavity upon restart, cf. section 3.4.

Smeared-ion forces in vacuum are also implemented. These are numerically exact and practically negligible.

## 3.8 Exclusion regions

This functionality enables excluding regions of space from the solvent. Any excluded region has its dielectric permittivity set to exactly 1, similarly to what happens in core regions (cf. `is_core_width`). This is useful for removing pockets of solvent that could otherwise appear in buried cavities, which are inaccessible to the solvent, yet the electronic density there is low enough to generate a dielectric with a permittivity notably larger than 1.

The regions are specified in a `%block is_dielectric_exclusions`, which looks like this:

```
%block is_dielectric_exclusions
sphere 20.0 22.0 18.0 4.0             ! x, y, z of centre; r (all in a0)
box 13.0 16.0  20.5 29.0  13.0 15.0   ! xmin xmax  ymin ymax  zmin zmax (all in a0)
xcyl 18.4 20.7 7.0                    ! y, z, r (all in a0)
%endblock is_dielectric_exclusions
```

The above excludes the solvent from a sphere centred at $(20, 22, 18)\,a_0$ with a radius of $4\,a_0$, from a box spanning from $(13, 20.5, 13)\,a_0$ to $(16, 29, 15)\,a_0$, and from a cylinder oriented along the X axis, passing through $y = 18.4\,a_0$, $z = 20.7\,a_0$ and a radius of $7\,a_0$. `sphere`, `box`, `xcyl`, `ycyl` and `zcyl` are the only region shapes supported now. All exclusion regions currently assume open boundary conditions **and do not work in PBC**. You can have as many as 10000 regions specified in the exclusion block.

It is crucial to ensure that discontinuities in the permittivity are avoided, because they prevent the solver from converging. Usually, exclusion regions can be chosen such that they merge quite smoothly with regions where the dielectric is naturally 1 (or reasonably close). If this is not possible, then the boundaries of the exclusion regions can be smoothed. This is achieved using a Fermi-Dirac function,

$$\varepsilon(d) = \varepsilon_\infty - \frac{\varepsilon_\infty - 1}{e^{d/d_0} + 1}, \tag{2}$$

where $d$ is the distance to the exclusion region boundary (and is negative if inside the exclusion region), and $d_0$ is the smearing length set by `is_dielectric_exclusions_smear`. By default, this is set to 0 $a_0$, giving hard-walled exclusion regions ($\varepsilon = 1$ inside and $\varepsilon = \varepsilon_\infty$ outside). But if exclusion regions interface directly with solvent regions, it should be chosen to be at least a couple of times larger than the multigrid spacing, so that the permittivity becomes sufficiently continuous for the solver to converge.

### 3.9 Solvent Polarization

The non-homogeneous Poisson equation:

$$\nabla \cdot (\varepsilon \nabla v) = -4\pi n_{\text{tot}} \tag{3}$$

can be recast in the form of a solvent polarization density:

$$\nabla \cdot \nabla v = -4\pi \left(n_{\text{tot}} + n_{\text{pol}}\right) \tag{4}$$

Subtracting the two, the polarization density is calculated as [8]:

$$n_{\text{pol}} = \frac{1}{4\pi} \nabla \cdot \left[(\varepsilon - 1)\, \nabla v\right] \tag{5}$$

$$n_{\text{pol}} = \frac{1}{4\pi} \left[(\varepsilon - 1)\, \nabla^2 v + \nabla\, (\varepsilon - 1) \cdot \nabla v\right] \tag{6}$$

The polarization potential is calculated by solving the following Poisson eq:

$$\nabla \cdot \nabla v_{\text{pol}} = -4\pi n_{\text{pol}} \tag{7}$$

This is done in properties calculation with `is_solvation_properties T` and 3-D grid data for $n_{\text{pol}}$ and $v_{\text{pol}}$ is output.

## 4 Keywords used in solvation calculations

### 4.1 Basic

- `is_implicit_solvent T/F` – turns on/off the implicit solvent. Default is off. Will be set automatically if auto solvation is used.

- `is_include_apolar T/F` – turns on/off the apolar energy terms. Default is on.
- `is_apolar_sasa_definition density/isodensity` – defines the method used in the difference method which calculates the solvent accessible surface area (SASA) of the dielectric cavity. 'Density' calculates the SASA by varying the electron density, and 'Isodensity' uses varying $\rho_0$ values. 'Density' is the recommended setting unless backwards compatibility with old versions is desired. Warning can be suppressed by defining this keyword. Default is 'Density'. Only affects MPSM. Use 'Density' or 'Isodensity' to suppress warnings. Default is 'Default'.
- `is_apolar_method SASA/SAV` – sets the definition of the cavitation term in terms of surface area or surface area with volume. Default is SASA.
- `is_apolar_scaling_factor` $x$ – controls the scaling of the apolar term with the aim of taking solute-solvent dispersion-repulsion into account. The default is 0.281075.
- `is_smeared_ion_rep T/F` – turns on/off the smeared-ion representation. Default is off, but if ONETEP detects you're running a solvation calculation, it will turn it on for you and let you off with a warning. When comparing results of two calculations (e.g. results in vacuum and in solvent), always ensure this is set identically in both calculations.
- `is_density_threshold` $x$ – sets the MPSM model parameter $\rho_0$ to $x$ (atomic units). The default is 0.00035, as per Ref. 1.
- `is_solvation_beta` $x$ – sets the MPSM model parameter $\beta$ to $x$ (no unit). The default is 1.3, as per Ref. 1.
- `is_bulk_permittivity` $x$ – sets the physical constant – solvent bulk permittivity $\epsilon_\infty$ to $x$ (no unit). The default is 78.54 (suitable for water near room temperature and pressure and at low frequencies) if implicit solvent is on, and 1.0 is implicit solvent is off.
- `is_solvent_surf_tension` $x$ – sets the physical constant – solvent surface tension $\gamma$ to $x$ (unit must be supplied). The default is 0.07415 N/m (which is suitable for water near room temperature).
- `is_solvent_pressure` $x$ – sets the pressure used to calculate the SAV contribution to the apolar term. Does not correspond to physical water pressure and is optimised to obtain minimal errors with respect to experimental free energies of solvation. Default is -0.35 GPa (which is suitable for water near room temperature).
- `is_dielectric_model FIX_INITIAL/SELF_CONSISTENT` – picks either the fixed cavity or the self-consistently updated cavity, as described in section 2.1.
- `is_auto_solvation` $x$ – automatically runs an in-vacuum calculation before any solvation calculation, thus relieving the user from the burden of manually restarting calculations. This

attempts to automatically control the directives for restarting, running two calculations (vacuum and solvated) in succession. Using this directive is a must when doing implicit-solvent geometry optimisation, implicit-solvent molecular dynamics, implicit-solvent transition state search or implicit-solvent forcetest. This directive is compatible with conduction calculations.

- `is_dielectric_function FGF/SOFT_SPHERE` – Defines the function used to create dielectric cavity. Switches between the charge density based MPSM and the atomic radius based soft sphere model.
- `is_soft_sphere_scale` $x$ – Scales the default Alvarez vdW radii provided in ONETEP. The default is 1.33. This does not apply to radii defined in the `is_soft_sphere_radii` block.
- `is_soft_sphere_delta` $x$ – Controls the steepness of the transition from vacuum to the bulk permitivity value. This applies to both default radii and those specified in the `is_soft_sphere_radii` block. The default is 0.5.
- `is_soft_sphere_radii` – Block sets the soft sphere radii for species defined. These values are unaffected by the scaling factor defined in `is_soft_sphere_scale`. Undefined species will use the default values defined by Alvarez vdW (Ref. 5). Units bohr. e.g.

```
%block is_soft_sphere_radii
Li 2.5
Pt 4.6
%endblock is_soft_sphere_radii
```

## 4.2 Advanced

The default settings usually work fine and the advanced settings should only be changed if you know what you're doing.

- `is_bc_coarseness` $x$ – changes the size of the blocks into which charge is coarsened when boundary conditions are calculated. The default is 5. Smaller values may subtly increase accuracy, but will incur a computational cost that grows as $x^{-3}$. This can be perfectly acceptable for smaller molecules. For larger molecules (1000 atoms and more) use 7 or more to reduce computational cost. For the effect of this parameter on accuracy, cf. Ref. 2.
- `is_bc_surface_coarseness` $x$ – changes the size of the surface blocks onto which charge is interpolated when boundary conditions are calculated. The default is 1 and is recommended. Larger values will improve computational cost (that grows as $x^{-2}$), but may decrease accuracy, especially for charged molecules. If the calculation of BCs becomes a bottleneck, prefer tweaking `is_bc_coarseness` $x$ instead.

- `is_bc_allow_frac_charge T/F` (new in v6.1.1.28) – when set to `T`, the calculation of boundary conditions for the multigrid solver will not check if the coarse-grained charge is close to an integer. This can be used in rare cases where you know this is not going to be a problem. The default is `F`.

- `is_separate_restart_files T/F` – allows the set of restart files used to construct the solute cavity in solvent to be distinct from the set of restart files used to construct the initial density. This is useful if you need to restart a solvated calculation, but still want to construct the cavity from the converged vacuum density, and not the partially-converged solvated density. See section 3.4.

- `is_solvation_properties T/F`– when set to `T` it will produce scalarfields of quantities relevant in solvation during a properties calculation. This is useful for visualising potentials, densities, Boltzmann ion concentrations, electrolyte accessibilities, etc. Ensure you supplied `dx_format T` and/or `cube_format T` and/or `grd_format T`.

- `is_smeared_ion_width` $x$ – sets the width of the smeared-ion Gaussians, $\sigma$, to $x$ (in units you supply). The default is 0.8 $a_0$ and should be OK for most calculations. Results should not depend on this parameter, but only if it's within rather narrow limits of sensibility. Too high values (anything larger than 1.0, roughly) are seriously unphysical, as they will lead to cores whose Gaussian tails stick out of the electronic density, especially for hydrogen atoms. This is very bad, since it does not change the energy *in vacuo* (the effect of the smearing, regardless of $\sigma$, is cancelled by the correction terms to energy), but changes the energy in solution (by polarising the solvent differently – in reality the cores are screened by the electrons). Too low values (anything smaller than 0.6, roughly), on the other hand, will lead to Gaussians so thin and tall that they will become very difficult for the multigrid solver to treat, requiring high orders and unreasonably fine grids to obtain multigrid convergence. See Ref. 2 for more details.

- `fine_grid_scale` $x$ – makes the ONETEP fine grid $x$ (no unit) times as fine as the coarse grid, $x$ does not have to be an integer. The solution of the P(B)E and associated finite-difference operations are performed on the fine grid (or its subset, for OBC). Increasing `fine_grid_scale` allows making this grid finer without unnecessarily increasing the kinetic energy cutoff of the calculation. The default is 2. Memory and computational effort increase with the cube of $x$.

- `is_dielectric_exclusions_smear` $x$ – sets the smearing for dielectric exclusion regions to $x$ (in the units you supply). See section 3.8.

## 4.3 Fine control over DL_MG

These keywords enable fine control over the behaviour of the DL_MG solver. See Ref. 6, 7 for more details, particularly regarding convergence control.

- `mg_use_cg T/F` (new in v6.3.1.6) – Turns on the conjugate gradient solver. This generally increases the stability of the solver, but is likely to reduce performance. It might be useful to turn this on if you have problems converging difficult cases – particularly in Poisson-Boltzmann solvation.

- `mg_use_error_damping T/F` – can be used to turn on/off error damping in the defect correction procedure. This is often necessary when solving the full (non-linearised) Poisson-Boltzmann equation, but will likely not do much for the linearised Poisson-Boltzmann equation or for the Poisson equation. Accordingly, the default depends on `is_pbe` and is `F` for `is_pbe NONE` and `is_pbe LINEARISED`, and `T` for `is_pbe FULL`.

- `mg_continue_on_error T/F` – if `T`, instructs the multigrid solver not to abort if a solution to the P(B)E cannot be converged to desired tolerances, and instead to return an underconverged solution. This can be useful for particularly stubborn cases, especially in Boltzmann solvation. Default is `F` when solving the Poisson equation and `T` if solving the Poisson-Boltzmann equation. If you want to turn it on for Boltzmann solvation, you will very likely need to increase `is_pbe_energy_tolerance` by a very large amount.

- `mg_defco_fd_order` $x$ – sets the discretization order used when solving the P(B)E to $x$ (no unit). Available values are 2, 4, 6, 8, 10 and 12, the default is 8. With 2 no defect correction is performed. Values of 4 and above employ defect correction. The lowest values (2 and 4) are not recommended, because they offer poor accuracy. Generally the largest value (12) will offer best accuracy, but this has to be weighed against a likely drop in performance (higher orders often take longer) and possibility of Gibbs-like phenomena that may occur when high orders are used with steeply-changing dielectric permittivity, as is the case for larger values of $\beta$. 8 or 10 is a good starting value. Results should not depend on the choice of this parameter, but performance and multigrid convergence will. See the troubleshooting section below for details. See Ref. 2, 7 for more details.

- `mg_max_iters_vcycle` $x$ – sets the maximum number of multigrid V-cycle iterations to $x$ (no unit). The default is 50. See Ref. 7 for a description of the solver, including the V-cycle scheme employed.

- `mg_max_iters_defco` $x$ – sets the maximum number of high-order defect correction iterations to $x$ (no unit). The default is 30. See Ref. 7 for a description of the solver, including the defect correction procedure.

- `mg_max_iters_newton` $x$ – sets the maximum number of Newton method iterations to $x$ (no unit). The default is 30. This is only relevant when solving the non-linear PBE. See Ref. 7 for a description of the inexact-Newton method employed by the solver in this scenario.
- `mg_max_iters_cg` $x$ (new in v6.3.1.6) – sets the maximum number of iterations for conjugate gradients to $x$ (no unit). The default is 50. This is only relevant when `mg_use_cg` is `T`.
- `mg_max_res_ratio` $x$ – sets the threshold for the consecutive residual ratio which determines when the multigrid solver gives up (positive real value, no unit, the default is 0.999). This should not require tuning.
- `mg_vcyc_smoother_iter_pre` $x$ – sets the number of V-cycle smoother iterations pre-smoothing (integer, no unit, the default is 2). Difficult systems, particularly in PBCs, might benefit from an increase of this value to 4 or 8.
- `mg_vcyc_smoother_iter_post` $x$ – sets the number of V-cycle smoother iterations post-smoothing (integer, no unit, the default is 1). Difficult systems, particularly in PBCs, might benefit from an increase of this value to 4 or 8.
- `mg_tol_res_rel` $x$ – Set the relative tolerance in the norm of the residual for the defect correction procedure to $x$ (no units, the default is 1.0e-2).
- `mg_tol_res_abs` $x$ – Set the absolute tolerance in the norm of the residual for the defect correction procedure to $x$ (atomic units, the default is 5.0e-2).
- `mg_tol_pot_rel` $x$ – Set the relative tolerance in the norm of the potential for the defect correction procedure to $x$ (no units, the default is 1.0e-6).
- `mg_tol_pot_abs` $x$ – Set the absolute tolerance in the norm of the potential for the defect correction procedure to $x$ (atomic units, the default is 1.0e-6).
- `mg_tol_vcyc_rel` $x$ – Set the relative tolerance for the norm of the residual in multigrid V-cycle iterations to $x$ (no units, the default is 1.0e-8).
- `mg_tol_vcyc_abs` $x$ – Set the absolute tolerance for the norm of the residual in multigrid V-cycle iterations to $x$ (atomic units, the default is 1.0e-5).
- `mg_tol_newton_rel` $x$ – Set the relative tolerance for the norm of the residual in Newton method iterations to $x$ (only applies when solving the nonlinear PBE, no units, the default is 1.0e-8).
- `mg_tol_newton_abs` $x$ – Set the absolute tolerance for the norm of the residual in Newton method iterations to $x$ (only applies when solving the nonlinear PBE, atomic units, the default is 1.0e-5).

- `mg_tol_cg_res_rel` $x$ (new in v6.3.1.6) – Set the relative tolerance in the norm of the residual for the conjugate gradients to $x$ (no units, the default is 1.0e-2). This is only relevant when `mg_use_cg` is `T`.
- `mg_tol_cg_res_abs` $x$ (new in v6.3.1.6) – Set the absolute tolerance in the norm of the residual for the conjugate gradients to $x$ (atomic units, the default is 5.0e-2). This is only relevant when `mg_use_cg` is `T`.

## 4.4 Expert

These will only be listed here and not discussed. The last three keywords are discussed in a separate document devoted to the real space local pseudopotential (see ONETEP website).

- `mg_granularity_power`,
- `is_surface_thickness`,
- `is_bc_threshold`,
- `is_core_width`,
- `is_check_solv_energy_grad`,
- `openbc_pspot_finetune_nptsx`,
- `openbc_pspot_finetune_f`,
- `openbc_pspot_finetune_alpha`.

# 5 Boltzmann solvation (solute with electrolyte)

ONETEP has the ability to perform Poisson-Boltzmann implicit solvent calculations, that is, to include electrolyte in the implicit solvent. The electrolyte is represented by point particles ("Boltzmann ions"), which interact with one another only in the mean-field sense, and affect the reaction field, providing a rudimentary model of screening. The model is described in Ref. 9 and Ref. 10. Users would be well-advised to read these first.

Boltzmann solvation calculations in ONETEP can be performed in OBC and in PBC alike. In PBC care must be taken to suitably neutralise the simulation cell so that the electrostatic energy does not diverge. ONETEP offers a number of schemes to achieve this, including a novel NECS scheme – see `is_pbe_neutralisation_scheme` in section 5.1 and carefully read Ref. 10.

The inclusion of the electrolyte leads to the well-known nonlinear Poisson-Boltzmann equation. ONETEP (or rather DL_MG) can solve this equation as is, or the linearised approximation can be used – see `is_pbe` in section 5.1.

Because Boltzmann ions are point particles, they tend to concentrate in the immediate vicinity of the solute, often reaching unphysical concentrations. A number of ways have been proposed to address this problem. ONETEP implements a steric potential approach to keep the Boltzmann ions sufficiently far from the solute – see Ref. 9 and `is_pbe_steric_pot_type` in section 5.1 for a description.

In the presence of the electrolyte a number of additional terms appear in the grand potential. These are clearly listed in the output if auto solvation is used:

```
Individual components of total energy in solvent:     hartree           kcal/mol
- Usual non-electrostatic DFT terms:       -26.31256445391999      -16511.383124
- Electrostatic fixed charge energy:         3.11407548141888        1954.111825
- Electrostatic mobile charge energy:       -0.00000610048106          -0.003828
- Accessibility (steric) correction:         0.00000440501188           0.002764
- Osmotic pressure contribution:            -0.00045050433991          -0.282696
- Ionic atmosph. rearrangement entropy:     -0.00082199171218          -0.515808
- Chemical potential contribution:           0.00082978766243           0.520700
- Apolar cavitation energy:                  0.02130850899275          13.371291
- Apolar dispersion-repulsion energy:       -0.01531921982762          -9.612955
 ---------------------------------------------------------------------------------
- Total energy in solvent:                -23.19294408719482      -14553.791830
```

Similarly, the calculation of the free energy of solvation will include additional terms due to the electrolyte:

```
Components of free energy of solvation:              hartree           kcal/mol
- Polar term in f.e. of solvation: (+)      -0.04757925126662         -29.856430
- Apolar (cavitation, dis., rep.): (+)       0.00598928916514           3.758336
- Non-es. electrolyte terms:       (+)      -0.00044440385885          -0.278868
- Energy of pure electrolyte:      (-)      -0.00048258128208          -0.302824
 ------------------------------------------------------------------------------
- Total free energy of solvation:           -0.04154568419718         -26.070310
```

If you chose not to use auto solvation, you will have to rely on the ENERGY COMPONENTS table to find the individual terms, while the energy of pure electrolyte will be printed out for you at the end of the in-solvent calculation.

## 5.1 Keywords controlling Boltzmann solvation

The following keywords control the Poisson-Boltzmann implicit solvation functionality, which allows performing calculations in implicit solvent containing electrolyte represented by Boltzmann ions.

- `is_pbe NONE/LINEARISED/FULL` – chooses the equation to be solved in implicit solvation. `NONE` chooses the (generalised) Poisson equation, which corresponds to solvation in the absence of an electrolyte. `LINEARISED` chooses the linearised Poisson-Boltzmann equation (LPBE), which is a simplified treatment of electrolyte. `FULL` chooses the full, non-linear Poisson-Boltzmann equation (NLPBE), which deals with the electrolyte without the simplifications offered by linearisation. The default is `NONE`.

Except where noted **(*)**, all the below keywords only have an effect if `is_pbe` is *not* `NONE`. Similarly, except where noted **(*)**, all the defaults given below only apply to calculations where `is_pbe` is *not* `NONE`.

- `is_pbe_temperature` $T$ – sets the temperature of the Boltzmann ions to $T$ (in K). The default is 300 K.
- `is_pbe_bc_debye_screening T/F` – includes (`T`) or does not include (`F`) the effect of Debye screening in the calculation of Dirichlet boundary conditions for calculations in solvent. This only has an effect in OBC. With Debye screening an additional multiplicative factor of $\exp(-r/\lambda_{\mathrm{D}})$, where $\lambda_{\mathrm{D}}$ is the Debye length, is included in the boundary conditions. This is exact for LPBE and an approximation in NLPBE. Turning off Debye screening will cause

ONETEP to use BCs that are appropriate for the case of no electrolyte, which will be unphysical. The default is `T`.

- `is_pbe_exp_cap` $c$ – – sets the exponential cap to $c$ (no unit). This is only relevant to `is_pbe FULL`. In solving the NLPBE it is a well-known issue that the exponential factors that appear in certain expressions (e.g. for the Boltzmann ion concentration) are prone to exploding (in the usual floating-point representation) when the value of the argument to the exp function is large. To retain numerical stability, the arguments to the exp function are typically capped, i.e. they are not allowed to exceed a predefined constant. The default in ONETEP is 0.0, which means $c$ is set to the default cap in DL_MG, which is currently 50.0. Specifying any value other than 0.0 will cause ONETEP to discard the default provided by DL_MG and to use the user-specified value.

- `is_pbe_neutralisation_scheme` *scheme* – chooses a specified neutralisation scheme.
  **(*)** This keyword and its defaults can also apply to `is_pbe NONE`. This is only relevant for PBC calculations with non-zero total solute charge. In this scenario the total system charge (solute + electrolyte) must be zero for the electrostatic energy not to diverge. There are many ways of ensuring charge neutrality. ONETEP implements the following:

  - `NONE` – ignores charge neutralisation. This is only meaningful for OBC or when the system is charge-neutral. This is the default in OBC.

  - `JELLIUM` – applies the common jellium neutralisation, shifting the charge density by its negative average, so that the average density is zero. This is the default for PBC with no electrolyte (`is_pbe NONE`).

  - `ACCESSIBLE_JELLIUM` – applies a modified jellium neutralisation (cf. Ref. 10, Sec. 3.3). This is only applicable when `is_pbe` is *not* `NONE`.

  - `COUNTERIONS_AUTO` – applies neutralisation by electrolyte concentration shift (NECS) (cf. Ref. 10, Sec. 3.1) with optimal shift parameters (cf. Ref. 10, Eqs. 14 and 15). This is the default in PBC with electrolyte (`is_pbe LINEARISED` or `is_pbe FULL`).

  - `COUNTERIONS_AUTO_LINEAR` – applies neutralisation by electrolyte concentration shift (NECS) (cf. Ref. 10, Sec. 3.1) with shift parameters derived from a linear approximation (cf. Ref. 10, Eq. 18).

  - `COUNTERIONS_FIXED` – applies neutralisation by electrolyte concentration shift (NECS) (cf. Ref. 10, Sec. 3.1) with shift parameters specified by the user via `%block_sol_ions`.

- `is_pbe_energy_tolerance` $E$ – sets the tolerance for the discrepancy between two expressions for the mean-field contribution to the grand potential to $E$ (in units you supply). The two expressions are (A): Ref. 10, Eq. 5, where individual terms are calculated according to

Eqs. 10, 12, 13, 15, and 16 *except* for the fixed electrostatic term (first term in brackets in Eq. 10), which is excluded here; and (B) Ref. 10, Eq. 31 (for PBC) or Eq. 35 (for OBC) *except* for the fixed electrostatic term (first term in brackets in Eq. 10), which is also excluded here. For `is_pbe FULL` we expect the two expressions to be identical (modulo numerical noise). For `is_pbe LINEARISED` we expect the two expressions to be identical to first order (modulo numerical noise). The check is useful for detecting poorly converged solutions of the PBE. The default is 0.001 kcal/mol for `is_pbe FULL`, and 0.05 kcal/mol for `is_pbe LINEARISED`. Normally you should not need to adjust this parameter. However, it might need to be increased, perhaps dramatically, if you set `mg_continue_on_error T`.

- `is_pbe_steric_pot_type X/H/M/S` – specifies the type of steric potential that will affect Boltzmann ions. This is to prevent them from unphysically concentrating in the immediate vicinity of the solute. The available options are:
  - `X` – no steric potential. This is not recommended, except in contrived test cases. This is currently the default, but this might change later.
  - `H` – hard-core potential (see below). The hard-core potential is infinite within the radial cutoff $r_c$, and zero elsewhere. Numerically this is realised by setting the accessibility to $\gamma = 0$ within the radial cutoff, and to $\gamma = 1$ elsewhere. This choice can pose numerical difficulties because of the infinite steepness, and is not recommended.
  - `M` – smoothed hard-core potential (see below). **This is the recommended choice**. Here the accessibility is defined as $\gamma = \frac{1}{2} + \frac{1}{2}\,\mathrm{erf}\,(\frac{r-r_c}{\sigma})$, with values below $10^{-7}$ then set to $10^{-90}$. The potential, as always, is $-kT \ln \gamma$. Here, $r_c$ is the radial cutoff, $\sigma$ is the smearing parameter.
  - `S` – soft-core potential, that is, a potential of the form $Ar^{-12}\,\mathrm{erf}\,(\alpha r)^{12}$ (see the `is_sc_*` keywords below). This potential does not seem to work well (too soft) and is not recommended.

  For both hard-core steric potentials the radial cutoff $r_c$ is determined as a sum of two components: the solvent radial cutoff $r_c^{\mathrm{solvent}}$, and the solute radial cutoff $r_c^{\mathrm{solute}}$. The solvent radial cutoff is set by the `species_solvent_radius` block, with one value per *solute* (sic!) species. The solute radial cutoff is determined through `is_hc_steric_dens_isovalue` and also depends on the solute species (see below). The solute cutoff is determined for each species separately, by examining the radial valence-electronic density profile coming from the pseudoatomic solver. The radial density is scanned from infinity to zero for a value equal to or larger than $n_0$ (specified via `is_hc_steric_dens_isovalue`). The radial coordinate of this value is taken as $r_c^{\mathrm{solute}}$. If `is_hc_steric_dens_isovalue` is negative, $r_c^{\mathrm{solute}} = 0$. Thus,

the solute radial cutoff is constant, and does depend on the current electronic density, only on the output of the atomic solver.

- `is_hc_steric_dens_isovalue` $n_0$ – sets the density isovalue used to determine $r_c^{\text{solute}}$ to $n_0$ atomic units. The default is 0.003. This only applies to `is_pbe_steric_pot_type H/M`.
- `is_hc_steric_smearing` $\sigma$ – sets the smearing width for the smoothed hard-core cutoff (`is_pbe_steric_pot_type M`) to $\sigma$ (in units you supply). The default is 0.4 $a_0$.
- `is_sc_steric_magnitude` $A$ – sets the magnitude of the soft-core steric potential to $A$ (in units you supply, dimension: energy $\times$ distance$^{12}$). Default is negative, to force users not to forget this parameter. This only applies to `is_pbe_steric_pot_type S`, which you should not be using anyway.
- `is_sc_steric_smoothing_alpha` $\alpha$ – sets the smoothing parameter of the soft-core steric potential to $\alpha$ (in units you supply, dimension: inverse distance). Default is 1.5 $a_0^{-1}$. This only applies to `is_pbe_steric_pot_type S`, which you should not be using anyway.
- `is_sc_steric_cutoff` $r_c$ – sets the radial cutoff for the soft-core steric potential to $r_c$ (in units you supply). Since the range of this potential is technically infinite, we truncate it to zero beyond a specified distance, $r_c$. This only applies to `is_pbe_steric_pot_type S`, which you should not be using anyway.
- `is_steric_write T/F` – if set to `T`, the steric potential and associated accessibility will be written out as scalarfields when initialised. Ensure you supplied `dx_format T` and/or `cube_format T` and/or `grd_format T`.
- `sol_ions` – is a block describing the Boltzmann ions in the system. The format for $n$ Boltzmann ions is as follows:

  `%block sol_ions`
  $ion\_species_1$ $charge_1$ $conc_1$ $x_1$
  $ion\_species_2$ $charge_2$ $conc_2$ $x_2$
  ...
  $ion\_species_n$ $charge_n$ $conc_n$ $x_n$
  `%endblock sol_ions`

  Here, $ion\_species_i$ is the name of the species of Boltzmann ion $i$ (which is irrelevant from the physical point of view), $charge_i$ is the charge on species $i$, $conc_i$ is the concentration of that species (in mol/L), and $x_i$, which is optional, is a NECS shift parameter for species $i$, relevant only for `is_pbe_neutralisation_scheme COUNTERIONS_FIXED`. For example, to define a 1M NaCl electrolyte, you would use:

```
%block sol_ions
Na +1 1.0
Cl +1 1.0
%endblock sol_ions
```

- `species_solvent_radius` – defines the solvent radial cutoff $r_{\mathrm{c}}^{\mathrm{solvent}}$ for every *solute* (sic!) species, as follows:

  `%block species_solvent_radius`
  $species_1$ $r_1$
  $species_2$ $r_2$
  ...
  $species_n$ $r_n$
  `%endblock species_solvent_radius`
  For example, to keep all the Boltzmann ions an extra 3.5 $a_0$ away from your methane solute, you would use:

```
%block species_solvent_radius
C 3.5
H 3.5
%endblock species_solvent_radius
```

# 6 Various hints for a successful start

- Use one of the examples provided on the ONETEP website as a starting point.
- Make sure both your vacuum and solvated calculations use smeared ions.
- Make sure the parameters of both your vacuum and solvated calculations are identical (box sizes, KE cutoffs, `k_zero`, `mg_defco_fd_order`, `is_smeared_ion_width`, `is_bc_coarseness`, `is_bc_surface_coarseness`). Or just use `is_auto_solvation T`.
- Choose `FIX_INITIAL` over `SELF_CONSISTENT` for `is_dielectric_model`.
- Use an `mg_defco_fd_order` of 8 and `is_smeared_ion_width` of 0.8. Specify them explicitly, as the defaults may change in the future.
- Do not mess with expert directives.
- In OBC, have at least about 10 bohr of vacuum/solvent around the edges of your molecule's NGWFs (not atomic positions) on each side of the simulation cell, *after taking the truncation into account* – cf. section 3.2.1.

- Always start your calculation in solution as a restart from a fully converged *in vacuo* calculation. Or just use `is_auto_solvation T`.

# 7 Troubleshooting: Problems, causes and solutions

- **Problem A**: ONETEP crashes (e.g. catching `SIGKILL` or `SIGSEGV`) when evaluating the boundary conditions or solving the P(B)E.
  **Cause (A1)**: You've run out of memory and the OOM killer killed the calculation. Solving the P(B)E often represents the peak memory usage of the calculation.
  **Solution (A1)**: Increase available memory (perhaps by shifting the MPI/OMP balance towards more threads and fewer MPI processes) or decrease box size or decrease grid fineness.
  **Cause (A2)**: You've run out of global stack space. Solving the P(B)E often represents the peak stack usage of the calculation.
  **Solution (A2)**: Increase stack size using `ulimit -s`. Make sure you do that on compute nodes, not the login node. Or, preferably, use `onetep_launcher` and its `-s` parameter.
  **Cause (A3)**: You've run out of per-thread stack space. Solving the P(B)E often represents the peak per-thread stack usage of the calculation.
  **Solution (A3)**: Increase per-thread stack size using `ulimit -s`. Make sure you do that on compute nodes, not the login node. Or, preferably, use `onetep_launcher` and its `-o` parameter.

- **Problem B**: Multigrid calculation does not converge (error message from DL_MG) or converges very slowly (as evidenced by the contents of a log file with a filename ending in `_dl_mg_log.txt`).
  **Cause (B1)**: (Only applies to OBC calculations) Charge is not correctly localized (cell is too small or molecule otherwise too close to cell edge).
  **Solution (B1)**: Check and fix the cell size, paying attention to the margin between the DL_MG grid and fine grid.
  **Cause (B2)**: Dielectric permittivity too steeply changing on the cavity boundary for the current grid size, finite differences struggling to approximate the changes. This is often the culprit if the calculation ran fine *in vacuo* but struggles in solvent.
  **Solution (B2)**: Preferable, but painful, solution is to make the grid finer (`fine_grid_scale`). Otherwise an increase or decrease of discretisation order may help (make sure it stays consistent across your calculations, though). A parameterisation with lower `is_solvation_beta` and `is_density_threshold` will usually help (make sure it stays consistent across your calculations, though).
  **Cause (B3)**: The smearing width is too small, making the smeared cores too thin and tall, which is difficult for the finite differences. This is often the culprit if the calculation also

struggles *in vacuo*.

**Solution (B3)**: Increasing `is_smeared_ion_width` will help (but mind the consequences), if it was too small in the first place. Increasing the discretisation order will help (especially if you've been using less than 10), but might lead to a similar problem (Cause (B2)) in solution.

**Cause (B4)**: Too lax thresholds for convergence of the defect correction in DL_MG.

**Solution (B4)**: To tighten the convergence threshold of the defect correction in DL_MG, adjust the values of `mg_tol_res_rel`, `mg_tol_res_abs`, `mg_tol_pot_rel` and `mg_tol_pot_abs`.

**Cause (B5)**: Too few defect correction iterations in DL_MG.

**Solution (B5)**: To increase the number of defect correction iterations in DL_MG, use `mg_max_iters_defco`, try 200 for good measure.

**Cause (B6)**: Too few smoother iterations in DL_MG, particularly if this is a Boltzmann calculation.

**Solution (B6)**: Increase the number of smoother iterations in DL_MG to 2 or 4 using `mg_vcyc_smoother_iter_pre` and `mg_vcyc_smoother_iter_post`.

**Cause (B7)**: Too few V-cycle iterations in DL_MG.

**Solution (B7)**: Increase the number of V-cycle iterations in DL_MG using `mg_max_iters_vcyc`, try 200 for good measure.

**Cause (B8)**: Too few Newton iterations in DL_MG. This only applies if you are solving the NLPBE in Boltzmann solvation.

**Solution (B8)**: Increase the number of Newton iterations using `mg_max_iters_newton`, try 100 for good measure.

**Cause (B9)**: Problem is too difficult for the solver – e.g. grids are not fine enough, the dielectric cavity has a steep boundary (usually happens when underconverged densities are used to generate it), Boltzmann-ionic concentrations changing too steeply, etc.

**Solution (B9)**: Try using the conjugate gradient approach – add `mg_use_cg T` to your input file. This is only available in versions v6.1.3.6 and newer.

- **Problem C**: Calculation struggles to converge LNV or NGWFs or does not converge at all. RMS gradient stalls.

  **Cause (C1)**: If you're using `is_dielectric_model SELF_CONSISTENT`, then this is normal, unless your grid is ridiculously fine (you will need `psinc_spacing 0.5` and `fine_grid_scale 3` or better, as a rule of thumb).

  **Solution (C1)**: Use `is_dielectric_model FIX_INITIAL` if possible. If you are sure you need `is_dielectric_model SELF_CONSISTENT`, make the grid finer and have a lot of memory.

  **Cause (C2)**: Density kernel is not converged enough.

  **Solution (C2)**: `Try minit_lnv 6` and `maxit_lnv 6` (for smaller molecules) or `minit_lnv 10` and `maxit_lnv 10` (for large molecules).

# 8 Frequently asked questions

## 8.1 What are the values for the model parameters?

Two sets of values will be proposed here. The first one will be called "high-beta" parameterisation. It offers the best quality (in terms of r.m.s. error from experiment) for both charged and neutral species. The drawback is that the high value of $\beta$ means the multigrid convergence is poor and it often takes a while to converge. Or it may not converge. This should be your first choice **only** if accuracy trumps anything else. The parameters are:

```
is_solvation_beta 1.6
is_density_threshold 0.00055
```

The second parameterisation, called "low-beta" should pose no problems to the multigrid solver under any circumstances. Quality should be only marginally worse for anions and neutrals and comparable or better for cations. These are the default parameters, and they are:

```
is_solvation_beta 1.3
is_density_threshold 0.00035
```

Both parameterisations assume `is_bulk_permittivity 78.54`, which is suitable for water. It should be noted that the model is deficient in its treatment of anions, consistently underestimating the magnitude of the solvation effect by 10-25%. Work is ongoing to fix this, until then a different parameterisation may be used if one is only interested in anionic species.

## 8.2 Can you do solvents other than water?

Yes, provided you know the dielectric permittivity of the solvent and its surface tension. Accuracy has not been extensively tested, but it should work.

## 8.3 Can you do mixed boundary conditions?

Not yet, but we might in the future.

## 8.4 Is implicit solvation compatible with conduction calculations?

Yes, to the best of our knowledge.

## 8.5 Is implicit solvation compatible with PAW?

Yes, to the best of our knowledge.

# 9 Known issues and untested functionality

- PBC are not yet currently compatible with the `is_dielectric_exclusions` block.
- PBC have not been tested in self-consistent cavity mode—only fixed cavity calculations are guaranteed to work.
- Forces and geometry optimisation have not been tested when using implicit solvent under PBC—only energy calculations are guaranteed to work.

# 10 Contact

General questions about implicit solvation in ONETEP should be directed to Jacek Dziedzic (`J.Dziedzic[-at-]soton.ac.uk`). Questions regarding Boltzmann (electrolyte) solvation should be directed to Arihant Bhandari (`A.Bhandari[-at-]soton.ac.uk`).

# References


[1] J. Dziedzic, H. H. Helal, C.-K. Skylaris, A. A. Mostofi, and M. C Payne, *Minimal parameter implicit solvent model for ab initio electronic-structure calculations*, EPL **95** (2011).

[2] J. Dziedzic, S. J. Fox, T. Fox, C. S. Tautermann, and C.-K. Skylaris, *Large-Scale DFT Calculations in Implicit Solvent – A Case Study on the T4 Lysozyme L99A/M102Q Protein*, International Journal of Quantum Chemistry **113** issue 6 (2013).

[3] D. A. Scherlis, J.-L. Fattebert, F. Gygi, M. Cococcioni, and N. Marzari, *A unified electrostatic and cavitation model for first-principles molecular dynamics in solution*, J. Chem. Phys. **124** (2006).

[4] G. Fisicaro, L. Genovese, O. Andreussi, S. Mandal, N. Nair, N. Marzari and S. Goedecker, *Soft-Sphere Continuum Solvation in Electronic-Structure Calculations*, J. Chem. Theory Comput. **13** (2017).

[5] S. Alvarez *A cartography of the van der Waals territories*, Dalton Trans. **42** (2013).

[6] L. Anton, J. Womack, and J. Dziedzic, *DL_MG multigrid solver* (2020) http://www.dlmg.org

[7] J. C. Womack, L. Anton, J. Dziedzic, P. J. Hasnip, M. I. J. Probert, and C.-K. Skylaris, J. Chem. Theory Comput. **14**, 1412 (2018).

[8] O. Andreussi, I. Dabo and N. Marzari, *Revised self-consistent continuum solvation in electronic-structure calculations*, J. Chem. Phys. **136** (2012).

[9] J. Dziedzic, A. Bhandari, L. Anton, C. Peng, J. C. Womack, M. Famili, D. Kramer, and C.-K. Skylaris, *Practical Approach to Large-Scale Electronic Structure Calculations in Electrolyte Solutions via Continuum-Embedded Linear-Scaling Density Functional Theory*, J. Phys. Chem. C **124** (2020).

[10] A. Bhandari, L. Anton, J. Dziedzic, C. Peng, D. Kramer, and C.-K. Skylaris, *Electronic Structure Calculations in Electrolyte Solutions: Methods for Neutralization of Extended Charged Interfaces*, J. Chem. Phys. **153** (2020).