
Article:
Probert, Matthew Ian James orcid.org/0000-0002-1130-9316, Womack, James, Anton, Lucian et al. (3 more authors) (2018) DL_MG:A Parallel Multigrid Poisson and Poisson–Boltzmann Solver for Electronic Structure Calculations in Vacuum and Solution. Journal of chemical theory and computation. pp. 1412-1432. ISSN 1549-9618

https://doi.org/10.1021/acs.jctc.7b01274

DL_MG: A parallel multigrid Poisson and Poisson-Boltzmann solver for electronic structure calculations in vacuum and solution

James C. Womack,[†] Lucian Anton,[‡] Jacek Dziedzic,[†,¶] Phil J. Hasnip,[§] Matt I. J. Probert,[§] and Chris-Kriton Skylaris[*,†]

†*School of Chemistry, University of Southampton, Highfield, Southampton, SO17 1BJ, United Kingdom*

‡*Cray U.K. Limited, Broad Quay House, Prince Street, Bristol, BS1 4DJ, United Kingdom*

¶*Faculty of Applied Physics and Mathematics, Gdańsk University of Technology, Gdańsk 80-233, Poland*

§*Department of Physics, University of York, Heslington, York YO10 5DD, United Kingdom*

E-mail: C.Skylaris@soton.ac.uk

Abstract

The solution of the Poisson equation is a crucial step in electronic structure calculations, yielding the electrostatic potential—a key component of the quantum mechanical Hamiltonian. In recent decades, theoretical advances and increases in computer performance have made it possible to simulate the electronic structure of extended systems in complex environments. This requires the solution of more complicated variants of the Poisson equation, featuring non-homogeneous dielectric permittivities, ionic concentrations with non-linear dependencies, and diverse boundary conditions. The analytic solutions generally used to solve the Poisson equation in vacuum (or with homogeneous permittivity) are not applicable in these circumstances and numerical methods must be used.

In this work, we present DL_MG, a flexible, scalable and accurate solver library, developed specifically to tackle the challenges of solving the Poisson equation in modern large-scale electronic structure calculations on parallel computers. Our solver is based on the multigrid approach and uses an iterative high-order defect correction method to improve the accuracy of solutions.

Using two chemically relevant model systems, we tested the accuracy and computational performance of DL_MG when solving the generalized Poisson and Poisson-Boltzmann equations, demonstrating excellent agreement with analytic solutions and efficient scaling to $\sim 10^9$ unknowns and 100s of CPU cores. We also applied DL_MG in actual large scale electronic structure calculations, using the ONETEP linear-scaling electronic structure package to study a 2615 atom protein-ligand complex with routinely available computational resources. In these calculations, the overall execution time with DL_MG was not significantly greater than the time required for calculations using a conventional FFT-based solver.

1 Introduction

What is the electrostatic potential corresponding to a given charge density? This deceptively simple question is key to modelling the electronic structure of atoms, molecules and materials, where the classical electrostatic potential forms a foundation upon which quantum mechanical many-body effects can be modeled. Consequently, developing efficient techniques for answering this question—by solving the Poisson equation—is a central concern for researchers interested in the electronic structure of matter.

For reasons of practicality, electronic structure calculations have historically tended to be restricted to the study of small systems in vacuum with fully open or fully periodic boundary conditions (BCs). In this case, the Poisson equation can be efficiently solved using analytic solutions (as described in section 2.1). However, in recent years it has become possible—perhaps even routine—to perform electronic structure calculations on systems numbering 100s or 1000s of atoms and to include the effect of the surrounding environment. This has been driven by prodigious

growth in the computing power available to researchers and theoretical developments allowing electronic structure calculations to scale efficiently with respect to system size and number of processors. In particular, progress in this area has been enabled by the development of so-called linear-scaling, or $O(N)$, methods, in which the asymptotic computational cost increases linearly with system size, N. These methods have been implemented in several software packages, including ONETEP,[1] BigDFT,[2] CONQUEST,[3] OpenMX,[4] Quickstep,[5] and SIESTA.[6]

When modelling large chemical systems, such as biomolecules and nanoparticles, neglecting the environment can have a substantial effect on the properties of the system. For example, without the screening effect of a solvent, it is possible for systems to develop unphysical surface states and dipole moments.[7] This issue can be resolved by simply including solvent molecules in the electronic structure calculation. However, this explicit approach is very costly, even using linear-scaling methods, because of the significant increase in the number of atoms that must be treated quantum mechanically and the need to statistically average over solvent configurations. In addition, it is generally the case that the electronic structure of the environment is not of interest and may complicate the interpretation of results.

The generalized Poisson and Poisson-Boltzmann equations (see section 2.1) offer a computationally inexpensive means of embedding a charge density in an electrostatic environment, avoiding the complexities of explicit modelling of the environment. Solving these equations yields an electrostatic potential which includes an implicit representation of the electrostatic effects of the environment, for example, a solvent. However, analytic solutions for these more complicated variants of the Poisson equation are only available for specific cases (see section 4.1.1), necessitating the use of numerical methods.

To be practical in the context of large scale electronic structure calculations, a numerical Poisson solver must:

- solve the Poisson equation for arbitrary input charge densities,
- have accuracy comparable to methods based on analytic solutions for the Poisson equation in vacuum (e.g. using FFTs to solve the equation in reciprocal space, section 2.1),

- scale efficiently with problem size, and
- scale efficiently to large numbers of parallel processors.

In addition, if the solver is to provide an implicit representation of the environment, it must also be able to solve the more complicated generalized Poisson and/or Poisson-Boltzmann equations.

Using the multigrid approach,[8–10] Poisson solvers which satisfy all of these requirements can be developed. Multigrid methods provide a framework in which relatively simple iterative solvers can be applied on a hierarchy of progressively coarsening grids, yielding rapid convergence at low computational cost (section 2.3). With careful design and selection of components, multigrid solvers can also achieve excellent parallel efficiency (see ch. 6 of Ref. 10).

The use of multigrid solvers for solving the Poisson equation in electronic structure calculations is well-established, with many publications describing their successful application in this context.[11–17] Multigrid methods have also been applied as efficient solvers for real-space discretizations of the Kohn-Sham eigenvalue equations in density functional theory (DFT).[18–21]

Solvers based on the multigrid approach have proven particularly effective for solving the generalized Poisson equation in implicit solvent models based on Fattebert and Gygi's electrostatic model[12,14–17,22,23] (section 2.2). The smoothly varying function used to represent the dielectric permittivity in these models poses no problem for multigrid solvers, requiring only that the operator stencil (appendix A) is modified to incorporate variable coefficients.

While multigrid is clearly well-suited for solving the Poisson equation in electronic structure calculations featuring electrostatic embedding, it is not the only approach in use. For example, Andreussi *et al.* implemented the self-consistent continuum solvation (SCCS) model[24]—a variant of Fattebert and Gygi's model—by recasting the generalized Poisson equation in terms of a polarization charge density and solving this self-consistently (see section 2.1). More recently, Fisicaro *et al.*[25] extended this work, presenting efficient solvers for the generalized Poisson and Poisson-Boltzmann equations based upon preconditioned conjugate gradient and self-consistent methods for use in the SCCS and similar models.

In this paper, we introduce DL_MG, a flexible, scalable and accurate Poisson solver library based

upon a high-order defect-corrected multigrid approach. The solver was designed specifically to tackle the challenges inherent in modern large-scale electronic structure calculations. In particular, the library was developed to provide a means of accounting for environmental effects in electronic structure calculations by efficient solution of the generalized Poisson and Poisson-Boltzmann equations.

In the following, we present the theoretical context for the development of DL_MG (section 2) and an overview of the implementation of the library (section 3), focusing particularly on the defect correction component (section 3.1.2). Through careful testing of the solver for chemically relevant model systems (section 4.1) and in large-scale electronic structure calculations (section 4.2) with ONETEP,[1] we demonstrate that the solver is able to scale efficiently to 100s of processors and $\sim 10^9$ grid points and deliver close agreement with known analytic results and established FFT-based Poisson solvers. In addition, since DL_MG is freely available under a permissive open source license, we provide some brief information for developers in appendix B to aid interested readers who may want to test and possibly integrate the library in their own codes.

2 Theory

2.1 Poisson and Poisson-Boltzmann equation

The electrostatic potential, $\phi_0(\mathbf{r})$, resulting from a given charge density, $n(\mathbf{r})$, in vacuum can be obtained by solving the Poisson equation:

$$\nabla^2 \phi_0(\mathbf{r}) = -4\pi n(\mathbf{r}). \tag{1}$$

The solution of this equation is relevant in the context of electronic structure calculations, where the potentials due to electronic and ionic densities, $n_{\text{elec}}(\mathbf{r})$ and $n_{\text{ionic}}(\mathbf{r})$, are required.

In open boundary conditions (OBCs), where the potential goes to zero as $\mathbf{r}$ goes to infinity, the non-periodic potential can be expressed in terms of the corresponding Green's function for the

Laplacian, ∇^2 (see ch. 10 of Ref. 26):

$$G(\mathbf{r}-\mathbf{r}') = -\frac{1}{4\pi}\frac{1}{|\mathbf{r}-\mathbf{r}'|}. \tag{2}$$

This yields the well-known form for Coulomb potential in OBCs:

$$\phi_0(\mathbf{r}) = \int \mathrm{d}\mathbf{r}' \frac{n(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|}, \tag{3}$$

where the integration is over all space.

Under periodic boundary conditions (PBCs), the potential corresponding to a given periodic charge density can be obtained directly by solving the equation in reciprocal space, i.e.

$$\widetilde{\phi}_0(\mathbf{G}) = 4\pi\frac{\widetilde{n}(\mathbf{G})}{|\mathbf{G}|^2}, \tag{4}$$

where $\widetilde{\phi}_0(\mathbf{G})$ and $\widetilde{n}(\mathbf{G})$ are the Fourier transforms of the real-space potential and charge density, respectively. This simple expression is of great utility in electronic structure calculations using periodic basis functions, where Fast Fourier Transforms (FFTs) can be employed to efficiently transform quantities between real- and reciprocal-space.

While the form of Eq. 4 is convenient, it also illustrates a particular difficulty encountered when solving the Poisson equation with PBCs, namely that the charge density must be neutral (i.e. $\widetilde{n}(\mathbf{0}) = 0$). Non-neutral charge densities result in a singularity in the potential at $\mathbf{G} = \mathbf{0}$. In practice, this issue is typically avoided by introducing a compensating homogeneous background charge which ensures that the overall charge in the periodic unit cell is neutral, equivalent to solving

$$\nabla^2\phi_0(\mathbf{r}) = -4\pi\left\{n(\mathbf{r}) - \langle n\rangle\right\}, \tag{5}$$

where $\langle n \rangle$ is the average charge density over the volume of the unit cell, V, i.e.

$$\langle n \rangle = \frac{1}{V} \int_V \mathrm{d}\mathbf{r}\, n(\mathbf{r}). \tag{6}$$

The subtraction of $\langle n \rangle$ from the real-space charge density, $n(\mathbf{r})$, in Eq. 5 is equivalent to setting $\widetilde{n}(\mathbf{0}) = 0$, thus avoiding singularities in Eq. 4. While this is a useful method for obtaining a solution from the Poisson equation when dealing with a non-neutral periodic density, it necessarily changes the nature of the problem—the potential obtained corresponds to the periodic density and the artificial neutralizing background charge.

In electronic structure calculations, it is convenient to deal with the interactions of the electronic and ionic components of the overall charge density separately. Since the electronic and ionic densities are independently non-neutral, neutralizing background charges must be used for each component when solving the Poisson equation in PBCs. As described in section 2.2, this has no impact on the energy of a neutral system, since the contributions due to the background charges in each term cancel out.

The generalized Poisson equation (GPE) ,

$$\nabla \cdot (\varepsilon(\mathbf{r}) \nabla \phi(\mathbf{r})) = -4\pi n(\mathbf{r}), \tag{7}$$

is a generalization of Eq. 1 in which the dielectric permittivity, $\varepsilon(\mathbf{r})$, can vary with position—Eq. 1 (which we will call the "standard" Poisson equation, or SPE) corresponds to the situation where $\varepsilon(\mathbf{r}) = 1$ over all space.[27]

While analytic solutions for the GPE can be obtained for specific cases (see, for example, section II.C of Ref. 25), Eq. 7 is typically solved using numerical methods (such as the multigrid approach, section 2.3). Such techniques allow the equation to be solved for complicated forms of $n(\mathbf{r})$ and $\varepsilon(\mathbf{r})$. An important application of these techniques is in electronic structure calculations (section 2.2), in which a quantum mechanical charge density is embedded in a polarizable dielectric medium, implicitly representing the environment (e.g. a solvent).

Eq. 7 may also be recast in a non-linear form which resembles the SPE (Eq. 1),

$$\nabla^2\phi(\mathbf{r}) = -4\pi(n(\mathbf{r}) + n_{\text{pol}}[\phi](\mathbf{r})), \tag{8}$$

where the polarization charge density, $n_{\text{pol}}[\phi](\mathbf{r})$, depends upon the potential. This form allows techniques for solving the simpler SPE to be employed (e.g. via Eq. 4), though the dependence of $n_{\text{pol}}(\mathbf{r})$ on the potential means the solution must be obtained via a self-consistent procedure. For further details, see Ref. 24, which describes an implicit solvent model based upon the self-consistent solution of Eq. 8, related to the model outlined in section 2.2.

The GPE may be further extended by introducing a potential-dependent density of mobile ions in the dielectric medium, $n_{\text{ions}}[\phi](\mathbf{r})$, i.e.

$$\nabla \cdot (\varepsilon(\mathbf{r})\nabla\phi(\mathbf{r})) = -4\pi(n(\mathbf{r}) + n_{\text{ions}}[\phi](\mathbf{r})). \tag{9}$$

This mobile ion density at a given $\mathbf{r}$ may be generally written as

$$n_{\text{ions}}[\phi](\mathbf{r}) = \lambda(\mathbf{r}) \sum_{i=1}^{m} q_i c_i[\phi](\mathbf{r}), \tag{10}$$

where $c_i[\phi](\mathbf{r})$ and q_i are, respectively, the local concentration and charge of ionic species i; m is the total number of ionic species present; and $\lambda(\mathbf{r})$ is a function which describes the accessibility of $\mathbf{r}$ to the mobile ions. When $c_i[\phi](\mathbf{r})$ takes the form of a Boltzmann distribution,

$$c_i[\phi](\mathbf{r}) = c_i^{\infty} \exp\left(-\frac{q_i\phi(\mathbf{r})}{k_{\text{B}}T}\right) \tag{11}$$

with bulk concentration c_i^{∞}, Boltzmann constant k_{B}, and temperature T, Eq. 9 becomes the Poisson-Boltzmann equation (P-BE):

$$\nabla \cdot (\varepsilon(\mathbf{r})\nabla\phi(\mathbf{r})) = -4\pi n(\mathbf{r}) - 4\pi\lambda(\mathbf{r}) \sum_{i=1}^{m} c_i^{\infty} q_i \exp\left(-\beta q_i \phi(\mathbf{r})\right), \tag{12}$$

where we have used $\beta = 1/k_{\mathrm{B}}T$.

For a given charge density $n(\mathbf{r})$, dielectric permittivity $\varepsilon(\mathbf{r})$, accessibility function $\lambda(\mathbf{r})$, and the charges and bulk concentration of mobile charges $\{q_i\}$ and $\{c_i^\infty\}$, the P-BE (Eq. 12) can be solved to yield an overall electrostatic potential, $\phi(\mathbf{r})$. The equation may therefore be applied in situations where a static charge density is embedded in a dielectric medium and surrounded by mobile charges.

An important application of the P-BE is in classical modelling of the electrostatics of biomolecules in ionic solutions, where the atoms constituting the biomolecule are typically represented by point charges, the solvent as a dielectric medium, and the concentrations of species of mobile ions in solution are represented by a Boltzmann distribution (Eq. 11). The use of Eq. 12 in biomolecular contexts has been reviewed in Refs. 28,29. The P-BE may be similarly applied in electronic structure calculations (section 2.2), but with the quantum mechanical electron charge density represented as a smooth function, rather than a collection of atom-centered point-charges. This allows the effect of a saline solution on the electronic structure of a solute to be modeled implicitly, without the need for atomistic modelling of either the solvent or dissolved ions.

The non-linear P-BE (NLP-BE) may be approximated by a simpler linearized form when the electrostatic potential, $\phi(\mathbf{r})$, is small. In this case, the Boltzmann term in Eq. 12 is approximated as the first two terms in a Maclaurin series:

$$\exp\left(-\beta q_i \phi(\mathbf{r})\right) \approx 1 - \beta q_i \phi(\mathbf{r}). \tag{13}$$

Inserting this approximation into Eq. 12 yields the linearized Poisson-Boltzmann equation (LP-BE):

$$\nabla \cdot (\varepsilon(\mathbf{r}) \nabla \phi(\mathbf{r})) = -4\pi n(\mathbf{r}) + 4\pi\lambda(\mathbf{r})\beta \sum_{i=1}^{m} c_i^\infty q_i^2 \phi(\mathbf{r}). \tag{14}$$

Note that for a neutral solution of mobile ions

$$4\pi\lambda(\mathbf{r}) \sum_{i}^{m} c_i^\infty q_i = 0 \tag{15}$$

and thus this term does not appear in Eq. 14.

In the standard Poisson-Boltzmann model outlined in this section, the mobile charges are point-like particles with a statistical distribution based on the overall electrostatic potential of the system (Eq. 11). As a consequence, the model neglects finite size effects, which can lead to unphysical accumulation of ions where the static charge density is large. This issue can be addressed by employing a size-modified Poisson-Boltzmann (SMPB) model. See Ref. 30 for a review of models of this type and Refs. 31,32 for recent work implementing and parameterizing an SMPB-based implicit solvent model for use in DFT calculations.

2.2 Electronic structure calculations

The classical electrostatic energy of a charge density interacting with itself is given by

$$E_{\text{es}}[n] = \frac{1}{2}\int \mathrm{d}\mathbf{r}\, n(\mathbf{r})\phi_0[n](\mathbf{r}), \tag{16}$$

where the potential, $\phi_0[n](\mathbf{r})$, is the solution of the SPE (Eq. 1). If the charge density represents the total charge of a collection of atoms, then this can be decomposed into contributions from the electrons and ionic cores, i.e.

$$E_{\text{es}}[n_{\text{elec}}, n_{\text{ion}}] = E_{\text{Hartree}}[n_{\text{elec}}] + E_{\text{elec-ion}}[n_{\text{elec}}, n_{\text{ion}}] + E_{\text{ion-ion}}[n_{\text{ion}}]. \tag{17}$$

The Hartree energy E_{Hartree}, and ion-ion energy $E_{\text{ion-ion}}$, are defined analogously to Eq. 16 for each density, though in practice they differ in how they address self-interaction. For the ion-ion term, the self-interaction is typically explicitly subtracted within $E_{\text{ion-ion}}$, while the electronic self-interaction is not considered in the classical electrostatic energy—in electronic structure methods, this is part of the exchange contribution to the total energy. The electron-ion interaction, where no self-interaction correction is necessary, is given by

$$E_{\text{elec-ion}}[n_{\text{elec}}, n_{\text{ion}}] = \int \mathrm{d}\mathbf{r}\, n_{\text{elec}}(\mathbf{r})\phi_0[n_{\text{ion}}](\mathbf{r}) = \int \mathrm{d}\mathbf{r}\, n_{\text{ion}}(\mathbf{r})\phi_0[n_{\text{elec}}](\mathbf{r}). \tag{18}$$

The overall classical electrostatic energy typically represents a significant fraction of the total energy computed in an electronic structure calculation.

Electronic structure calculations are generally concerned with the behavior of electrons in the presence of nuclei at set of fixed positions. In this situation, it is convenient to separate the total charge density into electronic and nuclear components, as in Eq. 17. The electron density, $n_{\text{elec}}(\mathbf{r})$, can then be treated as a continuous function, while the nuclear density, $n_{\text{nuc}}(\mathbf{r})$, is represented as a sum of point charges,

$$n_{\text{nuc}}(\mathbf{r}) = \sum_I Z_I \delta(\mathbf{r} - \mathbf{R}_I), \tag{19}$$

with positions $\{\mathbf{R}_I\}$ and charges $\{Z_I\}$. This allows the potentials corresponding to the two densities to be solved for independently, using methods appropriate to their form.

In self-consistent field (SCF) methods, such as DFT and Hartree-Fock theory, the electron density is constructed as a sum over one-electron orbitals, $\psi_i(\mathbf{r})$, weighted by their occupancies, f_i, i.e.

$$n_{\text{elec}}(\mathbf{r}) = \sum_i f_i \psi_i(\mathbf{r}) \psi_i^*(\mathbf{r}). \tag{20}$$

The orbitals, and hence the electron density, are obtained by solving one-electron Schrödinger equations of the general form

$$\left(-\frac{1}{2}\nabla^2 + \hat{V}_{\text{eff}}\right) \psi_i(\mathbf{r}), = \varepsilon_i \psi_i(\mathbf{r}). \tag{21}$$

where the electrostatic potentials associated with the nuclei and electrons are components of the effective potential $\hat{V}_{\text{eff}}$. These equations must be solved self-consistently, since the effective potential is dependent on the orbitals, in part due to the relationship between the electrostatic potential and electron density (Eq. 1).

While the nuclear charge density is generally fixed during an SCF calculation, the orbitals, and hence electron density, are updated as part of the iterative process. As a consequence, the electrostatic potential due to the electron density—the Hartree potential—must be repeatedly solved for during the SCF procedure. Efficient methods for solving the SPE for a given electron density

are therefore of great importance in the implementation of SCF approaches.

For SCF calculations in vacuum, the SPE (Eq. 1) can be solved using the analytic solutions described in section 2.1. The approach used in a particular calculation is generally determined by the nature of the underlying basis set—for periodic plane wave basis sets, the reciprocal space solution is convenient (Eq. 4) while for local, non-periodic basis sets, the OBC solution derived using a Green's function method (Eq. 3) is typically used. In the case of periodic plane wave basis sets, Fast Fourier Transforms (FFTs) allow the solution of the Poisson equation with computational effort scaling as $O(n \log n)$, where n is the number of grid points.[33]

While there are efficient methods for obtaining the Hartree potential using analytic solutions to the SPE, numerical approaches have some utility under certain circumstances. One such situation is the case where a periodic basis set is used to represent a finite system. To reduce the extent of spurious interactions between periodic images of the finite system, it is typical to use the "supercell" approximation,[34] in which the finite system is surrounded by a large volume of vacuum "padding". The additional padding required for this approach can be computationally expensive for basis sets which grow with cell size (for example, plane-waves). Real-space numerical approaches can be employed to efficiently solve for the electrostatic potential, while imposing open BCs. This completely eliminates electrostatic interactions between periodic images, while allowing the use of a periodic basis set—see Ref. 35, for example.

It is often useful to study the electronic structure of systems embedded in a medium with a nonhomogeneous dielectric permittivity, $\varepsilon(\mathbf{r})$. In this case, the total electrostatic potential can be obtained by solving the GPE (Eq. 7) and comprises two terms:

$$\phi(\mathbf{r}) = \phi_0(\mathbf{r}) + \phi_\mathrm{r}(\mathbf{r}), \tag{22}$$

where $\phi_0(\mathbf{r})$ is the usual electrostatic potential associated with the charge distribution of the system and $\phi_0(\mathbf{r})$ is a reaction potential due to the polarization of the dielectric medium, $\phi_\mathrm{r}(\mathbf{r})$. A key application of this model is in modelling solvent effects on electronic structure, using a polarizable

dielectric medium to implicitly represent the solvent environment.

As mentioned in section 2.1, the dearth of widely applicable analytic solutions for the GPE means that the equation is typically tackled using numerical approaches or recast as a non-linear form of the SPE and solved self-consistently (as described in Refs. 24,25). Some of the most widely used implicit solvent models employ an additional simplifying assumption that the system is separated into two regions in which the dielectric permittivity is homogeneous, i.e.

$$\varepsilon(\mathbf{r}) = \begin{cases} 1, & \text{solute cavity} \\ \varepsilon_\infty, & \text{bulk solvent.} \end{cases} \tag{23}$$

The solute cavity is defined in such a way that it incorporates the solute charge and the boundary between the two regions is discontinuous. In this model, it is possible to reformulate the problem of solving the GPE purely in terms of a polarization charge on the surface of the cavity. This apparent surface charge (ASC) defines the reaction potential, and so solving the GPE becomes a question of determining the ASC over the 2-D surface of the cavity—see Ref. 36 for an overview of ASC-type approaches.

In this work, we are concerned with the numerical solution of the full GPE in 3-D for an arbitrary nonhomogeneous dielectric permittivity. This provides a flexible foundation for the development of implicit solvent models in which the form of the dielectric permittivity is not restricted to the discontinuous piecewise form used in ASC approaches. In particular, solving the full GPE in 3-D allows the dielectric permittivity to smoothly transition between the bulk values within the solute cavity and solvent.

An advantage of solving the full GPE in 3-D with a smooth dielectric function is that this yields a continuous potential, thus evading the difficulties associated with discontinuous gradients which can arise in ASC approaches.[37] This was a motivating factor for Fattebert and Gygi in developing an electrostatic implicit solvent model based upon a smooth dielectric function for use in molecular dynamics, where accurate energy gradients are critical.[12,22] In Fattebert and Gygi's model $\varepsilon(\mathbf{r})$ is

a functional of the solute electron density,

$$\varepsilon(\mathbf{r}) = 1 + \frac{\varepsilon_\infty - 1}{2}\left(1 + \frac{1 - (n_{\mathrm{elec}}(\mathbf{r})/n_0)^{2\beta}}{1 + (n_{\mathrm{elec}}(\mathbf{r})/n_0)^{2\beta}}\right), \tag{24}$$

and is defined in terms of the electron density at $\mathbf{r}$, $n_{\mathrm{elec}}(\mathbf{r})$, the bulk permittivity of the solvent, ε_∞, and two parameters: β and n_0.

The use of the electron density to construct the solute cavity has the advantage that a good representation of the solute shape can be obtained using very few fitted parameters. Fattebert and Gygi's cavity is defined by only two fitted parameters—significantly fewer than the number required when employing the widely adopted method of constructing the cavity from atom-centered spheres. This model has since been elaborated and extended in a number of respects, including: inclusion of non-electrostatic cavitation and dispersion-repulsion effects;[15,23,24] alternative dielectric functions based on the electron density[24] and defined based on atom-centered functions;[14,38] use of open (Dirichlet) BCs and their efficient computation using coarse-graining;[16] and, extension of the model to the P-BE and size-modified variants.[25,31]

A variant of Fattebert and Gygi's model—the minimal parameter implicit solvent model, or MPSM[15,16]—was implemented in ONETEP,[1] an electronic structure package capable of performing calculations with a cost that scales linearly with the number of atoms, N. Using efficient parallel implementations of algorithms with formal $O(N)$ scaling, ONETEP is able to perform full DFT calculations on systems consisting of thousands of atoms.[39–41] In this context, it was vital to ensure that the implementation of the solvent model was compatible with overall $O(N)$ scaling of ONETEP and was able to operate efficiently in parallel.

The need for an efficient parallel GPE solver for use in large-scale MPSM calculations in ONETEP was the key motivation for the development of DL_MG. The multigrid approach was a natural choice for this application, for two key reasons. First, the multigrid method is well-suited to implementation in parallel and exhibits excellent computational scaling with respect to the grid size.[10] Second, the representation of the charge density and electrostatic potential on a regular

grid in ONETEP is ideal for use with a multigrid solver, allowing well-established and understood variants of the method to be used, as described in the following sections.

2.3 Multigrid

To solve the Poisson equation in situations where exact reciprocal space solutions are not available, real-space numerical approaches can be employed. In numerical approaches, a discretized version of the Poisson equation is required. In the context of electronic structure methods with periodic plane-wave-type basis sets, it is natural to discretize the problem on the regular real-space grid used to represent the electronic charge density, i.e.

$$\hat{A}_h u_h = f_h, \tag{25}$$

where u_h and f_h are, respectively, the potential we are solving for and source term (the charge density multiplied by a factor of -4π for Eqs. 1 and 7), both discretized on a regular grid with spacing h. $\hat{A}_h$ is the linear operator, the form of which depends on which of the variant of the Poisson equation we are considering: $\hat{A} \equiv \nabla^2$ for the SPE (Eq. 1) and $\hat{A} \equiv \nabla \cdot \varepsilon(\mathbf{r})\nabla$ for the generalized Poisson equation (Eq. 7).

The discretized Poisson equation forms a system of linear equations which are amenable to solution by stationary iterative methods, such as the Jacobi and Gauss-Seidel methods (see ch. 19 and ch. 2 of Refs. 33 and 9, respectively, for introductions to these and similar techniques). It is well known that stationary iterative methods can very effectively smooth high-frequency components of the error. However, the overall convergence of these methods towards the solution is limited by low-frequency components in the error, which are less effectively removed and become increasingly prevalent when using finer grids.[9]

Multigrid methods[8–10] simultaneously take advantage of the smoothing property of iterative solvers while addressing their slow rate of convergence. This is achieved by applying a hierarchy of progressively coarsening grids to the problem. Since low-frequency components of the error repre-

sented on a given grid appear as higher-frequency components on a relatively coarser grid, iterative methods can be applied on the coarser grid to rapidly attenuate the low-frequency components, avoiding the problematic slow convergence that arises with a single-grid approach.

Consider the general linear equation

$$\hat{A}u = f, \tag{26}$$

with the corresponding defect (or residual)

$$r = f - \hat{A}u, \tag{27}$$

and defect equation

$$\hat{A}e = r, \tag{28}$$

where f is the source term and u is the quantity we wish to solve for. We can define three basic operations:

Smoothing Apply an iterative method to remove higher frequency components of the error on a given grid, i.e. solve

$$\hat{A}_h u_h = f_h \tag{29}$$

starting with some initial guess, u_h, to obtain a smoothed result, $\overline{u}_h$, on a grid with spacing h.

Restriction Transfer the defect computed on a finer grid to a coarser grid:

$$\hat{I}_h^{2h} r_h = r_{2h}, \tag{30}$$

where $\hat{I}_h^H$ is the restriction operator which maps functions on the grid with spacing h to the coarser grid with spacing, H (in this example, the spacing is doubled).

Prolongation Transfer the error computed on a coarser grid to a finer grid:

$$\hat{I}_{2h}^h e_{2h} = e_h, \tag{31}$$

where $\hat{I}_H^h$ is the prolongation operator which maps functions on the coarse grid with spacing H to the finer grid with spacing, H (in this example, the spacing is halved).

When these operations are combined in an appropriate order, the resulting multigrid approach can significantly improve convergence compared to applying an iterative solver on a single grid.

A simple two-grid multigrid iteration, starting with the initial approximation, $u_h^{(m)}$, and producing an improved approximation, $u_h^{(m+1)}$, can be summarized as follows:

1	Smooth approximation	$\hat{A}_h \overline{u}_h^{(m)} = f_h$
2	Compute defect	$\overline{r}_h^{(m)} = f_h - \hat{A}_h \overline{u}_h^{(m)}$
3	Restrict defect	$\hat{I}_h^{2h} \overline{r}_h^{(m)} = \overline{r}_{2h}^{(m)}$
4	Solve for error	$\hat{A}_{2h} e_{2h}^{(m)} = \overline{r}_{2h}^{(m)}$
5	Prolong error	$\hat{I}_{2h}^h e_{2h}^{(m)} = e_h^{(m)}$
6	Apply correction	$u_h^{(m+1)} = \overline{u}_h^{(m)} + e_h^{(m)}$

This cycle ($u_h^{(m)} \rightarrow u_h^{(m+1)}$) can be repeated until a convergence criterion is satisfied. Since the computation of the error in step 4 is of the same form as the linear equation (Eq. 26) we wish to solve, the two-grid cycle can be applied recursively, leading to multigrid scheme involving a hierarchy of progressively coarser grids.

With multiple levels of coarse grids, the basic smoothing (Eq. 29), restriction (Eq. 30) and prolongation (Eq. 31) steps can be combined to produce a variety of recursive schemes. One such scheme is the "V-cycle", illustrated in Fig. 1, in which a single two-grid iteration between the most coarse and second-most-coarse grids is employed. As described in section 3.1.1, V-cycle-type multigrid iterations are employed in DL_MG. Other schemes are possible, such the W-cycle and F-cycle which differ from the V-cycle in the arrangement of steps between grid levels—see Ref. 10 for further details.

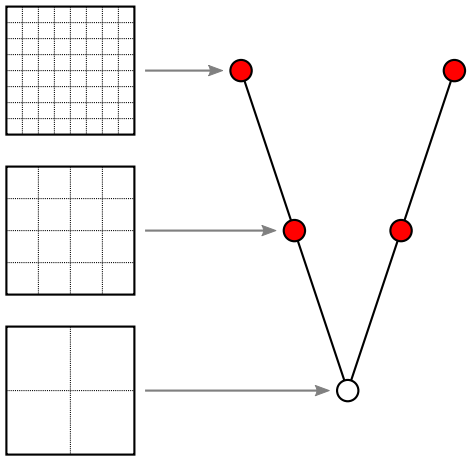

Figure 1: Illustration of a three-grid multigrid V-cycle

2.4 Defect correction

The representation of a continuous problem on a grid results in a "discretization error". In a finite difference method, this discretization error can be expressed as the remainder from truncating a Taylor series expansion of the function being discretized, e.g. for the forward difference derivative of $f(x)$ with grid spacing h:

$$\frac{f(x+h)-f(x)}{h} - f'(x) = \sum_{n=2}^{\infty} \frac{f^{(n)}(x)}{n!} h^{n-1} = O(h). \tag{32}$$

For small h, the leading term in the error in the discretized derivative in Eq. 32 is a first order polynomial in h. More generally, the discretization error is the difference between the exact solution to the continuous problem and the exact solution to the discretized problem, e.g. for the general linear problem of Eqs. 26 and 29, the discretization error is $|u - u_h|$.

The accuracy of a solution to a discretized problem is limited by the discretization error. To reduce this error, high-order finite difference approximations, in which the error asymptotically scales with higher powers of the grid spacing, h, can be employed. Such higher-order approximations have the advantage that relatively coarser grids can be used while maintaining the same level of accuracy compared to lower-order approximations.

Higher-order finite difference approximations are necessarily more complicated than lower-

order approximations, generally involving a greater number of terms. This corresponds to larger and/or more densely populated operator stencils (see appendix A), which can be challenging to implement in a manner which is computationally efficient. This is a particular issue for parallel implementations, where the application of larger stencil on coarse grids may result in the need to exchange halos which extend across multiple parallel processes.

It is possible to devise compact stencils (containing only points immediately adjacent to the central point) representing higher-order approximations than the usual 5-point 2-D or 7-point 3-D stencils. For example, using Mehrstellen discretization,[10,42] fourth-order accuracy is possible using compact 9-point 2-D and 19-point 3-D stencils—this compares favorably to the second-order accuracy obtained with the compact 5-point 2-D and 7-point 3-D discretizations of the Laplacian given in appendix A. However, these more complex stencils present additional challenges when implemented in a parallel solver. For example, the involvement of grid points at the corners of the stencil complicates halo exchange between parallel processes.

The high-order defect correction approach[10,43] provides a means by which approximations to high-order solutions can be obtained from a multigrid solver while avoiding the complexities of implementing a high-order multigrid scheme. This is achieved by iteratively correcting the solution obtained using a lower-order multigrid scheme using a higher-order discretization of the operator. The higher-order discretization of the operator is applied only on the fine grid on which multigrid solver deposits the solution, thus avoiding the difficulties associated with applying large, complicated, stencils in parallel on coarse grids.

The high-order defect correction procedure resembles the multigrid cycle described in section 2.3, in that an approximate error, e, is obtained by solving the defect equation,

$$\begin{aligned} \hat{A}\,(u - u') &= f - \hat{A}u' \\ \hat{A}e &= r, \end{aligned} \tag{33}$$

and is used to correct the approximate solution, u', i.e.

$$u = u' + e. \tag{34}$$

The multigrid and defect correction procedures differ in how the defect equation is solved. In a multigrid cycle the defect equation is solved on a coarser grid with a defect computed on a finer grid. In contrast, the high-order defect correction involves solving the defect equation with a lower-order discretization of the operator, using a defect computed using a higher-order discretization of the operator. In both cases, we solve the defect equation approximately (on a coarser grid, or with a lower-order operator discretization) so Eq. 34 yields an improved approximation, rather than the exact result.

Consider the high-order defect for an approximate solution, $u^{(i)}$, obtained via a multigrid scheme using a second-order-accurate operator, $\hat{A}_2$:

$$r_d^{(i)} = f - \hat{A}_d u^{(i)}. \tag{35}$$

The subscripts now refer to the order of accuracy of the operator, d, rather than grid spacing (in contrast to section 2.3) and the high-order operator $\hat{A}_d$ has $d > 2$. The defect equation may be approximately solved to second-order using the same second-order multigrid scheme,

$$\hat{A}_2 e_{2,d}^{(i)} = r_d^{(i)}, \tag{36}$$

to yield an approximation to the higher-order error $e_{2,d}^{(i)}$. The approximate error can then be used to correct the original approximation:

$$u^{(i+1)} = u^{(i)} + e_{2,d}^{(i)}. \tag{37}$$

This scheme can be applied iteratively, using the updated approximate solution $u^{(i+1)}$ to construct

a new defect (Eq. 35), and repeating the process until a convergence criterion is satisfied—the specific criteria available in DL_MG are described in section 3.

The iterative defect correction method outlined above is an effective method for reducing the discretization error in the solution obtained from a lower-order method. The scheme converges toward the solution for the higher-order problem,

$$\hat{A}_d u = f, \tag{38}$$

provided that

$$\rho(I - (\hat{A})_d^{-1} \hat{A}_{d'})) < 1 \tag{39}$$

is satisfied, where $\rho(M)$ is the spectral radius (i.e. largest eigenvalue) of a matrix M, I is the identity matrix and $\hat{A}_{d'}$ is the lower-order discretization of $\hat{A}$ ($d' < d$).[10]

For further details on the high-order defect correction, see Refs. 10 (ch.5), 43, and 16 (appendix B).

3 Implementation

3.1 The solver

3.1.1 Second order solver

The implementation of the second-order solver in DL_MG is based upon the "geometric multigrid" approach,[44] whereby the problem to be solved is expressed on a fixed hierarchy of coarsening grids, as described in section 2.3. This is distinct from the "algebraic multigrid" approach, which works directly with algebraic equations, rather than grids.[45]

The algorithms for DL_MG's geometric multigrid solver were selected following the standard recommendations for the Poisson and Poisson-Boltzmann equations (section 2.1) given in Refs. 10,46:

- Grid coarsening is achieved by doubling the grid-point separation in all dimensions at each multigrid level—this corresponds to the use of grids with spacing $2^n h$ with h the spacing of the finest grid ($n \in \mathbb{Z}$ and $n \geq 0$).

- The grid stencils used to apply the differential operator $\nabla \cdot \varepsilon \nabla$ on all grids are 3-D, 7-point second-order finite differences discretizations (see appendix A).

- Inter-grid transfers are performed with half-weight restriction and bilinear interpolation.

- Smoothing is performed using the Gauss-Seidel red-black (GS-RB) method (see, for example, Ref. 47).

Under the GS-RB scheme the grid is divided into two sets of points (red and black), with the points in each set depending only on the points in the other set. This has the advantage that the smoothing procedure can be applied to all the points in each set simultaneously, making the GS-RB smoother highly parallelizable.

The solver components described above can be used to construct an efficient solver for the Poisson and Poisson-Boltzmann equations with close to optimal computational scaling with respect to grid size, provided that the models used for the permittivity and charge density are smooth and without strong anisotropies.[10]

DL_MG was developed for use in large scale electronic structure calculations, the feasibility of which depends on the efficient use of parallel computing resources. The library was therefore designed to ensure good parallel performance on modern hardware, using widely adopted parallel frameworks (MPI and OpenMP) to ensure broad compatibility with existing electronic structure packages. In particular:

- Multigrid iterations are performed using the V-cycle (Fig. 1), as this generally recommended for parallel computations.[10,46]

- The distribution of global grid data among MPI processes is based upon a 3D Cartesian topology provided to DL_MG as an argument, allowing the solver to adopt the parallel data

decomposition of the calling program.

- Since grid coarsening is achieved by removing even index points in all directions, MPI communication is only necessary during grid transfer steps when dealing with points on the boundaries of the grid held on each MPI process.

The use of a sequence of progressively coarsening grids can be challenging for parallel implementations of multigrid. In particular, the number of active MPI ranks at each multigrid level can vary because as the grids become coarser, there are fewer points to share among parallel processes. Below a certain coarsening level, some MPI ranks may be assigned zero grid points. To deal with this variation in parallel data distribution, a separate MPI communicator, which includes only the active MPI ranks, is used to perform MPI communication at each multigrid level.

The communication of domain halos between MPI ranks—required in smoothing, prolongation and restriction steps—is done using non-blocking MPI sends and receives, allowing communication to be interleaved with useful computation. Since the 3-D differential operator is discretized as a 7-point stencil (appendix A), smoothing steps only require data exchange between MPI processes with local domains that share a face. For the inter-grid transfer steps (prolongation and restriction), points from MPI processes which hold local grids that only share edges or corners are necessary. The edge and corner points are efficiently communicated by means of ordered communication along axes between nearest neighbors[10]—this amounts to extending the size of the halos exchanged between MPI ranks with local domains which share faces so that the required edge and corner points are transferred along with the usual points along the shared face.

To take full advantage of modern multicore CPUs, DL_MG employs shared-memory parallelism within each MPI process via OpenMP threads. This is implemented as a single OpenMP parallel region, covering the V-cycle loop and the subroutine which builds the stencil coefficients.

The local grid held on each MPI process is decomposed into thread blocks and distributed to ensure equal work for all threads. The sizes of these blocks can be tuned to optimize cache utilization, and "first touch policy" (see, for example, Ref. 48) is used to ensure optimal memory access by OpenMP threads on NUMA architectures.

Communication between multi-threaded MPI processes is handled by the master thread, i.e. the so-called "funneled" mode.[49] This mode of communication was adopted to ensure portability between MPI implementations with differing support for multi-threaded communication—funneled mode is the simplest hybrid MPI/OpenMP mode which allows overlapping of computation and communication.[50] Data transfers between MPI buffers and halos are parallelized using OpenMP threads, employing "single" directives to assign one thread per local grid side to allow halos along each direction to be copied asynchronously.

Although DL_MG has been designed to take full advantage of hybrid MPI/OpenMP parallelism, support for MPI and OpenMP is not a requirement. When running calculations on a single workstation, it might be desirable to use only shared-memory parallelism. Alternatively, a distributed-memory parallelism only approach might be preferred when DL_MG is called from an application which is designed to spawn one MPI process per CPU core. DL_MG is flexible in this respect—the library can be compiled with or without MPI or OpenMP, and can therefore be applied in contexts where only one type of parallelism is desired (or none at all).

The algorithm used by DL_MG to solve the NLP-BE is based on a specialized inexact-Newton method.[51] In short, the linear multigrid solver is used to find an approximate solution of the linearized system of equations which correspond to a Newton iteration. A damping factor for the linear correction is also computed in order to ensure global convergence. See Ref. 51 for a detailed description of this approach (referred to as the "Damped-Inexact-Newton method").

For the SPE, GPE and P-BE, DL_MG uses the same general convergence test, based upon the norm of the residual:

$$|r^{(i)}| < \max(\tau^{\text{abs}}, \tau^{\text{rel}}|f|), \tag{40}$$

where $r^{(i)}$ is the residual at iteration i, f is the source term and τ^{abs}, τ^{rel}, are user-configurable absolute and relative convergence thresholds, respectively. The definition of the residual depends on the equation being employed—for linear equations (SPE, GPE, linearized Poisson-Boltzmann), the general form is Eq. 27, while for non-linear equations an extra non-linear term, $N(u)$, is added,

i.e.

$$r = f - \hat{A}u - N(u). \tag{41}$$

Using the maximum of the absolute and relative thresholds allows flexible control of convergence and can help avoid numerical issues when the source term is small.

The software library was developed in Fortran 95, using modules and derived data types for information encapsulation and to maintain a hierarchical structure. See appendix B for an introduction to the application programming interface and organization of the package, or the online documentation for a more detailed account.[52]

3.1.2 Defect correction

As described in section 2.4, the high-order defect correction[10,43] in DL_MG is applied on the fine grid, i.e. the grid on which input data is provided from the calling program. The second-order multigrid solver described in section 3.1.1 is used to approximately solve the defect equation (Eq. 36) for the residual computed using a high-order discretization of the differential operator. In practice, this is implemented as a loop, with the second-order solver repeatedly called to approximately solve the defect equation. In each iteration, the approximate potential is corrected using the second-order solution to the defect equation (Algorithm 1).

Algorithm 1 High-order defect correction

1: $i = 0$
2: Solve $\hat{A}_2 u^{(0)} = f$
3: **while** not converged **do**
4: Compute $r_d^{(i)} = f - \hat{A}_d u^{(i)}$
5: Solve $\hat{A}_2 e_{2,d}^{(i)} = r_d^{(i)}$
6: Correct $u^{(i+1)} = u^{(i)} + e_{2,d}^{(i)}$
7: $i = i + 1$
8: **end while**

The defect correction procedure is considered to have converged when the following criteria

are satisfied:

$$|u^{(i)} - u^{(i-1)}| < \max(\tau_u^{\text{abs}}, \tau_u^{\text{rel}}|u^{(i-1)}|), \tag{42}$$

$$|r_d^{(i)}| < \max(\tau_{r_d}^{\text{abs}}, \tau_{r_d}^{\text{rel}}|r_d^{(0)}|), \tag{43}$$

where the most-recently updated potential and defect are $u^{(i)}$ and $r_d^{(i)}$, the initial (uncorrected) defect is $r_d^{(0)}$, and where the absolute, $\tau_{u,r_d}^{\text{abs}}$, and relative, $\tau_{u,r_d}^{\text{rel}}$, convergence thresholds are user configurable.

The combination of these two conditions ensures that the iterative process does not stop too early due to temporary satisfaction of either condition—a truly converged solution will be converged with respect to both the residual and the error in the potential.

The use of absolute thresholds, τ^{abs}, ensures that convergence can be achieved in cases where the relative threshold is problematic, for example where $|u^{(i-1)}|$ or $|r_d^{(0)}|$ are small and it may be difficult to converge with respect to the relative threshold due to accumulated round-off errors in the procedure. The convergence thresholds and the maximum number of iterations have defaults set to values that were tuned for the class of problems solved in ONETEP[1] and CASTEP[53] calculations.

The differential operator $\hat{A}_d = [\nabla \cdot (\varepsilon\nabla)]_d$ used to compute the defect in Algorithm 1 is applied using 1-D finite difference representations of the first and second derivative operators. The overall operator can be trivially expressed in terms of these "bare" derivative operators by applying the product rule, yielding a high-order defect with the following form:

$$r_d^{(i)}(\mathbf{r}) = \alpha n(\mathbf{r}) - (\nabla_d \varepsilon(\mathbf{r})) \cdot (\nabla_d \phi(\mathbf{r})) - (\nabla_d^2 \varepsilon(\mathbf{r}))\phi^{(i)}(\mathbf{r}), \tag{44}$$

where $\alpha n(\mathbf{r})$ is the source term with α a constant which depends on the unit system (in atomic units it is -4π); $\phi^{(i)}$ is the approximate potential from the i^{th} defect correction iteration; and ∇_d, ∇_d^2 are d^{th} order finite difference discretizations the gradient and Laplacian operators. In the case of the P-BE, a further term is subtracted from the defect, the form of which depends on whether the linear or non-linear form of the equation is being solved.

The 3-D gradient and Laplacian operators (Eq. 44) used in the defect correction are expressed in terms of 1-D finite difference approximations to the first and second derivatives, i.e.

$$\nabla_d = \sum_{i=x,y,z} \left(\frac{\partial}{\partial x_i} \right)_d \tag{45}$$

$$\nabla_d^2 = \sum_{i=x,y,z} \left(\frac{\partial^2}{\partial x_i^2} \right)_d . \tag{46}$$

The stencils for these 1-D operators were derived automatically, using a computer algebra system[54] to perform the following procedure:

1. For a generic function $f(x)$ sampled at $n+1$ points x_i with equal spacing h, construct the unique $n-1$ order interpolating polynomial, $P(x)$.

2. Compute the symbolic k^{th} order derivative of the polynomial, $P^{(k)}(x) = \partial^k P(x)/\partial x^k$.

3. Evaluate $P^{(k)}(x_j)$ where x_j is one of the interpolation points, $\{x_i\}$, and simplify the expression.

This procedure yields general expressions of the form

$$P^{(k)}(x_j) = \frac{1}{h^k} \sum_{i=1}^{n+1} s_i f(x_i), \tag{47}$$

where h is the grid point spacing, $\{s_i\}$ are a set of constants and $\{f(x_i)\}$ are the values of the function at the interpolation points $\{x_i\}$, which include the point at which the derivative is taken, x_j.

These expressions describe "n^{th}-order" 1-D finite difference stencils, for taking the derivative at x_j, with coefficients, s_i, i.e.

$$\frac{1}{h^k} \left[\begin{array}{ccc} s_{j-n} & \cdots & s_n \end{array} \right]_h f(x) = \frac{1}{h^k} \sum_{i=j-n}^{n} s_i f(x_0 + ih), \tag{48}$$

where we have re-numbered the summation to make x_0 the point at which the derivative is taken. Using this scheme, arbitrarily large stencils can be constructed for taking the derivative at any of

the interpolation points used to construct the polynomial.

1-D stencils for the first and second derivatives of orders $4, 6, 8, 10$ and 12 are available in DL_MG. For each available stencil, the derivative can be taken at any of the grid points involved, i.e. the stencils for all possible forward, central and backwards differences are available. In periodic BCs, central differences stencils are always used, while under open BCs, forward and backward differences stencils are employed at the grid boundaries.

A note on the nomenclature: in this work, where we describe the 1-D stencils used in the defect correction (Eq. 48) as "n^{th}-order", we are referring to the order of the interpolating polynomial used to construct the stencil. We refer to all forward-, backward- and central-differences stencils derived from an interpolating polynomial of order-n as "n^{th}-order", regardless of the order of derivative being discretized. This is not the same as the order of the discretization error which we have previously referred to (e.g. Eq. 32), since this depends on the order of the derivative and also whether a given discretization benefits from the cancellation of terms when expanded in Taylor series.

The high-order 1-D discretizations of differential operators used in the defect correction have large stencils, which increase in size with the order of discretization—an n^{th}-order stencil will, in general, include contributions from $n + 1$ points. This poses a challenge when using distributed-memory parallelism, since applying these operator stencils at the boundary of the local domain requires the exchange of large halos between MPI processes. Where the local domain is narrow, halos may extend over the local grids on more than one MPI rank, increasing the complexity of communication. Handling these extended halos efficiently is the main difficulty in computing derivatives with higher order discretization over a distributed domain.

To enable efficient exchange of extended halos during the defect correction procedure, DL_MG builds two maps on each MPI process which describe the data which must be sent to and received from other MPI processes. Each of these maps is essentially a list of data blocks, containing the MPI rank and the relevant global index coordinates of the block of halo data to be sent or received. The halo exchanges are done with non-blocking send-receive MPI communication, allowing data

to be dynamically copied to halo arrays as it is received.

The implementation of the high-order defect correction in DL_MG supports fully open (Dirichlet), fully periodic and mixed open/periodic BCs. As with the second-order multigrid solver the high-order defect correction is accelerated using hybrid MPI/OpenMP parallelism, employing the same 3-D Cartesian topology for decomposition of the global grid among MPI processes. Within each MPI process, the local computation of the derivatives used to construct the defect (Eq. 44) is parallelized using OpenMP threads.

Algorithm 1 describes the implementation of the defect correction procedure in DL_MG for the simplest case—correcting the second-order solution to the linear forms of the Poisson equation (SPE and GPE, Eqs. 1 and 7). In more complicated cases, the algorithm is modified. For example, for difficult-to-converge problems, the algorithm can be augmented with an error damping procedure. This is achieved by damping the correction of the potential (Algorithm 1, line 6)

$$u^{(i+1)} = u^{(i)} + s\, e^{(i)}_{2,d} \tag{49}$$

with $s \in (0, 1]$, such that

$$|A_d u^{(i+1)} - f| < |A_d u^{(i)} - f|\,, \tag{50}$$

i.e. the defect for the corrected potential, $u^{(i+1)}$ is smaller than the defect for the uncorrected potential $u^{(i)}$.

In practice, the damping factor, s, is systematically reduced (starting from $s = 1$) by a fraction, $q < 1$, until Eq. 50 is satisfied—see Algorithm 2. If s becomes smaller than a prescribed value the entire defect correction process is halted with an error. This procedure can be enabled with an optional argument of the solver subroutine, but should be used only when the standard defect correction procedure does not converge, since it involves costly repeated evaluations of the high-order defect.

Algorithm 1 is also modified when solving the P-BE. While the LP-BE (Eq. 14) may be solved using Algorithm 1 or Algorithm 2, with a modified linear operator $\hat{A}_d$, the NLP-BE requires further

Algorithm 2 High-order defect correction with error damping ($q < 1$)

$i = 0$
Solve $\hat{A}_2 u^{(0)} = f$
while not converged **do**
    Compute $r_d^{(i)} = f - \hat{A}_d u^{(i)}$
    Solve $\hat{A}_2 e_{2,d}^{(i)} = r_d^{(i)}$
    $s = 1$
    repeat
        Correct $u^{\text{damp}} = u^{(i)} + s\, e_{2,d}^{(i)}$
        Compute $r_d^{\text{damp}} = f - \hat{A}_d u^{\text{damp}}$
        $s = q\, s$
    until $|r_d^{\text{damp}}| < |r_d^{(i)}|$
    $u^{(i+1)} = u^{\text{damp}}$
    $i = i + 1$
end while

modification of the scheme.

In Algorithms 1 and 2, the linear defect equation (Eq. 36) is solved approximately using the second-order multigrid solver. However, the defect for the NLP-BE (Eq. 12) includes a non-linear term (see Eq. 41), and thus cannot satisfy this linear equation. To overcome this difficulty, the defect equation is solved for the P-BE linearized at the current approximation to the potential, i.e.

$$\hat{A}_2 e_{2,d}^{(i)} + N'(u^{(i)}) e_{2,d}^{(i)} = r_d^{(i)}, \tag{51}$$

where $N'(u^{(i)})$ is the first derivative of the non-linear Boltzmann term with respect to the potential, u, evaluated at the current approximation to the potential, $u^{(i)}$. Note that this linearization of the Poisson-Boltzmann equation is distinct from the linearization in Eq. 14, which is linearized for potentials close to zero, rather than close to the current approximation of the potential.

3.2 Electronic structure software

DL_MG exposes an API (appendix B) which allows an external program to call solver routines from the library in the context of a larger procedure, for example an electronic structure calculation. The changes required to use the solver in an external program are small. First, the build procedure for the program should be modified so that the library is appropriately linked. Second, calls to DL_MG initialization and solver procedures, with appropriate arguments, should be inserted into the main program where required (see appendix B). Other modifications may be necessary to transform the quantities used by DL_MG to a suitable form, if they are not stored in a compatible representation in the external program. For example, if the charge density is stored in a form other than a regular grid, then it must be converted to this format before being provided to DL_MG.

Since the creation of DL_MG,[55] the library has been interfaced with several electronic structure codes, notably ONETEP,[1] CASTEP[53] and PSI4.[56] In this work, we present results from ONETEP calculations employing DL_MG (see section 4.2). For results obtained with DL_MG in CASTEP and PSI4, see Refs. 57 and 58, respectively.

When called from an external program, DL_MG will typically need to operate within additional constraints imposed by the program. For example, the nature of the overall implementation of the external program may require that specific grid sizes, numbers of parallel processes/threads or MPI topologies are employed. This is in contrast to the synthetic tests considered in sections 4.1.1 and 4.1.2, where the problem size and number of parallel processes could be varied flexibly.

In the specific case of ONETEP, the size of the fine grid passed to the solver is related to the kinetic energy cutoff used to construct the underlying psinc basis.[59,60] The MPI topology over which this global grid is distributed is 1-D, with the grid divided into "slabs" along one coordinate direction.[39] The total number of MPI processes is also restricted—this cannot exceed the number of atoms used in the calculation.

The additional constraints associated with a ONETEP calculation pose little difficulty for DL_MG. The 1-D MPI topology can simply be provided to DL_MG via an appropriately set-up MPI communicator. The sizing of the grid requires a little more care, and for each Cartesian

direction i should satisfy the condition

$$N_i = q_i 2^{n_i} + \delta_i, \tag{52}$$

where N_i is the number of grid points; n_i and q_i are positive integers ($q_i \leq 20$); and δ_i is 1 or 0 for open or periodic BCs. Roughly speaking, q_i determines the size of the coarsest grid level, while n_i determines the number of multigrid levels. For the typical grid sizes encountered in ONETEP ($N_i = 10^2$ to 10^3), these conditions can be satisfied by passing to DL_MG a slightly truncated grid (for open BCs) or by slightly increasing the scale factor used to produce the fine grid (for periodic BCs).

The implementation of an implicit solvent model in an electronic structure code involves more than simply interfacing the code with an efficient solver for the GPE or P-BE. The details of this implementation will depend upon the solvent model and the underlying theoretical formalism employed in the electronic structure package. For example, the dielectric permittivity $\varepsilon(\mathbf{r})$ and ion accessibility functions $\lambda(\mathbf{r})$ are model-dependent and must be constructed by the electronic structure code. Similarly, any method of accounting for the non-electrostatic components of solvation (e.g. cavitation, dispersion-repulsion) must be done outside of the Poisson solver, which deals only with the electrostatic terms. For an account of the implementation of an implicit solvent model in ONETEP which includes electrostatic and non-electrostatic components (based on Fattebert and Gygi's electrostatic model[12,22] described in section 2.2), see Refs. 15,16.

4 Results

4.1 Solver testing

4.1.1 Numerical validation

DL_MG includes a comprehensive suite of self-tests which allows results computed using the solver to be validated against known analytic solutions to the SPE, GPE and P-BE. The test suite is

intended to prevent regressions during code development, but can also be used to rigorously study the accuracy and convergence of the solver.

To examine the numerical behavior of DL_MG, two tests which model physical systems relevant to chemical physics were selected from the test suite and run with varying grid sizes and orders of finite differences used in the defect correction.

All calculations presented in this section were run in parallel (8 MPI processes, 4 OpenMP threads per process) on a single workstation, using a development version of DL_MG version 2.0 compiled using gfortran 5.3.1[61] and linked to the Intel MPI library 2017.[62]

The first test represents the type of problem encountered in electronic structure calculations where an implicit solvent is represented using a smoothly varying dielectric function (as in Fattebert and Gygi's electrostatic solvent model and variants,[12,22] described in section 2.2). Since the dielectric function is general and non-homogeneous, this requires the solution of the GPE (Eq. 7). We shall refer to this test case as "`erf_eps`", as in DL_MG's test suite.

The second test models the interaction of an ionic solution with a charged surface, for example an electrode immersed in an electrolyte. This situation may be studied by solution of the P-BE (Eq. 12), representing the solvent via a homogeneous dielectric permittivity and using Boltzmann distributions to describe the concentrations of mobile ions in solution. This test will be referred to as "`pbez`", following the name used in DL_MG's test suite.

The `erf_eps` test is based upon the model system proposed by Fisicaro *et al.* in Ref. 25 to represent an isolated solute embedded in implicit solvent. In this situation, the overall electrostatic potential due to the solute charge and polarization of the dielectric medium is obtained by solving the GPE for the solute charge $n(\mathbf{r})$, and the dielectric permittivity $\varepsilon(\mathbf{r})$, which switches smoothly between a bulk value $\varepsilon(\mathbf{r}) = \varepsilon_\infty$ far from the solute and vacuum value $\varepsilon(\mathbf{r}) = 1$ close to the solute.

Defining the electrostatic potential as a normalized Gaussian function,

$$\phi(\mathbf{r}) = \left(\frac{1}{2\pi\sigma^2}\right)^{3/2} \exp\left(-\frac{|\mathbf{r}-\mathbf{R}|^2}{2\sigma^2}\right) \tag{53}$$

and using a dielectric permittivity constructed using an error function,

$$\varepsilon(\mathbf{r}) = 1 + \frac{(\varepsilon_\infty - 1)}{2}\left[1 + \mathrm{erf}\left(\frac{|\mathbf{r} - \mathbf{R}| - d_0}{\Delta}\right)\right] \tag{54}$$

the corresponding charge density can be derived analytically, i.e.

$$\begin{aligned} n(\mathbf{r}) = -\frac{1}{4\pi}\frac{\phi(\mathbf{r})}{\sigma^2}\Bigg[\varepsilon(\mathbf{r})\left(\frac{|\mathbf{r} - \mathbf{R}|^2}{\sigma^2} - 3\right) \\ -\frac{(\varepsilon_\infty - 1)|\mathbf{r} - \mathbf{R}|}{\pi^{1/2}\Delta}\exp\left(-\left(\frac{|\mathbf{r} - \mathbf{R}| - d_0}{\Delta}\right)^2\right)\Bigg]. \end{aligned} \tag{55}$$

The Gaussian potential, error-function-based permittivity and corresponding density used in the `erf_eps` model are defined by a set of parameters: the center of the Gaussian potential and dielectric cavity $\mathbf{R}$; the permittivity in the bulk solvent ε_∞; the distance of the center of the transition region of the permittivity (where $\varepsilon(\mathbf{r}) = (\varepsilon_\infty + 1)/2$) from the center of the Gaussian potential, d_0; and parameters controlling the widths of the Gaussian potential, σ, and the transition region of the permittivity, Δ.

We examined the accuracy of the solutions produced by DL_MG for the `erf_eps` model, using the parameters suggested in Ref. 25 ($\sigma = 0.5\,a_0$, $d_0 = 1.7\,a_0$, $\Delta = 0.3\,a_0$, $\varepsilon_\infty = 78.36$) with a cubic simulation cell with side length $10\,a_0$ and for three grid sizes: 209^3, 305^3 and 401^3. Dirichlet BCs were used in all directions, with $\phi(\mathbf{r})$ set to zero at the boundaries. The accuracy of the solution, defined as the maximum difference between the analytic and numerical solutions over all grid points, with increasing finite difference order in the high-order defect correction is plotted in Fig. 2.[63]

Fig. 2 clearly demonstrates that the maximum error in the solution for the `erf_eps` rapidly decreases as the order of finite differences used in the high-order defect correction procedure is increased. The magnitude of the error for a given finite difference order generally decreases as the grid size is increased, in line with expectations since increasing the number of grid points for fixed simulation cell dimensions implies a finer grid. For the 305^3 and 401^3 grids, the maximum

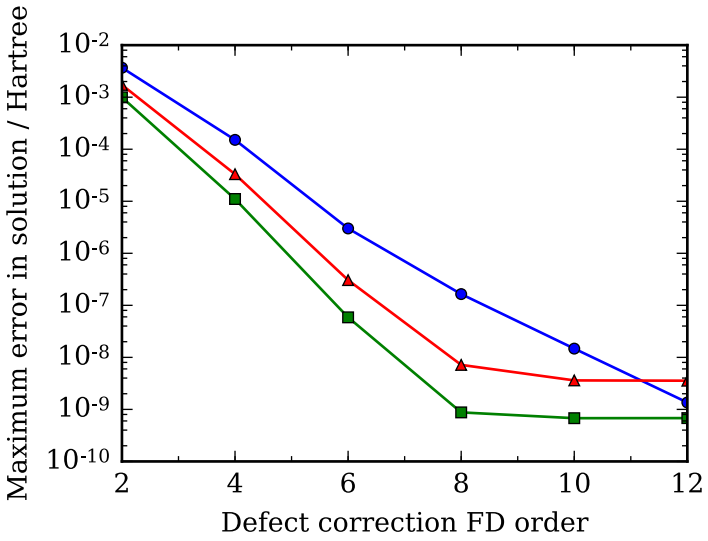

Figure 2: Maximum error in the numerical solution for the `erf_eps` test, measured against the analytic solution (Eq. 53), for increasing orders of finite differences used in the high-order defect correction. The error is plotted for three grid sizes: 209^3 (blue circles), 305^3 (red triangles) and 401^3 (green squares). A finite difference order of 2 implies that no high-order defect correction was performed, i.e. the error is for the uncorrected second-order multigrid solution. The functional forms and parameters used to construct the `erf_eps` model for these calculations are described in section 4.1.1.

error appears to plateau above 8^{th}-order finite differences, while the 209^3 grid does not exhibit this effect—for 12^{th}-order finite differences, the error from the 209^3 grid is slightly smaller than the error for the 305^3 grid. This difference in behavior is likely to be related to the number of defect correction iterations required to achieve convergence (based on the tests described in Eqs. 42 and 43). All the calculations run with 209^3 grid points required 3 defect correction iterations to converge, while the larger grids required 2 defect correction iterations. As a consequence, the final defect and error norms ($|r_d^{(i)}|$ and $|u^{(i)} - u^{(i-1)}|$ in Eqs. 42 and 43) for 209^3 are smaller than the corresponding norms for 305^3.

The difference in convergence behavior observed for different grid sizes at high finite difference orders is interesting, however the key result illustrated by Fig. 2 is that the application of the high-order defect correction can reduce the maximum error in the solution by several orders of magnitude. For the grids tested here, the maximum error for 12^{th}-order finite differences was at least a factor of $\sim 10^{-6}$ smaller than the maximum error for the second-order multigrid solver alone.

The model system used in the `pbez` test is a 1:1 salt solution (e.g. NaCl in H_2O) in contact with an infinite planar surface of homogeneous charge. Assuming that the ionic concentrations are described by Boltzmann distributions (Eq. 11), the electrostatic potential for the system can be found by solving the P-BE. The problem considered in the `pbez` test is further simplified by assuming a homogeneous dielectric permittivity, singly charged ionic species, and that the accessibility function $\lambda(\mathbf{r}) = 1$ everywhere. In this case, the electrostatic potential can be found by solving a simplified P-BE in 1-D,

$$\frac{\partial^2 \phi(z)}{\partial z^2} = -\frac{4\pi c_0}{\varepsilon_\infty} \left[\exp\left(-\beta\phi(z)\right) - \exp\left(\beta\phi(z)\right)\right], \tag{56}$$

where c_0 is the bulk concentration of the salt, ε_∞ is the homogeneous permittivity of the solvent, $\beta = 1/(k_\mathrm{B}T)$ and the z coordinate direction is normal to the plane of the charged surface. The potential due to the charged surface enters into the equation via boundary conditions, i.e.

$$\phi(z) = \begin{cases} \phi_\mathrm{surf}, & z = 0 \\ 0 & z \to \infty, \end{cases} \tag{57}$$

where ϕ_surf is the value of the potential at the planar surface.

The 1-D P-BE in Eq. 56 can be solved analytically for the BCs of Eq. 57. For the general non-linear case, the electrostatic potential is

$$\phi(z) = 2\beta^{-1} \ln\left(\frac{\exp\left(\beta\phi_\mathrm{surf}/2\right) + 1 + \left(\exp\left(\beta\phi_\mathrm{surf}/2\right) - 1\right)\exp\left(-\kappa z\right)}{\exp\left(\beta\phi_\mathrm{surf}/2\right) + 1 - \left(\exp\left(\beta\phi_\mathrm{surf}/2\right) - 1\right)\exp\left(-\kappa z\right)}\right) \tag{58}$$

with the inverse Debye length for singly-charged 1:1 ionic solutions

$$\kappa = \left(\frac{8\pi c_0}{\varepsilon_\infty k_\mathrm{B} T}\right)^{1/2}. \tag{59}$$

See Ref. 64 for details of the derivation of Eq. 58.[65]

We used the `pbez` test to examine the accuracy of DL_MG when solving the non-linear

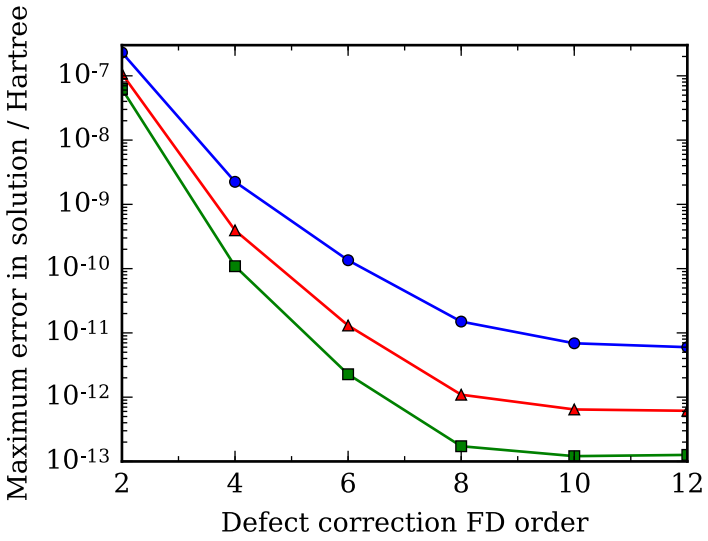

Figure 3: Maximum error in the numerical solution for the `pbez` test, measured against the analytic solution (Eq. 58), for increasing orders of finite differences used in the high-order defect correction. The error is plotted for three grid sizes: 208×208×209 (blue circles), 304×304×305 (red triangles) and 400×400×401 (green squares). A finite difference order of 2 implies that no high-order defect correction was performed, i.e. the error is for the uncorrected second-order multigrid solution. The functional forms and parameters used to construct the `pbez` model for these calculations are described in section 4.1.1.

P-BE. The test was performed in a cubic simulation cell with side length $10\,a_0$, with $\varepsilon_\infty = 80$, $c_0 = 0.1\,\mathrm{mol\,dm^{-3}}$, $T = 300\,\mathrm{K}$, and $\phi_{\mathrm{surf}} = 200\,\mathrm{mV}$. Three grid sizes were used: $208 \times 208 \times 209$, $304 \times 304 \times 305$ and $400 \times 400 \times 401$.[66] The accuracy of the numerical solution (defined as for the `erf_eps` test case) with increasing finite difference order in the high-order defect correction is plotted in Fig. 3.

The general trend for rapid reduction in the maximum error for increasing orders of finite differences seen for the `erf_eps` test (Fig. 2) is reproduced in Fig. 3. Similarly, as observed for the `erf_eps` test, the absolute magnitude of the error for a given order of finite differences decreases as the number of grid points is increased. This effect is more consistent for `pbez` than `erf_eps`, with smaller grids yielding larger errors for all orders of finite difference. Again, this can be attributed to the number of defect correction iterations required to achieve convergence—for `erf_eps` this was different for 209^3 versus the other grid sizes, while for `pbez` this is the same for all grid sizes. Unlike for `erf_eps`, the norms of the final defect $|r_d^{(i)}|$, and error $|u^{(i)} - u^{(i-1)}|$, at each order of

finite differences consistently decrease for increasing grid sizes.

As noted for the `erf_eps` test, the key result of interest is overall reduction in this error achieved by applying the defect correction. The results for the `pbez` test indicate that, as with `erf_eps`, the error in the solution may be very significantly decreased by application of the defect correction. For the grid sizes used in Fig. 3, the maximum error is at least $\sim 10^{-5}$ times smaller with 12^{th}-order finite differences than for the second-order multigrid solver without defect correction.

4.1.2 Performance tests

DL_MG was originally conceived for use in large-scale electronic structure calculations on massively parallel computers. For the solver to fulfill this purpose, it must be able to scale efficiently with problem size and number of parallel processors. To examine the scaling of computational cost in these two circumstances for problems of the type which would be encountered in electronic structure calculations, we used the `erf_eps` and `pbez` test cases described in section 4.1.1. Using these synthetic test cases the number of processors and size of the problem could be varied systematically and the performance of DL_MG studied in isolation.

Figs. 4 and 5 plot the scaling of execution time with respect to problem size, for cubic grids with between 577^3 ($\sim 10^8$) and 1345^3 ($\sim 10^9$) grid points (the grids used for `pbez` are less one grid point in the x and y directions, for the reasons explained in section 4.1.1). These calculations were run across 6 nodes on the EPSRC MMM Hub "Thomas" supercomputer[67] with 64 MPI processes and 2 OpenMP threads per process, using a development version of DL_MG version 2.0 compiled with gfortran 4.9.2[61] and linked to the Intel MPI library 2017.[62] The defect correction was performed with 12^{th}-order finite differences for all grid sizes and the parameters used to construct the models were as described in section 4.1.1.

For `erf_eps` (Fig. 4), the total computational cost and the cost attributed to the multigrid solver and high-order derivative computation increases linearly with respect to the number of grid points, N_{grid}, for the grid sizes used. In addition, for each of the components of the total cost plotted (second-order multigrid solver, computing high-order derivatives and the communication

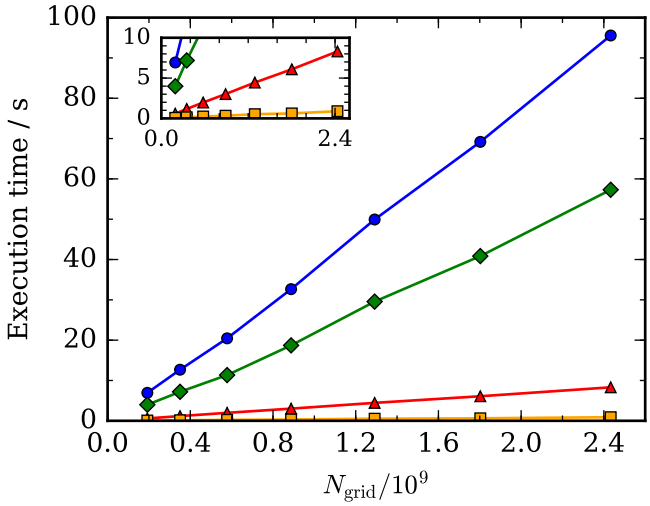

Figure 4: Execution time to reach solution for the `erf_eps` test for increasing problem size. The total time (blue circles) is plotted alongside time spent in the second-order multigrid solver (green diamonds) and computing high-order derivatives (red triangles). The portion of the time for high-order derivative computation spent preparing and communicating halo data between MPI processes is also plotted (yellow squares). The plotted values are minimum times taken over 5 repetitions, where the time recorded for each repetition is the maximum over all MPI processes. The functional forms and parameters used to construct the `erf_eps` model for these calculations are as for Fig. 2 (see section 4.1.1). The computational details of these calculations are described in section 4.1.2.

of high-order derivative halo data), the cost is seen to increase linearly with respect to grid size. The cost of the second-order multigrid solver dominates the overall computational cost, suggesting that future work to optimize the performance of DL_MG should focus upon the core multigrid solver, rather than the high-order defect correction.

The scaling of the computational cost per V-cycle for geometric multigrid is known to be $O(N_{\mathrm{grid}})$ (see Ref. 10 for a derivation of this). The overall cost of obtaining a high-order defect-corrected solution to the Poisson equation from DL_MG would be expected to exhibit $O(N_{\mathrm{grid}})$ scaling, as observed in Fig. 4, only if the number of defect correction and multigrid V-cycles is constant and independent of N_{grid}. For the grid sizes considered in Fig. 4, this was generally the case—2 defect correction iterations were required for all grid sizes, while the number of multigrid V-cycles for each of these defect iterations was constant across all grid sizes (6 and 3).

A detailed theoretical convergence analysis of the algorithms employed in DL_MG is beyond

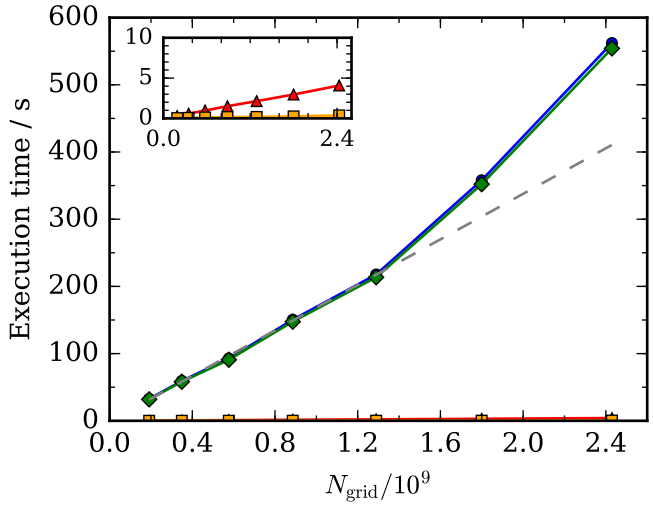

Figure 5: Execution time to reach solution for the `pbez` test for increasing problem size. The quantities plotted are as in Fig. 4, with the addition of a least-squares linear fit to the total execution times for the five smallest grid sizes (dashed gray line). The functional forms and parameters used to construct the `pbez` model for these calculations are as for Fig. 3 (see section 4.1.1). The computational details of these calculations are described in section 4.1.2.

the scope of this work. Nevertheless, it is clear from Fig. 4 that the cost to obtain a defect-corrected solution to the GPE for the `erf_eps` test scales linearly with respect to grid size, within the range of grid sizes tested. Given that the `erf_eps` test is designed to mimic the situation of an isolated molecule in implicit solvent, and the grid sizes used in electronic structure calculations are typically in the range of grids tested here, it is likely that $O(N_{\mathrm{grid}})$ scaling would also apply in practical implicit solvent calculations.

In Fig. 5, the overall execution time for `pbez` test is $\sim$ 5 to 6 times larger than for `erf_eps` for a given grid size.[68] In this case close-to-linear scaling of computational cost with respect to N_{grid} is observed, though the overall scaling is less clear than for `erf_eps`.

As described in section 3.1.2, DL_MG obtains defect-corrected solutions to the NLP-BE by linearizing the defect equation for the NLP-BE at the current approximation to the potential (Eq. 51). In this scheme, the initial second-order solution to the NLP-BE is obtained by the inexact-Newton method outlined in section 3.1.1. Consequently, there are three iterative procedures to consider in the `pbez` test—the second-order multigrid solution of linearized versions of the P-BE, the

inexact-Newton procedure, and the high-order defect correction.

As for `erf_eps`, the number of defect correction iterations required to satisfy the convergence tests in `pbez` (Eqs. 42 and 43) is independent of grid size—this is 1 iteration for all grid sizes tested. Similarly, the number of iterations required to converge the inexact-Newton procedure was 6 for all grid sizes. Interestingly, the number of V-cycle iterations required to obtain a second-order multigrid solution increased with grid size. For the initial second-order solution (within the inexact-Newton method), 4 iterations are required for the 4 smallest grids, but for larger grids, this increases with grid size, rising to 8 for the largest grid. Similarly, for the approximate second-order solution of the high-order defect equation, 7 iterations are required for all grids except the two largest, which require 8 and 10 V-cycle iterations. This explains why total execution times for the largest grids in Fig. 5 are somewhat greater than would be expected for a linear fit to the first 5 points. This is illustrated in the figure by the inclusion of a least-squares linear fit to the total execution times for all but the two largest grids.

The scaling of computational cost for the `erf_eps` and `pbez` tests with respect to number of parallel processes for a fixed problem size (i.e. strong scaling) is plotted in Figs. 6 and 7. These calculations were performed on grids with 1089^3 and $1088 \times 1088 \times 1089$ grid points for `erf_eps` and `pbez`, respectively, and were run on the EPSRC MMM Hub "Thomas" supercomputer with between 8 and 216 MPI processes and 1, 2, or 4 OpenMP threads per process. The global grid data was divided equally along each coordinate direction for distribution to MPI processes, so each process held an equal (or near-equal) cuboid portion of the grid. For all calculations, 12^{th}-order finite differences were used and the parameters for constructing the model systems were as described in section 4.1.1.

The parallel speedup data presented in Figs. 6 and 7 indicates that significant speedups can be achieved by increasing the number of processes. For `erf_eps` (Fig. 6), the speedup with respect to number of MPI processes is near-linear for all N_{MPI}, N_{OMP} combinations (where N_{MPI} and N_{OMP} are the total number of MPI processes and number of OpenMP threads per process, respectively). The prefactor for the scaling is less than one, which implies that in this regime, the addition of each

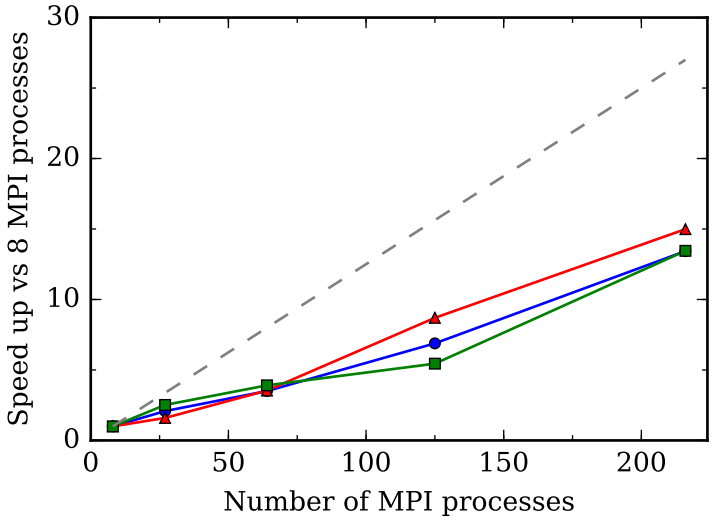

Figure 6: Parallel speedup for the `erf_eps` test on a 1089^3 grid for increasing numbers of MPI processes. Speedup is plotted for 1 (blue circles), 2 (red triangles) and 4 (green squares) OpenMP processes per MPI process, and is with respect to the calculation performed with 8 MPI processes (and the corresponding number of OpenMP threads per process). The plotted speedup values are calculated using the minimum total calculation time over 5 repetitions, where the time recorded for each repetition is the maximum over all MPI processes. The ideal speedup (i.e. $N_{\mathrm{MPI}}/8$) is plotted as a gray dashed line. The functional forms and parameters used to construct the `erf_eps` model for these calculations are as for Fig. 2 (see section 4.1.1). The computational details of these calculations are described in section 4.1.2.

MPI process offers a constant, but less-than-ideal speedup. The difference in speedup obtained using different number of OpenMP threads per MPI process is small, though for higher-core counts, it appears that 2 OpenMP threads offers the best speedup per additional MPI process.

The strong-scaling behavior of the `pbez` test is more complicated than for `erf_eps`. Fig. 7 shows that the speedup achieved for a given number of MPI processes, $S(N_{\mathrm{MPI}})$, is strongly dependent on the number of OpenMP threads per process. With 2 and 4 threads per process, the scaling behavior is very good. Near-ideal speedup is observed for 2 and 4 OpenMP threads per process for up to 125 MPI processes. The overall trend in this case is for a slow decrease in the performance improvement offered per additional parallel process (i.e. decreasing parallel efficiency, $S(N_{\mathrm{MPI}})/N_{\mathrm{MPI}}$), in line

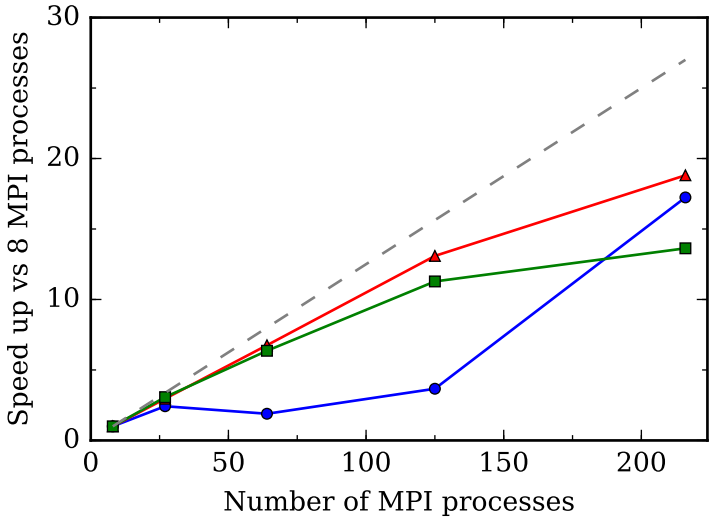

Figure 7: Parallel speedup for the `pbez` test on a $1088 \times 1088 \times 1089$ grid for increasing numbers of MPI processes. Speedup with respect to 8 MPI processes for 1 (blue circles), 2 (red triangles) and 4 (green squares) OpenMP threads per process is plotted, with the values calculated as for Fig. 6. The functional forms and parameters used to construct the `pbez` model for these calculations are as for Fig. 3 (see section 4.1.1). The computational details of these calculations are described in section 4.1.2.

with Amdahl's law.[69,70] To illustrate this, Fig. 8 presents a least-squares fit to Amdahl's law,

$$S_{\text{Amdahl}}(S, p) = \frac{1}{(1-p) + p/S_{\text{ideal}}}, \tag{60}$$

for 2 OpenMP threads per MPI process, under the assumption that the fraction of the execution time amenable to parallelization, p, experiences ideal speedup, S_{ideal}. This fit yielded a value of $p = 0.98367$.

While the parallel speedup for the `pbez` test with 2 and 4 OpenMP threads per MPI process follows the expected trend, with 1 OpenMP thread per MPI process the behavior is more erratic, with the speedup for 64 and 125 MPI processes substantially lower than would be expected. For 64 MPI processes with 1 OpenMP thread per process, the speedup is actually lower than for 27 MPI processes. This appears to be a consequence of contention for hardware resources.

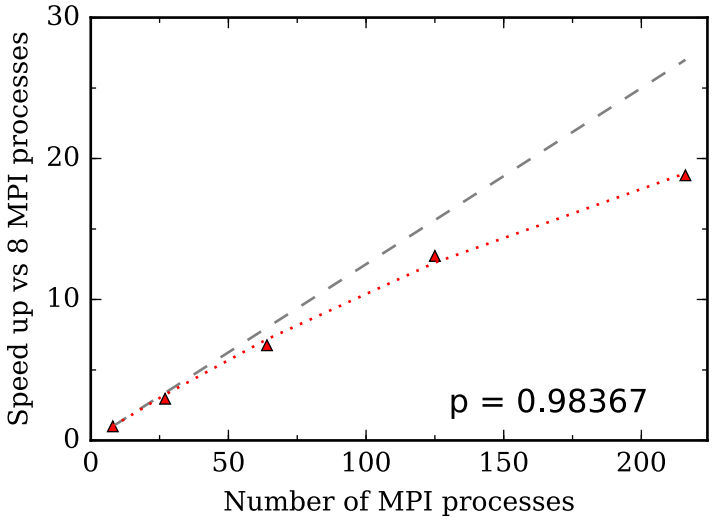

Figure 8: Parallel speedup for the `pbez` test on a $1088 \times 1088 \times 1089$ grid for increasing numbers of MPI processes. The ideal speedup with respect to 8 MPI processes (gray dashed line) and measured speedup with for 2 OpenMP processes per MPI process (red triangles) are plotted, as in Fig. 7. Additionally a least-squares fit of the 2 OpenMP thread data to Amdahl's law (Eq. 60) is plotted (red dotted line, with $p = 0.98367$).

For the `pbez` test, the number of compute nodes allocated for the problem was

$$\max(2, \text{roundup}(N_{\text{MPI}} \times N_{\text{OMP}}, 24)/24), \tag{61}$$

i.e. the next nearest multiple of 24 to the number of cores required, with a minimum of 2 nodes (roundup(x, y) rounds x up to the next multiple of y). Multiples of 24 were used since each node on the EPSRC MMM Hub "Thomas" machine on which these calculations were performed had 24 physical cores, while a minimum of 2 compute nodes was necessary because the memory requirements to run `pbez` on a $1088 \times 1088 \times 1089$ grid exceeded the memory available on a single node. While this represents a realistic allocation of resources, it results in discrepancies in the amount of resources available per MPI process for different numbers of OpenMP threads. For example, with 64 MPI processes, the amount of nodes requested is 3 (72 cores), 6 (144 cores) and 11 (264 cores) with 1, 2 and 4 OpenMP threads per process, respectively. The hardware resources available per MPI process are substantially less for 1 OpenMP thread and thus these resources will

be more contested for operations which occur on a per-process (not per-thread) basis (e.g. MPI communication).

To verify that the unusual speedup behavior for 1 OpenMP thread was due to more contested resources, we repeated the calculations presented in Fig. 7, but artificially allocated identical numbers of compute nodes for tests with 1, 2 and 4 OpenMP threads per process. In this case, the poor speedup for 1 OpenMP thread per process vanished, yielding instead the expected trend resembling the 2 and 4 OpenMP thread lines plotted in Fig. 7.

Overall, Figs. 6 and 7 demonstrate that DL_MG efficiently scales from 10s to 100s of processor cores, yielding significant performance improvements at typical core counts used in parallel electronic structure calculations. 2 OpenMP threads per process offers the best speedup in these particular tests, and use of > 1 OpenMP thread per MPI process is recommended to avoid issues with contention for hardware resources, as seen in Fig. 7.

4.2 Electronic structure calculations

In this section, we consider the numerical accuracy and computational performance of DL_MG when used as a Poisson solver in ONETEP,[1] an electronic structure package designed to perform calculation with a cost that scales linearly with the number of atoms, N.

All DFT results presented in this section were computed using the PBE exchange-correlation functional[71,72] and norm-conserving pseudopotentials from the Rappe-Bennett pseudopotential library[73] (GGA-optimized).[74]

To evaluate the numerical accuracy of DL_MG in ONETEP, we performed single point DFT energy calculations for a periodic 448 atom graphene sheet in vacuum, using DL_MG to solve the SPE. These calculations were performed on the ARCHER UK national supercomputer[75] with 48 MPI processes and 4 OpenMP threads per process, using a binary compiled using gfortran 5.1[61] and linked to FFTW[76,77] and the Cray MPI libraries.[78]

The 448 atom graphene sheet was generated with a C-C bond length of 1.43 Å and made periodic in the xy plane of a 34.31 Å $\times$ 34.67 Å $\times$ 31.75 Å cell. All DFT calculations were fully

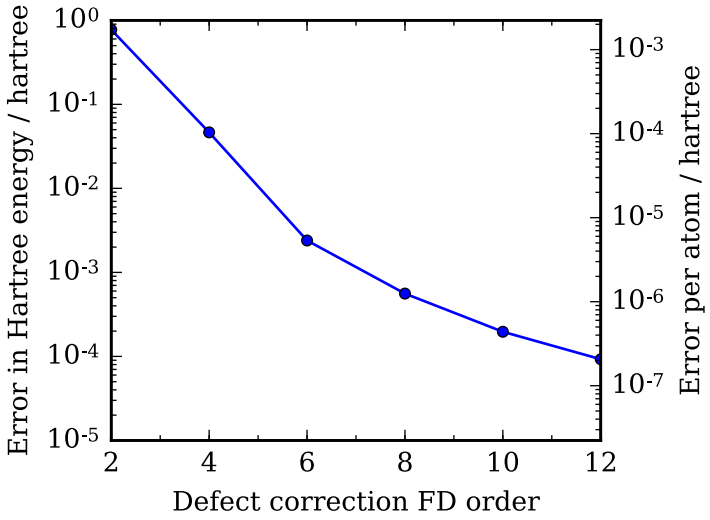

Figure 9: Absolute error in the Hartree energy for a 448 atom graphene sheet computed using DL_MG to solve the SPE in ONETEP. The total error and error per atom are plotted as a function of order of finite differences used in the defect correction procedure. The error is calculated with respect to the Hartree energy obtained from a reciprocal space solution to the SPE (Eq. 4).

self-consistent and the SPE was solved (for multigrid and in reciprocal space) on a $256 \times 264 \times 240$ grid.

Fig. 9 shows the error in the electrostatic energy due to the electron density (the Hartree energy) computed for increasing order of finite differences used in the defect correction for the 448 atom graphene sheet. The error is with respect to the electrostatic energy computed when the SPE is solved in reciprocal space (Eq. 4). As seen in the earlier results for the `erf_eps` and `pbez` tests (Figs. 2 and 3), the error decreases rapidly as the order of finite differences is increased. Since the reference Hartree energy is 21323.190421 E_{h}, the relative error in this energy for 12^{th}-order finite differences is $\sim 10^{-9}$.

The superficial similarity between Fig. 9 and Figs. 2 and 3 belies the significant differences in the calculations performed and the nature of the errors being computed. In the case of the `erf_eps` and `pbez` synthetic tests, the error was computed as the maximum difference (over all grid points) between the numerical solution of the GPE or P-BE from DL_MG and an analytic solution (Eqs. 53 and 53). In contrast, the error plotted in Fig. 9 represents the error in the electrostatic energy.

Table 1: Summary of results obtained for single-point DFT calculations on a T4 lysozyme-catechol complex, performed using ONETEP, where the SPE is solved in reciprocal space (RS) or using DL_MG (MG). For the ONETEP calculations performed with the RS and MG approaches, energies E (total and Hartree), timings t (total and for SPE solution) and SCF iterations are reported. "SCF iterations" refers to the number of outer loop iterations in which ONETEP's strictly localized orbitals are optimized (see Ref. 39). The percentage of the total time spent solving the SPE and the absolute and relative differences in the energies and execution times for the two SPE solution methods are also included. All MG calculations were performed using 12^{th}-order finite differences in the defect correction. Timing data was taken from the repetition with minimum total time, t_{total}, for three identical repetitions of the calculation.

	MG	RS	$\lvert E_{\text{RS}} - E_{\text{MG}} \rvert$	$\left\lvert \frac{E_{\text{RS}} - E_{\text{MG}}}{E_{\text{RS}}} \right\rvert$
E_{total} / E_{h}	−11632.5015	−11632.5026	1.04×10^{-3}	8.96×10^{-8}
E_{Hartree} / E_{h}	331669.7612	331669.7628	1.55×10^{-3}	4.66×10^{-9}
SCF iterations	13	13		
	MG	RS	$\lvert t_{\text{RS}} - t_{\text{MG}} \rvert$	$\left\lvert \frac{t_{\text{RS}} - t_{\text{MG}}}{t_{\text{RS}}} \right\rvert$
t_{total} / s	10213	8546	1668	1.95×10^{-1}
t_{SPE} / s	1642	34	1608	4.68×10^{1}
% t_{total} for SPE	16.08	0.40		

This is the result of a self-consistent DFT calculation in which the electrostatic potential is re-evaluated multiple times, forming part of the one-electron Hamiltonian (see Eq. 21). The small error incurred from using DL_MG to solve the SPE is thus a very strong validation of the accuracy of the electrostatic potential produced by DL_MG—any significant error in the potential would be compounded during the SCF procedure.

The behavior of DL_MG when solving the Poisson equation for a large biological system was examined by performing single point DFT energy calculations in ONETEP on a 2615 atom T4 lysozyme-catechol complex. The solvation of this complex was previously studied in Ref. 15 using the MPSM, a variant of the Fattebert-Gygi electrostatic solvation model described in section 2.2.

To evaluate the numerical accuracy of the results produced by ONETEP with DL_MG, the complex was first studied in vacuum with periodic BCs. This allowed the results obtained with DL_MG to be directly compared to the results obtained when solving the SPE using the standard reciprocal space approach (Eq. 4) employed in ONETEP. Table 1 shows the results of these calculations, which were run on the EPSRC MMM Hub "Thomas" machine, using a ONETEP binary (linked to DL_MG), compiled using the Intel Fortran compiler 17.0.1 and Intel MPI 2017.

The calculations were run on 120 cores (40 MPI processes with 3 OpenMP threads per process), which represents a typical resource allocation for a job of this size. A $129.5\,a_0 \times 129.5\,a_0 \times 129.5\,a_0$ simulation cell was used and the SPE was solved (for both methods) on a 512^3 grid, corresponding to a grid point spacing of $0.253\,a_0$.

The excellent numerical agreement in energies computed using DL_MG and the reciprocal space approach to solve the SPE seen in Fig. 9 is evident in Table 1. The total energy and Hartree component (Eqs. 16 and 17) computed using these two approaches agree to within $\sim 10^{-3}\,E_\mathrm{h}$. Considering the large magnitude of the energies, this represents very good agreement, corresponding to relative errors of $\sim 10^{-8}$ and $\sim 10^{-9}$ for the total and Hartree energies, respectively.

The execution times reported in Table 1 indicate that the defect-corrected multigrid approach is considerably more costly than the reciprocal space method. The time spent solving the SPE with DL_MG is nearly 50× the time spent solving this in reciprocal space. This is not surprising, given the well-known superior performance of FFT-based solutions to the SPE on uniform grids (see, for example, Ref.[79]).

While DL_MG is substantially outperformed by the reciprocal space method when solving the SPE, the strength of DL_MG is in its flexibility. Eq. 4 is only applicable to the SPE in periodic BCs, while DL_MG can be applied to solve more complicated variants of the Poisson equation (e.g. GPE, Eq. 7; and P-BE, Eq. 12) with fully open, fully periodic and mixed open/periodic BCs. As described in section 2.2, the solution of these variants of the Poisson equation enables electronic structure calculations to be performed in the presence of implicit solvent.

Table 2 summarizes the results of a free energy of solvation calculation performed on the same 2615 atom T4 lysozyme-catechol complex considered in Table 1. In these calculations, DL_MG was used to solve the SPE and GPE with fully open BCs, allowing the free energy of solvation to be computed using the MPSM.[15,16] As before, the calculations were run on the EPSRC MMM Hub "Thomas" machine, using the same ONETEP binary used in the vacuum PBC calculations and 120 cores (40 MPI processes with 3 OpenMP threads per process).

The physical parameters used in the solvent model were for solvation in H_2O (bulk permittivity,

Table 2: Summary of results obtained for free energy of solvation calculations on a T4 lysozyme-catechol complex, performed with ONETEP using DL_MG to solve the SPE (vacuum) and GPE (solvent). The results are for an "auto-solvation" calculation, where the vacuum and solvent energies required to evaluate the free energy of solvation, ΔG_{solv}, are computed in a single job. The total and electrostatic energies, E_{total} and E_{es}, in vacuum and solvent are reported. E_{es} is the energy due to the total charge density (electrons and ionic cores) of the complex interacting with the total electrostatic potential obtained by solving the SPE (vacuum) or GPE (solvent), subject to the approximation of smeared ionic core charges (described in Ref. 16). The number of SCF iterations for each calculation component is as defined for Table 1. The timings are for the full auto-solvation calculation: t_{total} is the total time, t_{PE} is time spent solving the SPE/GPE in DL_MG (with 12^{th}-order finite differences), and t_{BC} is the time spent computing coarse-grained boundary conditions in ONETEP (see Ref. 16). Timing data was taken from the repetition with minimum total time, t_{total}, for three identical repetitions of the calculation.

	Vacuum	Solvent
E_{total} / E_{h}	−11632.0051	−11635.8353
E_{es} / E_{h}	1306.7135	1303.8631
ΔG_{solv} / E_{h}	-3.8303	
SCF iterations	19	5
	Auto-solvation	
t_{total} / s	29259	
t_{SPE} / s	5037	
t_{BC} / s	7412	
% t_{total} for SPE/GPE	17.2	
% t_{total} for BCs	25.3	

$\varepsilon_\infty = 78.54$, and surface tension, $\gamma = 4.7624 \times 10^{-5}\, E_{\text{h}} a_0^{-2}$), and default values were used for the empirically determined model parameters. The SPE and GPE were solved on a 505^3 grid, which represented a slightly truncated version of the cubic simulation cell used in the PBC calculations ($128.2304\, a_0 \times 128.2304\, a_0 \times 128.2304\, a_0$)—this was necessary to satisfy DL_MG's grid size constraints for OBCs (Eq. 52).

The free energy of solvation computed for the T4 lysozyme-catechol complex in this work differs from the value presented in Ref. 15 by $\sim 10^{-2}\, E_{\text{h}}$. This is $< 1\%$ of the magnitude of the value and represents very good agreement considering that the calculation settings used in this work were not tuned for numerical agreement with Ref. 15.

The total execution time for the free energy of solvation calculation reported in Table 2 is $\sim 3\times$ the time taken for the vacuum PBC calculation on the same system (MG column in Table 1). Given that the calculation of the free energy of solvation involves separate calculations in vacuum and

solvent and the use of open BCs requires the costly computation of Dirichlet BCs, it is not surprising that the execution time is substantially greater. The time spent computing the BCs in ONETEP (using the coarse-graining technique described in Ref. 16) is actually greater than the time spent solving the SPE and GPE for this particular calculation.

The time spent solving the SPE and GPE in the solvation calculation is also $\sim 3\times$, the time spent solving the SPE in the vacuum PBC calculation, while the fraction of overall calculation time occupied by the solver is approximately the same at 16 to 17%.

It is tempting to compare the solver times in the vacuum PBC (Table 1) and solvation OBC (Table 2) calculations in light of number of SCF iterations in each calculation (26 for both parts of the solvation calculation and 13 for the vacuum PBC calculation). However, the discrepancies in the calculations prevent us from drawing meaningful insights from the apparent discrepancy between the $2\times$ increase in SCF iterations vs. $3\times$ increase in solver time. In particular, the different sizes of grids used in these calculations changes the number of multigrid levels available: the vacuum PBC calculations (512^3 grid) used 9 levels, while the solvation calculations (505^3 grid) used 4.

Overall, the execution times presented in Table 2 demonstrate that, using DL_MG to solve the SPE and GPE, large-scale electronic structure calculations in implicit solvent are accessible with modest computational resources and with execution times which are not substantially different to calculations in vacuum. Even when compared to the vacuum PBC calculation where the SPE is solved in reciprocal space, the total execution time for the solvation calculation is only $3.4\times$ larger.

5 Conclusions

We have described the implementation of DL_MG, a general-purpose Poisson solver library, and examined its numerical accuracy and computational performance when applied to chemically relevant model systems and in large scale electronic structure calculations.

In section 4.1.1, we demonstrated that DL_MG's defect-corrected multigrid approach could accurately solve the generalized and Poisson-Boltzmann variants of the Poisson equation for two

model systems involving implicit solvent. These results (Figs. 2 and 3) demonstrated the critical importance of the high-order defect correction (sections 2.4 and 3.1.2) in obtaining accurate solutions—with the second-order multigrid solver alone, the error in the solutions obtained was several orders of magnitude larger for both model systems.

The scaling of computational cost with respect to problem size was examined in section 4.1.2, where the solver library was seen to scale efficiently to problems with billions of unknowns for the two model systems (Figs. 4 and 5). Linear, or near-linear scaling was observed in both model systems for the range of grid sizes tested, which was selected to represent typical grid sizes used in electronic structure calculations ($\sim 100^3$ to $\sim 1000^3$). We also demonstrated the capability of DL_MG to scale efficiently to 100s of cores, typical of the parallel resources used in medium to large scale electronic structure calculations (Figs. 6 and 7).

We reported the results of electronic structure calculations in vacuum and solution performed with ONETEP, using DL_MG to solve the standard and generalized variants of the Poisson equation (section 4.2). Since the SPE (with fully periodic BCs, Eq. 1) is amenable to solution using FFT-based techniques already available in ONETEP (Eq. 4), we were able to compare numerical results obtained using multigrid and FFT-based solvers. The electrostatic energies obtained using potentials returned by DL_MG were in excellent agreement with energies yielded by a reciprocal space solution to the SPE (Fig. 9 and Table 1). The error in the energies calculated using DL_MG (with respect to the energy computed using the reciprocal space solution) improved with the order of finite differences used in the high-order defect correction, demonstrating similar behavior to the model systems (section 4.1.1).

The differences in the energies computed using DL_MG and the reciprocal space approach plotted in Fig. 9 emphasize the importance of the defect correction for obtaining chemically meaningful results. The 448 atom graphene sheet used in these calculations is typical of the types of surface that may be used in studying the interaction of large systems with a support (see for example Ref. 80 for a recent study of the interaction of Pt nanoparticles with a graphene monolayer using ONETEP). In such studies, small energy differences of $\sim 10^{-3}\,E_\mathrm{h}$ or less are chemically

relevant. Our results for this particular graphene sheet suggest that $\geq 8^{\text{th}}$-order finite differences are necessary to obtain this level of accuracy in electrostatic energies computed with DL_MG. Note that, for energy differences, error cancellation may allow this level of accuracy to be achieved with lower order finite differences, as described in Ref. 16.

Fully self-consistent DFT calculations were performed on a 2615 atom T4 lysozyme-catechol complex, representative of the kinds of systems studied in modern large-scale electronic structure calculations. These calculations were performed on 120 cores on a tier 2 supercomputer in order to produce timings representative of the typical usage of modern electronic structure packages, such as ONETEP. Using DL_MG to solve the GPE with open BCs, we measured the execution time required to compute the free energy of solvation for the entire complex, and found this to be only $\sim 3.4\times$ the time taken to compute a single-point energy for the system in vacuum using a reciprocal space SPE solver and periodic BCs. This is a small increase in total cost when it is considered that computing the free energy of solvation requires single-point calculations in both vacuum and solvent and that computation of open BCs involves significant additional computational work (see Table 2).

To assess the performance of DL_MG against an alternative solver, we examined the relative performance of DL_MG and the reciprocal space approach for solving the SPE in periodic BCs. For the specific case of the T4 lysozyme-catechol complex on 120 cores, we found that the reciprocal space approach far outperformed DL_MG, with DL_MG occupying $\sim 50\times$ more of the total execution time. Despite this large difference in time spent in the solver, the overall execution time for the calculation using DL_MG was only $\sim 20\%$ larger, indicative of the larger fixed costs of other parts of the calculation.

As discussed in section 4.2, the superior performance of the reciprocal space approach for solving the SPE in periodic BCs is well-known.[79] For this reason, we recommend that DL_MG is made available alongside established FFT-based Poisson solvers in electronic structure codes. Under the specific circumstances where the reciprocal space solution (Eq. 4) is applicable, users can benefit from the great efficiency of this method. Where non-periodic BCs or more complicated

variants of the Poisson equation are required, DL_MG may be used. This approach has been successfully adopted in ONETEP, where the Poisson solver is selected based on the nature of the calculation being performed.

DL_MG is a well-tested and stable library suitable for use in production calculations, as attested by the results presented in this work. Nevertheless, as always with scientific software, there is plenty of scope for improvement and extension.

In terms of code optimization, it is clear that future work in this area should focus on the second-order multigrid solver, rather than the defect correction, since the fraction of time spent evaluating high-order derivatives is negligible compared to the time spent in the multigrid solver (Figs. 4 and 5).

A key practical aspect of the code which would benefit from further development is the grid size constraint. In order for DL_MG to operate with a sufficient number of multigrid levels, the external program must provide data on grids that satisfy specific size constraints (Eq. 52). We are currently investigating methods by which this constraint may be eliminated, for example by having DL_MG interpolate input data onto an optimally sized grid internally.

Finally, the results we have presented in this work demonstrate that DL_MG is a flexible, scalable and accurate Poisson solver library. We therefore hope that interested readers will consider downloading the code and evaluating it for their own purposes—the library is released under a permissive open source license and is currently available to download.[52]

Acknowledgement

The authors gratefully acknowledge funding under the embedded CSE program (projects eCSE01-004 and eCSE07-006) of the ARCHER UK National Supercomputing Service (http://www.archer.ac.uk), which has supported the development and extension of the DL_MG multigrid solver library. We are also grateful for funding under the distributed CSE program of HECToR, the former UK National Supercomputing Service, which supported the initial development of DL_MG.

J.D. and C.K.S. acknowledge funding from the Engineering and Physical Sciences Research Council (EPSRC Grant No. EP/J015059/1).

We are grateful to the UK Materials and Molecular Modelling Hub (partially funded by EPSRC, EP/P020194/1) and the ARCHER UK National Supercomputing Service (http://www.archer.ac.uk) for the computational resources used in this work. We thank the UKCP consortium (EPSRC Grant No. EP/P022561/1) for providing additional resources on the ARCHER supercomputer.

A Stencils

When considering the discrete representation of differential operators, such as those featuring in the standard and generalized Poisson equations (Eqs. 1 and 7), it is often convenient to think in terms of stencils. This concept and associated notation is clearly defined in Ref. 10. We provide a brief summary here for the convenience of interested readers.

The stencil for an operator discretized on a regular grid describes the set of grid points in the locality of a point of interest which are involved in the application of the operator. It is common to refer to a stencil in terms of the number of points involved. For example, the forward difference approximation to the derivative in Eq. 32 corresponds to a two-point stencil on a 1-D grid, with the point of interest x and adjacent point $x + h$. For multidimensional grids, and higher-order finite difference approximations, larger numbers of grid points are involved.

The utility of the stencil concept is in the compact expression of the "shape" of a discretized operator on a grid. In particular, the geometric arrangement of the points involved in a discretized operator can easily be discerned using "stencil notation" As an example, consider the SPE discretized on a 2-D grid,

$$\hat{L}_h \phi_h(x, y) = -4\pi n_h(x, y), \tag{62}$$

with discretized Laplacian $\hat{L}_h$, potential $\phi_h(x, y)$, and density $n(x, y)$. A 5-point stencil (with

discretization error $O(h^2)$) has the following form

$$\begin{aligned}\hat{L}_h\phi_h(x,y) &= \frac{1}{h^2}\left[-4\phi_h(x,y)+\phi_h(x-h,y)+\phi_h(x+h,y)+\phi_h(x,y-h)+\phi_h(x,y+h)\right] \\ &= \frac{1}{h^2}\left[\begin{array}{ccc} & 1 & \\ 1 & -4 & 1 \\ & 1 & \end{array}\right]_h \phi_h(x,y).\end{aligned} \tag{63}$$

The second line of Eq. 63 uses the compact stencil notation described in Ref. 10, where the geometric relationship between the grid points involved in the stencil is clearly demonstrated.

A general expression of the action of an operator discretized on a 2-D grid on a function on that grid is, in stencil notation:

$$\left[\begin{array}{ccccc} \ddots & \vdots & \vdots & \vdots & \iddots \\ \cdots & s_{-1,1} & s_{0,1} & s_{1,1} & \cdots \\ \cdots & s_{-1,0} & s_{0,0} & s_{1,0} & \cdots \\ \cdots & s_{-1,-1} & s_{0,-1} & s_{1,-1} & \cdots \\ \iddots & \vdots & \vdots & \vdots & \ddots \end{array}\right]_h f_h(x,y) = \sum_{i,j} s_{i,j} f_h(x+ih, y+jh). \tag{64}$$

This is trivially extended to 3-D grids by combining layers of 2-D stencils. For example, a 3-D 7-point stencil (with discretization error $O(h^2)$) for the Laplacian can be written:

$$\frac{1}{h^2}\left[\left[\begin{array}{ccc} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{array}\right]\left[\begin{array}{ccc} 0 & 1 & 0 \\ 1 & -6 & 1 \\ 0 & 1 & 0 \end{array}\right]\left[\begin{array}{ccc} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{array}\right]\right]_h . \tag{65}$$

B Information for developers

The DL_MG library has been designed to be simple to interface with existing electronic structure packages.

The current version of the library (v2.0) is written in Fortran 95 and packaged with a GNU Makefile which automatically compiles the source code into a single static library. The library has no substantial external dependencies and can be compiled with modern Fortran compilers from Cray, Intel and GNU. Compilation with MPI and OpenMP is typically as simple as using the an MPI compiler wrapper (e.g. `mpif90`) and adding the vendor-specific flag to compile with OpenMP support (e.g. `-fopenmp` for gfortran).

The typical procedure for calling DL_MG from within an existing electronic structure code is very simple:

- Initialize the solver using `dl_mg_init`.
- Call `dl_mg_solver` with appropriate arguments.

The arguments that must be passed to the initialization and solver routines depend on the nature of the problem being solved (e.g. equation type), the type of parallelism employed (if any), and whether default parameters (e.g. convergence tolerances) are being overridden.

For a typical use case, where DL_MG is used to solve the GPE across several MPI processes and the default convergence tolerances are used, the calls to `dl_mg_init` and `dl_mg_solver` might take the following forms:

```
call  dl_mg_init(nx, ny, nz, dx, dy, dz, bc, gstart, gend, &
                 mg_comm, report_unit, report_file, ierror)

call dl_mg_solver(eps, eps_mid, alpha, rho, &
                  pot, fd_order, ierror)
```

In these subroutine calls the global grid has dimensions (`nx`, `ny`, `nz`) and `dx`, `dy`, `dz`, grid point spacing along x, y and z. The boundary conditions are determined by the integer constant `bc`—for Dirichlet BCs, `bc = DL_MG_BC_DIRICHLET`.

The MPI processes which will be used to solve the GPE and their Cartesian topology are described by the MPI communicator `mg_comm`, and for each MPI process the start and end points of the locally held grid within the global grid is given by the integer vectors `gstart` and `gend`.

DL_MG outputs detailed information to a log file while running, which is useful when debugging issues or tuning parameters. The log file has the name `report_file` and associated Fortran IO unit `report_unit`.

The type of equation to solve is inferred from the arguments provided when calling the overloaded `dl_mg_solver` routine. For the GPE (Eq. 7), we need to provide the dielectric permittivity `eps`, and charge density `rho` as input, and the corresponding electrostatic potential `pot` for output, all with the dimensions of the local grid held on this rank (i.e. `gend(:) - gstart(:) + 1`). In addition, we require the values of the dielectric permittivity at the points located halfway between the points of the global grid in each Cartesian direction, `eps_mid`.

The order of finite difference stencil (Eq. 48) used in the high-order defect correction is determined by `fd_order` (4, 6, 8, 10 or 12) and `alpha` is a multiplicative constant defined by the unit system (in the atomic units used throughout this paper, `alpha` is -4π). Finally, DL_MG may return integer-valued error codes through `ierror`.

This interface is designed to be simple and clean, but also offers a large amount of configuration options behind optional arguments. For example, it is possible to finely tune the absolute and relative convergence parameters for the multigrid V-cycle, inexact-Newton method and high-order defect correction (Eqs. 40, 42 and 43) via optional arguments to `dl_mg_solver`. For further details, see the developer documentation provided with the source code.[52]

C `erf_eps` test

The `erf_eps` synthetic test (described in section 4.1.1) is a useful analytic model which imitates the situation where a small molecule is solvated in an implicit solvent which is represented by a smoothly varying dielectric function (e.g. Eq. 24). The test implemented in DL_MG is based on the model described by Fisicaro *et al.* in Ref. 25, for which we have reproduced the analytic forms of the electrostatic potential (Eq. 53) and dielectric permittivity (Eq. 54). We have also provided the corresponding form of the charge density (Eq. 55), in order that developers of other Poisson

solvers may make use of this useful model.

For interested readers, we include here some of the intermediate steps in the derivation of Eq. 55 from Eqs. 53 and 54.

We start by rearranging the GPE to obtain an expression in terms of the charge density and expanding the divergence in terms of the product rule, i.e.

$$n(\mathbf{r}) = -\frac{1}{4\pi}\left[\varepsilon(\mathbf{r})\nabla^2\phi(\mathbf{r}) + (\nabla\varepsilon(\mathbf{r}))\cdot(\nabla\phi(\mathbf{r}))\right]. \tag{66}$$

The derivatives $\nabla\varepsilon(\mathbf{r})$, $\nabla\phi(\mathbf{r})$ and $\nabla^2\phi(\mathbf{r})$ (using the definitions of $\varepsilon(\mathbf{r}$ and $\phi(\mathbf{r})$ in Eqs. 53 and 54) are

$$\nabla\varepsilon(\mathbf{r}) = \frac{(\varepsilon_\infty - 1)}{\Delta|\mathbf{r}-\mathbf{R}|}\frac{(\mathbf{r}-\mathbf{R})}{\pi^{1/2}}\exp\left(-\left(\frac{|\mathbf{r}-\mathbf{R}|^2 - d_0}{\Delta}\right)^2\right), \tag{67}$$

$$\nabla\phi(\mathbf{r}) = -\frac{\phi(\mathbf{r})}{\sigma^2}(\mathbf{r}-\mathbf{R}), \tag{68}$$

$$\nabla^2\phi(\mathbf{r}) = \frac{\phi(\mathbf{r})}{\sigma^2}\left(\frac{|\mathbf{r}-\mathbf{R}|^2}{\sigma^2} - 3\right). \tag{69}$$

Substituting Eqs. 67 to 69 into Eq. 66 leads directly to the form of the charge density in the `erf_eps` test (Eq. 55)

References

(1) Skylaris, C.-K.; Haynes, P. D.; Mostofi, A. A.; Payne, M. C. Introducing ONETEP: Linear-scaling density functional simulations on parallel computers. *J. Chem. Phys.* **2005**, *122*, 084119.

(2) Mohr, S.; Ratcliff, L. E.; Genovese, L.; Caliste, D.; Boulanger, P.; Goedecker, S.; Deutsch, T. Accurate and efficient linear scaling DFT calculations with universal applicability. *Phys. Chem. Chem. Phys.* **2015**, *17*, 31360–31370.

(3) Gillan, M. J.; Bowler, D. R.; Torralba, A. S.; Miyazaki, T. Order-N first-principles calculations with the conquest code. *Comput. Phys. Commun.* **2007**, *177*, 14–18.

(4) Duy, T. V. T.; Ozaki, T. A three-dimensional domain decomposition method for large-scale DFT electronic structure calculations. *Comput. Phys. Commun.* **2014**, *185*, 777–789.

(5) VandeVondele, J.; Krack, M.; Mohamed, F.; Parrinello, M.; Chassaing, T.; Hutter, J. Quickstep: Fast and accurate density functional calculations using a mixed Gaussian and plane waves approach. *Comput. Phys. Commun.* **2005**, *167*, 103–128.

(6) Soler, J. M.; Artacho, E.; Gale, J. D.; García, A.; Junquera, J.; Ordejón, P.; Daniel Sánchez-Portal, The SIESTA method for ab initio order- N materials simulation. *J. Phys.: Condens. Matter* **2002**, *14*, 2745.

(7) Lever, G.; Cole, D. J.; Hine, N. D. M.; Haynes, P. D.; Payne, M. C. Electrostatic considerations affecting the calculated HOMO–LUMO gap in protein molecules. *J. Phys.: Condens. Matter* **2013**, *25*, 152101.

(8) Brandt, A. Multi-level adaptive solutions to boundary-value problems. *Math. Comp.* **1977**, *31*, 333–390.

(9) Briggs, W.; Henson, V.; McCormick, S. *A Multigrid Tutorial, Second Edition*; Other Titles in Applied Mathematics; Society for Industrial and Applied Mathematics, 2000; DOI: 10.1137/1.9780898719505 DOI: 10.1137/1.9780898719505.

(10) Trottenberg, U.; Oosterlee, C. W.; Schüller, A. *Multigrid*; Academic Press, 2001.

(11) Merrick, M. P.; Iyer, K. A.; Beck, T. L. Multigrid Method for Electrostatic Computations in Numerical Density Functional Theory. *J. Phys. Chem.* **1995**, *99*, 12478–12482.

(12) Fattebert, J.-L.; Gygi, F. Density functional theory for efficient ab initio molecular dynamics simulations in solution. *J. Comput. Chem.* **2002**, *23*, 662–666.

(13) Dabo, I.; Kozinsky, B.; Singh-Miller, N. E.; Marzari, N. Electrostatics in periodic boundary conditions and real-space corrections. *Phys. Rev. B* **2008**, *77*, 115139.

(14) Sánchez, V. M.; Sued, M.; Scherlis, D. A. First-principles molecular dynamics simulations at solid-liquid interfaces with a continuum solvent. *J. Chem. Phys.* **2009**, *131*, 174108.

(15) Dziedzic, J.; Helal, H. H.; Skylaris, C.-K.; Mostofi, A. A.; Payne, M. C. Minimal parameter implicit solvent model for ab initio electronic-structure calculations. *EPL* **2011**, *95*, 43001.

(16) Dziedzic, J.; Fox, S. J.; Fox, T.; Tautermann, C. S.; Skylaris, C.-K. Large-scale DFT calculations in implicit solvent—A case study on the T4 lysozyme L99A/M102Q protein. *Int. J. Quantum Chem.* **2013**, *113*, 771–785.

(17) Garcia-Ratés, M.; López, N. Multigrid-Based Methodology for Implicit Solvation Models in Periodic DFT. *J. Chem. Theory Comput.* **2016**, *12*, 1331–1341.

(18) Briggs, E. L.; Sullivan, D. J.; Bernholc, J. Large-scale electronic-structure calculations with multigrid acceleration. *Phys. Rev. B* **1995**, *52*, R5471–R5474.

(19) Briggs, E. L.; Sullivan, D. J.; Bernholc, J. Real-space multigrid-based approach to large-scale electronic structure calculations. *Phys. Rev. B* **1996**, *54*, 14362–14375.

(20) Beck, T. L.; Iyer, K. A.; Merrick, M. P. Multigrid methods in density functional theory. *Int. J. Quantum Chem.* **1997**, *61*, 341–348.

(21) Bernholc, J.; Hodak, M.; Lu, W. Recent developments and applications of the real-space multigrid method. *J. Phys.: Condens. Matter* **2008**, *20*, 294205.

(22) Fattebert, J.-L.; Gygi, F. First-principles molecular dynamics simulations in a continuum solvent. *Int. J. Quantum Chem.* **2003**, *93*, 139–147.

(23) Scherlis, D. A.; Fattebert, J.-L.; Gygi, F.; Cococcioni, M.; Marzari, N. A unified electrostatic and cavitation model for first-principles molecular dynamics in solution. *J. Chem. Phys.* **2006**, *124*, 074103.

(24) Andreussi, O.; Dabo, I.; Marzari, N. Revised self-consistent continuum solvation in electronic-structure calculations. *J. Chem. Phys.* **2012**, *136*, 064102.

(25) Fisicaro, G.; Genovese, L.; Andreussi, O.; Marzari, N.; Goedecker, S. A generalized Poisson and Poisson-Boltzmann solver for electrostatic environments. *J. Chem. Phys.* **2016**, *144*, 014103.

(26) Arfken, G. B.; Weber, H. J.; Harris, F. E. *Mathematical Methods for Physicists: A Comprehensive Guide*; Academic Press, 2011.

(27) We have adopted the nomenclature used in Ref. 25 for the variants of the Poisson equation. The standard (Eq. 1) and generalized (Eq. 7) forms of the equation are also sometimes referred to as the "homogeneous" and "nonhomogeneous" Poisson equations, respectively, where the (non-)homogeneity of the equation refers to the structure of the dielectric permittivity, $\varepsilon(\mathbf{r})$. We eschew this terminology because it is potentially ambiguous—the descriptor "homogeneous" has a specific and different meaning when describing differential equations. Note that the generalized Poisson equation is also sometimes referred to as the "variable-coefficient Poisson equation" (see, for example, Ref. 81).

(28) Fogolari, F.; Brigo, A.; Molinari, H. The Poisson–Boltzmann equation for biomolecular electrostatics: a tool for structural biology. *J. Mol. Recognit.* **2002**, *15*, 377–392.

(29) Lu, B.; Zhou, Y.; Holst, M.; McCammon, J. Recent progress in numerical methods for the Poisson-Boltzmann equation in biophysical applications. *Commun. Comput. Phys.* **2008**, *3*, 973–1009.

(30) Grochowski, P.; Trylska, J. Continuum molecular electrostatics, salt effects, and counterion binding—A review of the Poisson–Boltzmann theory and its modifications. *Biopolymers* **2008**, *89*, 93–113.

(31) Ringe, S.; Oberhofer, H.; Hille, C.; Matera, S.; Reuter, K. Function-Space-Based Solution

Scheme for the Size-Modified Poisson–Boltzmann Equation in Full-Potential DFT. *J. Chem. Theory Comput.* **2016**, *12*, 4052–4066.

(32) Ringe, S.; Oberhofer, H.; Reuter, K. Transferable ionic parameters for first-principles Poisson-Boltzmann solvation calculations: Neutral solutes in aqueous monovalent salt solutions. *J. Chem. Phys.* **2017**, *146*, 134103.

(33) Press, W. H.; Teukolsky, S. A.; Vetterling, W. T.; Flannery, B. P. *Numerical Recipes in C: The Art of Scientific Computing*, 2nd ed.; Cambridge University Press: Cambridge, UK, 1992.

(34) Payne, M. C.; Teter, M. P.; Allan, D. C.; Arias, T. A.; Joannopoulos, J. D. Iterative minimization techniques for \textit{ab initio} total-energy calculations: molecular dynamics and conjugate gradients. *Rev. Mod. Phys.* **1992**, *64*, 1045–1097.

(35) Hine, N. D. M.; Dziedzic, J.; Haynes, P. D.; Skylaris, C.-K. Electrostatic interactions in finite systems treated with periodic boundary conditions: Application to linear-scaling density functional theory. *J. Chem. Phys.* **2011**, *135*, 204103.

(36) Tomasi, J.; Mennucci, B.; Cammi, R. Quantum Mechanical Continuum Solvation Models. *Chem. Rev.* **2005**, *105*, 2999–3094.

(37) York, D. M.; Karplus, M. A Smooth Solvation Potential Based on the Conductor-Like Screening Model. *J. Phys. Chem. A* **1999**, *103*, 11060–11079.

(38) Fisicaro, G.; Genovese, L.; Andreussi, O.; Mandal, S.; Nair, N. N.; Marzari, N.; Goedecker, S. Soft-sphere continuum solvation in electronic-structure calculations. *J. Chem. Theory Comput.* **2017**,

(39) Skylaris, C.-K.; Haynes, P. D.; Mostofi, A. A.; Payne, M. C. Implementation of linear-scaling plane wave density functional theory on parallel computers. *Phys. Status Solidi B* **2006**, *243*, 973–988.

(40) Hine, N. D. M.; Haynes, P. D.; Mostofi, A. A.; Skylaris, C. K.; Payne, M. C. Linear-scaling density-functional theory with tens of thousands of atoms: Expanding the scope and scale of calculations with ONETEP. *Comput. Phys. Commun.* **2009**, *180*, 1041–1053.

(41) Wilkinson, K. A.; Hine, N. D. M.; Skylaris, C.-K. Hybrid MPI-OpenMP Parallelism in the ONETEP Linear-Scaling Electronic Structure Code: Application to the Delamination of Cellulose Nanofibrils. *J. Chem. Theory Comput.* **2014**, *10*, 4782–4794.

(42) Collatz, L. *The Numerical Treatment of Differential Equations*; Springer Berlin Heidelberg: Berlin, Heidelberg, 1960; DOI: 10.1007/978-3-642-88434-4.

(43) Schaffer, S. Higher order multigrid methods. *Math. Comp.* **1984**, *43*, 89–115, S1.

(44) Wesseling, P.; Oosterlee, C. W. Geometric multigrid with applications to computational fluid dynamics. *Journal of Computational and Applied Mathematics* **2001**, *128*, 311–334.

(45) Stüben, K. A review of algebraic multigrid. *Journal of Computational and Applied Mathematics* **2001**, *128*, 281–309.

(46) Chow, E.; Falgout, R. D.; Hu, J. J.; Tuminaro, R. S.; Yang, U. M. In *Parallel Processing for Scientific Computing*; Heroux, M. A., Raghavan, P., Simon, H. D., Eds.; SIAM series on Software, Environments and Tools; SIAM, 2006.

(47) Zhang, J. Acceleration of five-point red-black Gauss-Seidel in multigrid for Poisson equation. *Appl. Math. Comput.* **1996**, *80*, 73–93.

(48) Terboven, C.; an Mey, D.; Schmidl, D.; Jin, H.; Reichstein, T. Data and Thread Affinity in Openmp Programs. Proceedings of the 2008 Workshop on Memory Access on Future Processors: A Solved Problem? New York, NY, USA, 2008; pp 377–384.

(49) Gropp, W.; Torsten, H.; Thakur, R.; Lusk, E. *Using Advanced MPI: Modern Features of the Message-Passing Interface*; MIT Press, 2014.

(50) Chapman, B.; Jost, G.; van der Pas, R. *Using OpenMP: Portable Shared Memory Parallel Programming*; 2014.

(51) Holst, M. J.; Saied, F. Numerical solution of the nonlinear Poisson–Boltzmann equation: Developing more robust and efficient methods. *J. Comput. Chem.* **1995**, *16*, 337–364.

(52) Anton, L.; Womack, J. C.; Dziedzic, J. DL_MG multigrid solver. 2017; `https://ccpforge.cse.rl.ac.uk/gf/project/dl-mg/`.

(53) Clark, S. J.; Segall, M. D.; Pickard, C. J.; Hasnip, P. J.; Probert, M. I. J.; Refson, K.; Payne, M. C. First principles methods using CASTEP. *Z. Kristallogr.* **2005**, *220*, 567–570.

(54) Wolfram Research Inc, *Mathematica, Version 11.2*; 2017.

(55) Anton, L.; Dziedzic, J.; Skylaris, C.-K.; Probert, M. *Multigrid solver module for ONETEP, CASTEP and other codes*; 2013; http://www.hector.ac.uk/cse/distributedcse/reports/onetep/.

(56) Parrish, R. M.; Burns, L. A.; Smith, D. G. A.; Simmonett, A. C.; DePrince, A. E.; Hohenstein, E. G.; Bozkaya, U.; Sokolov, A. Y.; Di Remigio, R.; Richard, R. M.; Gonthier, J. F.; James, A. M.; McAlexander, H. R.; Kumar, A.; Saitow, M.; Wang, X.; Pritchard, B. P.; Verma, P.; Schaefer, H. F.; Patkowski, K.; King, R. A.; Valeev, E. F.; Evangelista, F. A.; Turney, J. M.; Crawford, T. D.; Sherrill, C. D. Psi4 1.1: An Open-Source Electronic Structure Program Emphasizing Automation, Advanced Libraries, and Interoperability. *J. Chem. Theory Comput.* **2017**, *13*, 3185–3197.

(57) Womack, J. C.; Anton, L.; Dziedzic, J.; Hasnip, P. J.; Probert, M. I. J.; Skylaris, C.-K. *Implementation and optimisation of advanced solvent modelling functionality in CASTEP and ONETEP*; 2017; p 30, http://www.archer.ac.uk/community/eCSE/eCSE07-06/eCSE07-06.php.

(58) Howard, J. C.; Womack, J. C.; Dziedzic, J.; Skylaris, C.-K.; Pritchard, B. P.; Crawford, T. D. Electronically Excited States in Solution via a Smooth Dielectric Model Combined with

Equation-of-Motion Coupled Cluster Theory. *J. Chem. Theory Comput.* **2017**, *13*, 5572–5581.

(59) Skylaris, C.-K.; Mostofi, A. A.; Haynes, P. D.; Diéguez, O.; Payne, M. C. Nonorthogonal generalized Wannier function pseudopotential plane-wave method. *Phys. Rev. B* **2002**, *66*, 035119.

(60) Mostofi, A. A.; Haynes, P. D.; Skylaris, C.-K.; Payne, M. C. Preconditioned iterative minimization for linear-scaling electronic structure calculations. *J. Chem. Phys.* **2003**, *119*, 8842–8848.

(61) The gfortran team, *Using GNU Fortran*; 2017; https://gcc.gnu.org/onlinedocs/.

(62) Intel Corporation, *Intel® MPI Library Developer Reference for Linux* OS*; 2017; https://software.intel.com/en-us/intel-fortran-compiler-17.0-user-and-reference-guide.

(63) For completeness, we note that the definition of the Gaussian potential used in DL_MG's test suite differs from Eq. 53 by a factor of $\pi^{3/2}$, i.e. $\phi_{\mathrm{DL\_MG}}(\mathbf{r}) = \pi^{3/2}\phi(\mathbf{r})$. The values of potential (and the error in this) used in plots and quoted in the text for `erf_eps` correspond to the definition used in DL_MG.

(64) Butt, H.-J.; Graf, K.; Kappl, M. *Physics and Chemistry of Interfaces*, 3rd ed.; John Wiley & Sons, 2013.

(65) To be consistent with the other equations in this paper, Eqs. 56 to 59 are expressed in atomic units. When expressed in SI units (as in Ref. 64) these equations necessarily feature additional constants.

(66) The grid dimensions differ slightly from those used in the `erf_eps` test, since in the `pbez` test the boundary conditions are periodic in x and y, while the `erf_eps` test uses fully open BCs in all directions. DL_MG requires an even number of grid points along periodic directions and odd numbers of grid points along open directions (see Eq. 52).

(67) The EPSRC MMM Hub "Thomas" supercomputer is a UK national tier 2 supercomputer facility for materials and molecular modelling (MMM). At the time of writing, the supercomputer consists of 720 compute nodes, connected with Intel Omni-Path interconnect. Each node consists of 2×12 core Intel Broadwell processors sharing 128 GiB of RAM. For more details, see https://mmmhub.ac.uk/.

(68) Although the grid dimensions used for `pbez` and `erf_eps` differ by a single grid point in the x and y directions (section 4.1.1), the resulting difference in overall number of grid points is negligible, allowing direct comparison of the timings for the grid sizes used in each test case.

(69) Amdahl, G. M. Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities. Proceedings of the April 18-20, 1967, Spring Joint Computer Conference. New York, NY, USA, 1967; pp 483–485.

(70) Hill, M. D.; Marty, M. R. Amdahl's Law in the Multicore Era. *Computer* **2008**, *41*, 33–38.

(71) Perdew, J. P.; Burke, K.; Ernzerhof, M. Generalized Gradient Approximation Made Simple. *Phys. Rev. Lett.* **1996**, *77*, 3865–3868.

(72) Perdew, J. P.; Burke, K.; Ernzerhof, M. Generalized Gradient Approximation Made Simple [Phys. Rev. Lett. 77, 3865 (1996)]. *Phys. Rev. Lett.* **1997**, *78*, 1396–1396.

(73) Bennett, J. W. Discovery and Design of Functional Materials: Integration of Database Searching and First Principles Calculations. *Physics Procedia* **2012**, *34*, 14–23.

(74) A sulfur pseudopotential from a suite of pseudopotentials generated by K. Refson to supplement the Rappe-Bennett library was also used in calculations on the T4 lysozyme complex reported in section 4.2. ONETEP- and CASTEP-compatible versions of the Rappe-Bennett library and K. Refson's supplementary set of pseudopotentials are available to download from the CASTEP project page on CCPForge.[82]

(75) ARCHER is the UK's national supercomputing service, based on a Cray XC30 supercomputer. At the time of writing, ARCHER consists of 4920 nodes connected with an Aries interconnect. Each node contains 2×12 core Intel Ivy Bridge processors, with standard nodes sharing 64 GiB between the two processors. For further information, see https://www.archer.ac.uk/.

(76) FFTW version 3.3.4.10. `http://www.fftw.org/`.

(77) Frigo, M.; Johnson, S. G. The Design and Implementation of FFTW3. *Proc. IEEE* **2005**, *93*, 216–231.

(78) Cray Inc., *XC™ Series Programming Environment User Guide*, s-2529-17.05 ed.; 2017; https://pubs.cray.com/content/S-2529/17.05/xctm-series-programming-environment-user-guide-1705-s-2529.

(79) Gholami, A.; Malhotra, D.; Sundar, H.; Biros, G. FFT, FMM, or Multigrid? A comparative Study of State-Of-the-Art Poisson Solvers for Uniform and Nonuniform Grids in the Unit Cube. *SIAM J. Sci. Comput.* **2016**, *38*, C280–C306.

(80) Verga, L. G.; Aarons, J.; Sarwar, M.; Thompsett, D.; Russell, A. E.; Skylaris, C.-K. Effect of graphene support on large Pt nanoparticles. *Phys. Chem. Chem. Phys.* **2016**, *18*, 32713–32722.

(81) Nagel, J. R. Numerical Solutions to Poisson Equations Using the Finite-Difference Method [Education Column]. *IEEE Antenn. Propag. M.* **2014**, *56*, 209–224.

(82) CASTEP project page on CCPForge. `https://ccpforge.cse.rl.ac.uk/gf/project/castep/`.