Implicit solvation in ONETEP

Jacek Dziedzic & James C. Womack

June 2011, last updated July 2020

Unless explicitly specified otherwise, this manual pertains to newer ONETEP versions – v4.4.x, where $x \geq 6$; and v4.5.1.0 and later. These versions feature a number of changes to defaults and input syntax. In particular, since v4.5.13.5 several input keywords have been changed – the keywords from before (version v4.4.6 up to v4.5.13.5) and after (versions $\geq$ v4.5.13.5) these changes are provided. Please make sure to check your version of ONETEP to ensure you are using the correct keywords. If you are using a version of ONETEP older than v4.4.6, please consult "Implicit Solvation (older versions)" instead.

1 The model

For a brief overview of the model used in ONETEP, see [3]. For a more detailed description, including the optimal choice of parameters, see [4].

ONETEP includes solvation effects by defining a smooth dielectric cavity around the solvated molecule. In contrast to PCM-based approaches, the transition from a dielectric permittivity of 1 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, $\epsilon(r)$, to the electronic density, $\rho(r)$, yielding an isodensity model. This function, defined in [1], eq.7, and in [3] eq.1 depends on three parameters – ϵ_∞, the bulk permittivity of the solvent; ρ_0, the electronic density threshold, where the transition takes place and β, which controls the steepness of the change in ϵ. The nonhomogeneous Poisson equation (NPE) is then solved to obtain the potential due to the molecular density in the nonhomogeneous dielectric. A more general case, where salt ions can be added to the solvent (specified via concentrations), requires solving the Poisson-Boltzmann (PB) equation.

1.1 The model: smeared-ions

Because the NPE is solved for the molecular density, yielding the molecular potential, a numerical trick termed the smeared-ion formalism is used to reconcile this with the usual DFT way of thinking in terms of valence-electronic and core densities separately. In this formalism nuclear cores are modelled by narrow positive Gaussian distributions and the usual energy terms are re-cast (cf. [4], Appendix):

- the usual Hartree energy is now replaced by the "molecular Hartree 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, under open boundary conditions;
- 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, σ, which controls the width of the Gaussians placed on the ions. See [4] for more details on the choice of this parameter. The default value is almost always OK.

1.2 The model: open boundary conditions

In the smeared-ion formalism the molecular Hartree energy is not obtained in reciprocal space, like in standard ONETEP calculations, but rather by solving the Poisson equation (homogeneous in vacuum, nonhomogeneous in solution) in real space, under open boundary conditions (BCs). This is another reason why solvated calculations should be compared only with *in vacuo* calculations performed with smeared ions – the boundary conditions need to be consistent for the energy differences to be physically meaningful.

Since the (molecular) Hartree energy is calculated under open boundary conditions, the remaining energy terms must use open BCs for consistency. The core-core energy is evaluated by means of a direct Coulombic sum instead of the Ewald technique. The local pseudopotential term is calculated in real-space rather than in reciprocal space. This happens automatically once smeared ions are turned on. Solvation calculations in ONETEP require no corrections due to periodicity (in contrast to the original model of [1]), as they are natively performed under open BCs. On the other hand, solvation calculations using periodic or mixed boundary conditions are currently not supported in ONETEP (but we're working on it — see section 6.3 for details of new experimental support for fully periodic boundary conditions).

1.3 The model: self-consistently changing cavity

Since the dielectric cavity is determined wholly by the electronic density, it 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 polarization 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. [1] eqs. 5 and 14). Because these terms vary rapidly over very localised regions of space, their accurate calculation requires unreasonably fine grids and becomes prohibitively difficult for larger molecules. On the other hand, neglecting these terms while nevertheless changing the cavity shape in response to the changes in the electronic density results in lack of convergence. Below we outline three proposed solutions to this problem.

1.3.1 Fixed cavity

A solution which is straightforward, but an approximation, consists in **fixing** the cavity and not allowing it to change shape. This is realized by performing an *in vacuo* calculation first, then restarting a solvated calculation from a converged *in vacuo* density. In this way the final *in vacuo* density will be used to generate the cavity, which will remain fixed for the duration of the solvated calculation. This should yield solvation energies within several percent of the accurate, self-consistent calculation, cf. [3]. As the cavity remains fixed, the difficult extra terms no longer need to be calculated. The extra error is small while the memory and CPU requirements are greatly reduced (because the grid does not need to be made finer), thus this is the recommended solution. Note that **this is still a self-consistent process** (the necessary terms are included in the Hamiltonian), only the cavity is kept fixed.

The restarting can be performed in two fashions, either manually or automatically. For the manual approach, ensure you have `write_denskern T` and `write_tightbox_ngwfs T` in the vacuum calculation. Also ensure you have `is_smeared_ion_rep T`, as per 1.1. Once the calculation in vacuum is complete, restart in solvent by adding `read_denskern T`, `read_tightbox_ngwfs T`, and `is_implicit_solvent T`. It's probably a good idea to run this on a copy, or else the restart files from the solvated calculation are going to overwrite the restart files from vacuum. You can also disable the writing out of restart files in solution.

A more elegant solution, is to use `is_auto_solvation T` alongside `is_implicit_solvent T`. This will automatically perform a vacuum calculation, save restart files, initiate a solvated calculation, read the restart files, generate the cavity and continue with a solvated calculation. In this case the restart files are given distinct extensions (e.g. `.dkn` vs. `.vacuum_dkn`).

1.3.2 Quasi-consistently-updated cavity

The second solution, reducing the error by about a factor of two, but twice as awkward, is to employ what we've termed quasi-self-consistent updating of the cavity. A fully converged *in vacuo* calculation is performed similarly to the first approach. Again, the solvated calculation proceeds by means of a restart and uses a fixed cavity, however it is terminated after several, say 3, steps, writing out the new density. Another restart is then performed, with the cavity being updated, as it is read from the saved files. This is again terminated after, say 3, steps and the process is repeated until convergence. In this way the cavity is updated in between restarts, avoiding the need to calculate the difficult extra term. This, however, means that the cavity is still fixed when calculating gradients, doing line searches and so on, and the process is not strictly variational, converging to an energy that is (expected to be) somewhere between the approximation of approach one and the true self-consistent energy. The awkwardness is ameliorated by the use of a script that automatically does the restarts within one PBS job, so this does not require more than one qsub.

1.3.3 Fully self-consistently-updated cavity

The third solution is to perform calculations with the cavity self-consistently responding to changes in density (as in [1]), but as mentioned earlier, this is costly, because it requires grids that are finer than default. This is achieved by setting `is_dielectric_model SELF_CONSISTENT` and increasing `fine_grid_scale` to something like 3. You might be able to get away with 2.5, you might need 3.5 or more. The memory and CPU cost increase with the cube of this value (which is 2.0 by default). Ideally, one such calculation could be performed to assess the magnitude of the error introduced by using one of the two simpler approaches. This error in the free energy of solvation is expected to be less than 3-4% percent for charged species and less than 1% for neutral species, meaning calculations with a fixed cavity are preferred.

1.4 The model: cavitation energy

The model in ONETEP includes the apolar cavitation term in the solvent-accessible surface-area (SASA) approximation, thus assuming the cavitation energy is proportional to the surface area of the cavity, the constant of proportionality being the (actual physical) surface tension of the solvent, γ, 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 will be used). This can be done using `is_solvent_surf_tension`. In contrast to previous versions, the actual physical value of surface tension is now expected (e.g. about 0.074 N/m for water).

1.5 The model: 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 cavitation uses, albeit with a smaller, and negative, prefactor. In practice this is most easily achieved by simply scaling the cavitation term down by a constant factor. A good scaling factor is 0.281705, which is what our model uses by default. The keyword controlling this parameter is `is_apolar_scaling_factor` and its argument is a unitless value.

2 The implementation

The NPE is solved by means of a multigrid solver. Currently ONETEP is interfaced to a solver called DL_MG [5, 6]. The solver is distributed with ONETEP and is compiled in by default. Solving the NPE is a memory- and time-consuming process, and you should expect solvation calculations to take about 2-3 times longer compared to standard ONETEP. The memory requirement of the solver grows cubically with the grid size along one dimension, so extra vacuum/bulk solvent is not free anymore, as soon as smeared ions are turned on!

The solver uses a multigrid approach to solve the NPE 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 [4, 6] for more information on the defect correction approach used in DL_MG.

Another limitation of the multigrid solver is the that 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. Now consider a grid with 174 points (and so 173 grid-edges). 173 is prime, and this grid cannot be subdivided at all.

Knowing about these limitations, ONETEP will sometimes slightly reduce 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 the multigrid 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 just over 1 bohr from your system.

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 your NGWFs extend into the margin, you're screwed.

Note that when using the recently introduced experimental support for fully periodic boundary conditions, the grid is resized in a different way in order to maintain the periodicity of the simulation cell — see section 6.3.2 for details.

2.1 Basic directives used in solvation calculations

- `is_implicit_solvent T/F` – turns on/off the implicit solvent. Default is off.
- `is_include_apolar T/F` – turns on/off the apolar energy terms. Default is on.
- `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 solvation parameter ρ_0 to x (atomic units). The default is 0.00035, as per [3].
- `is_solvation_beta` x – sets the solvation parameter β to x (no unit). The default is 1.3, as per [3].
- `is_bulk_permittivity` x – sets the solvation parameter ϵ_∞ 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 solvation parameter γ to x (unit must be supplied). The default is 0.07415 N/m (which is suitable for water near room temperature).
- `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.
- `mg_defco_fd_order` x ($\geq$ v4.5.13.5) or `is_discretization_order` x ($<$ v4.5.13.5)– sets the discretization order used when solving the NPE 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 weighted 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 β. 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 [4, 6] for more details.
- `is_smeared_ion_width` x – sets the width of the smeared-ion Gaussians, σ, to x (bohr). The default is 0.8 and should be OK for most calculations. The 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 cloud, especially in hydrogen atoms. This is very bad, since it does not change the energy *in vacuo* (the effect of the smearing, regardless of σ, 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 [4] for more details.
- `is_dielectric_model FIX_INITIAL/SELF_CONSISTENT` – picks either the fixed cavity or the self-consistently changing cavity, as described in section 1.3.
- `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 NPE and associated finite-difference operations are performed on (a subset of) the fine grid. Increasing `fine_grid_scale` allows for 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_auto_solvation` x – automatically runs a 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. Since v4.4 this directive is compatible with conduction calculations.

2.2 Advanced directives used in solvation calculations

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

- `is_multigrid_max_iters` x ($<$ v4.5.13.5 only)– changes the maximum number of multigrid iterations to x (no unit). The default is 100.
  - This keyword has been replaced in versions $\geq$ v4.5.13.5 with more specific keywords for controlling maximum numbers of iterations for different components of the solver (see below).
- `mg_max_iters_vcycle` x ($\geq$ v4.5.13.5 only)– sets the maximum number of multigrid V-cycle iterations to x (no unit). The default is 50. See [6] for a description of the solver, including the V-cycle scheme employed.
- `mg_max_iters_defco` x ($\geq$ v4.5.13.5 only)– sets the maximum number of high-order defect correction iterations to x (no unit). The default is 30. See [6] for a description of the solver, including the defect correction procedure.
- `mg_max_iters_newton` x ($\geq$ v4.5.13.5 only)– 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 Poisson-Boltzmann equation. Implicit solvation with Boltzmann ions is experimental functionality (see section 6.1). See [6] for a description of the inexact-Newton method employed by the solver.
- `mg_max_res_ratio` x ($\geq$ v5.3.1.11) – 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 ($\geq$ v5.3.1.11) – 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 ($\geq$ v5.3.1.11) – 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_continue_on_error T/F` ($\geq$ v5.3.4.4) – Instructs the multigrid solver not to abort if a solution to the NPE 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, you will very likely need to increase `is_pbe_energy_tolerance` by a very large amount.
- `is_multigrid_error_tol` x ($<$ v4.5.13.5 only) – changes the error tolerance used for the termination condition for the multigrid solver to x (no unit). The default is 1E-5. Smaller values (like 1E-7) will add negligible increase in accuracy at a significant computational cost. Larger values (like 1E-3) will incur significant loss of accuracy at a significant reduction of computational cost.
  - This keyword has been replaced in versions $\geq$ v4.5.13.5 with more specific keywords for controlling the convergence of different components of the solver (see below).

- `is_multigrid_defect_error_tol` x ($<$ v4.5.13.5 only) – changes the error tolerance used for the termination condition for every defect correction iteration in the multigrid solver to x (no unit). The default is 1E-2. Smaller values (like 1E-4) may be used to help in corner cases where the solver does not converge, but should only be used when necessary, since they increase the computational cost. Larger values (like 1) might decrease computational cost, but can break convergence, especially with higher orders.
  - This keyword has been replaced in versions $\geq$ v4.5.13.5 with more specific keywords for controlling the convergence of different components of the solver (see below).
- Convergence thresholds for controlling components of the multigrid solver ($\geq$ v4.5.13.5 only, see [5, 6] for more details about convergence control in DL_MG):
  - `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 Poisson-Boltzmann equation, 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 Poisson-Boltzmann equation, atomic units, the default is 1.0e-5).
- `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. [4].
- `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 possible, it's better to use `is_bc_coarseness` x with a larger x to speed up the calculation.
- `is_separate_restart_files T/F` (new in 3.5.3.2) – allows a different set of restart files to be used to construct the dielectric cavity in solvent, from the set of restart files to be used to

construct the 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.

- `is_solvation_properties T/F` (new in 5.3.2.5) – 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.

2.3 Expert directives used in solvation calculations

These will only be listed here and not discussed. The last three directives are discussed in a separate document devoted to the real space local pseudopotential.

- `mg_granularity_power` ($\geq$ v4.5.13.5) or `is_multigrid_nlevels` ($<$ v4.5.13.5),
- `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`.

2.4 Can you do solvation forces yet? Can you do geometry optimisation?

Yes! Since 3.5.8.0 the force terms due to implicit solvent should be automatically calculated any time you do standard force calculations. The formulas employed are exact (to numerical accuracy) when a fully-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 the geometry 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.

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.

Smeared-ion forces in vacuum are also implemented.

2.5 Can you do implicit solvation under PBCs or mixed boundary conditions?

This functionality is under development and experimental support for implicit solvent with fully periodic boundary conditions is available (see section 6.3).

2.6 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.

3 Various hints for a successful start

- Use one of the examples provided 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` (or `is_discretization_order` for < v4.5.13.5), `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` (or `is_discretization_order` for < v4.5.13.5) 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.
- Have at least about 10 bohr of vacuum/solvent around your molecule's NGWFs (not atomic positions) on each side of the simulation cell. Minimize the margin discussed in "The implementation".
- Always start your calculation in solution as a restart from a fully converged *in vacuo* calculation. Or just use `is_auto_solvation T`.

4 Troubleshooting: Problems, causes and solutions

- **Problem**: ONETEP crashes (MPI gets killed) when evaluating the boundary conditions or solving the NPE.
  Cause (1): You've run out of memory and the OOM killer killed the calculation. Solving the NPE represents the peak memory usage of the calculation.
  Solution (1): Increase memory or decrease box size or decrease grid fineness.
  Cause (2): You've run out of stack space. Solving the NPE represents the peak stack usage of the calculation.

Solution (2): Increase stack size using `ulimit -s`. Make sure you do that on compute nodes, not the login node.

- **Problem**: Multigrid calculation does not converge or converges very slowly. Multigrid iterations are the ones denoted with "MG" – they consist in defect-correcting the second-order solution to the NPE. Normally the multigrid calculation should converge within several iterations (in vacuum) and umpteen iterations (in solution). If it takes more iterations, the multigrid struggles to converge. In really bad cases it will diverge, which will stop the calculation with an error message.
  (Note: For more recent versions of ONETEP, information about the MG iterations is output to a log file with a filename ending `_dl_mg_log.txt`, while for older versions this is output into the ONETEP output file. If your ONETEP output file contains lines starting with the the text "MG DL_MG" then you have a newer version and should examine the log file for DL_MG iteration information. Otherwise, this should be directly available in the ONETEP output file.)
  Cause (1): Charge is not correctly localized (cell is too small).
  Solution (1): Check and fix the cell size, paying attention to the margin between the MG grid and fine grid.
  Cause (2): Dielectric permittivity too steeply changing on the cavity boundary for the current grid size, finite differences struggling to approximate the changes. This is the culprit if the calculation ran fine *in vacuo* but struggles in solvent.
  Solution (2): Preferable, but painful, solution is to make the grid finer (`fine_grid_scale`). Otherwise an increase or decrease of discretization order may help (make sure it stays consistent across your calculations, though). A parameterization with lower `is_solvation_beta` and `is_density_threshold` will usually help (make sure it stays consistent across your calculations, though).
  Cause (3): The smearing width is too small, making the smeared cores too thin and tall, which is difficult for the finite differences. This is the culprit if the calculation also struggles *in vacuo*.
  Solution (3): Increasing `is_smeared_ion_width` will help (but mind the consequences), if it was too small in the first place. Increasing the discretization order will help (especially if you've been using less than 10), but might lead to a similar problem (Cause (2)) in solution.
  Cause (4): Too lax thresholds for convergence of the defect correction in DL_MG.
  Solution (4): Decrease `is_multigrid_defect_error_tol` to 1E-3 or less (for $<$ v4.5.13.5). For $\geq$ v4.5.13.5, multiple variables are used to control the convergence of the defect correction in DL_MG (see section 2.2). To tighten the convergence threshold of the defect correction in DL_MG for $\geq$ v4.5.13.5, decrease the values of `mg_tol_res_rel`, `mg_tol_res_abs`, `mg_tol_pot_rel` and `mg_tol_pot_abs`.

- **Problem**: Calculation struggles to converge LNV or NGWFs or does not converge at all. RMS gradient stalls.
  Cause (1): 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 (1): Prefer `is_dielectric_model FIX_INITIAL`. If you definitely want `is_dielectric_model SELF_CONSISTENT`, make the grid finer and have a lot of memory.
  Cause (2): Density kernel is not converged enough.
  Solution (2): Try `minit_lnv 6` and `maxit_lnv 6` (for smaller molecules) or `minit_lnv 10` and `maxit_lnv 10` (for large molecules).

5 What are the values for the model parameters?

Two sets of values will be proposed here. The first one will be called "high-beta" parameterization. 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 β 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 parameterization, 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 parameterizations 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 parameterization may be used if one is only interested in anionic species.

6 Recent changes

6.1 New in version 4.4

ONETEP version 4.4 includes a number of changes to the DL_MG solver that should be transparent to end-users (apart from improvements in performance). There are a few visible changes, outlined below.

- `is_multigrid_error_damping T/F` (< v4.5.13.5 only) can be used to turn on 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 (the latter being the only supported scenario currently). Accordingly, the default depends on `is_pbe` and is `F` for `is_pbe NONE` and `is_pbe LINEARISED`, and `T` for `is_pbe FULL`.
  - This is replaced by `mg_use_error_damping` in $\geq$ v4.5.13.5.
- `is_multigrid_verbose T/F` can be used to output cross-sections of quantities that are of interest during multigrid calculations to text files. For instance it might be desirable to examine the permittivity to verify whether a pocket in a molecule is solvent accessible or not. The cross sections are always performed along the X direction, for a given value of Y and Z. The following quantities are output:
  - Total density (rho),

  - Boundary condition for the potential (bound),
  - Dielectric permittivity at gridpoint (eps_full),
  - Dielectric permittivity offset by half grid point along X (eps_half_x),
  - Dielectric permittivity offset by half grid point along Y (eps_half_y),
  - Dielectric permittivity offset by half grid point along Z (eps_half_z),
  - Steric potential (steric_pot) – only for `is_pbe` different from `NONE`.
  - Steric weight a.k.a. accessibility (steric_w) – only for `is_pbe` different from `NONE`.

  In addition the following quantities are output (to a separate file) for every defect correction iteration:

  - Total defect (defect),
  - Sum of source and Poisson contributions to the defect (defect_stpt),
  - Boltzmann contribution to the defect (defect_bt).
  - The potential value at which the derivative of the Boltzmann term is computed (der_pot) – only for `is_pbe` different from `NONE`.

- `is_multigrid_verbose_y` y – used in combination with `is_multigrid_verbose`. Specifies the offset along the Y axis to be y (make sure you provide units).
- `is_multigrid_verbose_z` z – used in combination with `is_multigrid_verbose`. Specifies the offset along the Z axis to be z (make sure you provide units).

Note: The `is_multigrid_verbose`, `is_multigrid_verbose_y`, `is_multigrid_verbose_z` keywords are not available in versions of ONETEP $>$ 4.5.6.0 without using the `devel_code` value `MG:USE_ONETEP_DEFCO=[T/F]:MG`, which activates an older version of the defect correction procedure. This is not generally recommended.

The following keywords control the experimental Poisson-Boltzmann solver functionality, which allows performing calculations in implicit solvent containing Boltzmann ions. While this functionality works to the best of our knowledge, we do not provide support for it yet. Below is the bare list of keywords for controlling it. These will be described further once this functionality becomes supported.

- `is_pbe`,
- `is_pbe_temperature`,
- `is_pbe_exp_cap`,
- `is_pbe_use_fas` ($<$ v4.5.13.5),
- `mg_pbe_use_fas` ($\geq$ v4.5.13.5),
- `is_pbe_bc_debye_screening`,
- `is_pbe_steric_pot_type` ($\geq$ v5.3.4.0),
- `is_pbe_neutralisation_scheme` ($\geq$ v5.3.3.7),

- is_pbe_energy_tolerance ($\geq$ v5.3.3.7),
- is_steric_write,
- is_hc_steric_cutoff ($<$ v5.3.1.11),
- is_hc_steric_smearing ($\geq$ v5.3.1.11),
- is_hc_steric_dens_isovalue ($\geq$ v5.3.1.11),
- is_sc_steric_cutoff,
- is_sc_steric_magnitude,
- is_sc_steric_smoothing_alpha,
- sol_ions,
- species_solvent_radius ($\geq$ v5.3.4.0).

6.2 New in version 4.5.12.8

Here we expose functionality for 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 $Y = 18.4\,a_0$, $Y = 20.7\,a_0$ 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. 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 (as of version 4.5.15.19) the boundaries of the exclusion regions can be smoothed. This is achieved using a Fermi-Dirac function,

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

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 Bohr, 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.

The values of the permittivity can be easily examined via `is_multigrid_verbose` (for $< 4.5.6.0$).

6.3 New in version 4.5.12.12: Experimental PBC solvation support

Experimental support for implicit solvent calculations in periodic BCs has been implemented. This is under active development and you should therefore treat any results obtained with this functionality with caution.

Using this functionality, the implicit solvent model can be applied in full PBCs, i.e. where the system is periodic along all simulation cell directions. Support for mixed open/periodic BCs is planned for the future, but is not currently supported.

BCs can be specified individually for the multigrid solver, local pseudopotential, ion-ion interaction and smeared ion representation using the following new 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 (Coulombic) 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).

6.3.1 Key points on using the new 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.

6.3.2 Grid sizes in periodic BCs

Under periodic BCs, 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 2) 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 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. [5, 6] for details about these requirements).

Modifying the fine grid scale factor causes ONETEP to use the modified fine grid throughout the calculation. 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. This is usually 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. In open BCs, this is not an issue, since the grid used by the multigrid solver is simply a truncated version of the usual fine grid, with the same grid point spacing.

6.3.3 Known issues and untested functionality

- Periodic BCs are not yet currently compatible with the `is_dielectric_exclusions` block.
- Periodic BCs have not been tested in self-consistent cavity mode—only fixed cavity calculations are known to work.
- Forces and geometry optimization have not been tested when using implicit solvent under periodic BCs—only energy calculations are known to work.
- Care should be taken when dealing with charged solutes as a neutralizing background charge is required when solving the Poisson equation for non-neutral systems under periodic BCs (this will have implications for the polarization of the implicit solvent).

7 Questions?

Questions about implicit solvation in ONETEP should be directed to Jacek Dziedzic (`J.Dziedzic[-at-]soton.ac.uk.`) or to James Womack (`J.C.Womack[-at-]soton.ac.uk.`)

References

[1] Scherlis, Fattebert, Gygi, Cococcioni and Marzari, J. Chem. Phys. **124** (2006).

[2] Floris, Tomasi and Ahuir, J. Comp. Chem. **12** (1991).

[3] Dziedzic, Helal, Skylaris, Mostofi and Payne, EPL **95** (2011).

[4] Dziedzic, Fox, Fox, Tautermann and Skylaris, International Journal of Quantum Chemistry **113** issue 6 (2013).

[5] L. Anton, J. Womack, and J. Dziedzic, DL_MG multigrid solver, 2017, http://www.dlmg.org

[6] 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).