Validation of the PyGBe code for Poisson-Boltzmann equation with boundary element methods[1]

Christopher D. Cooper, Lorena A. Barba

30 January 2013

The PyGBe code solves the linearized Poisson-Boltzmann equation using a boundary-integral formulation. We use a boundary element method with a collocation approach, and solve it via a Krylov-subspace method. To do this efficiently, the matrix-vector multiplications in the Krylov iterations are accelerated with a treecode, achieving $\mathcal{O}(N \log N)$ complexity. The code presents a Python environment for the user, while being efficient and fast. The core computational kernels are implemented in Cuda and interface with the user-visible code with PyCuda, for maximum ease-of-use combined with high performance on GPU hardware. This document provides background on the model and formulation of the numerical method, evidence of a validation exercise with well-known benchmarks—a spherical shell with a centered charge and one with an off-center charge— and a demonstration with a realistic biological geometry (lysozyme molecule).

Continuum model for biomolecular electrostatics

Background

The application domain of the PyGBe code (pronounced 'pig-bee') is electrostatics in biology. At a microscopic level, electrostatic effects control a majority of biological processes,[2] for example, the affinity of two biological compounds to bind or the catalytic power of enzymes. Calculating electrostatic energies is an effective way to study protein systems, often being capable of providing quantitative predictions.

[2] A. Warshel, P. K. Sharma, M. Kato, and W. W. Parson. Modeling electrostatic effects in proteins. *Biochim. Biophys. Acta*, 1764:1647–1676, 2006

In biology, proteins are never isolated; they are dissolved in water with salts and other proteins. Hence, the interactions between charges in a biological system are *screened* by the presence of water with dissolved ions. In this case, the classical treatment of electrostatics effects uses the Debye-Hückel approximation, where charges obey a Boltzmann distribution. Combined with the Poisson equation for the potential field, we obtain the Poisson-Boltzmann equation, which is the starting point of our model.

The model for biomolecular electrostatics based on the Poisson-Boltzmann equation is a semi-macroscopic approach: the region occupied by water (with dissolved ions) is represented by a continuum dielectric, and the region occupied by the protein is represented by a low-dielectric material with embedded point charges at the atom locations. It is referred to as an *implicit solvent* model.

The most popular approach to compute electrostatic interactions is fully microscopic, i.e., accounts for every single atom in the system (including the solvent) and the all-pair interactions via force fields. This is the explicit-solvent approach of molecular dynamics simulations. It is the most rigorous approach, but can lead to extremely costly computations, since the number of solvent molecules around the protein is very large.

Mathematical model

In the continuum model, the *solvent-excluded* surface area defines an interface between the region inside the protein and the solvent area.

The Poisson equation with point charges applies in the region inside the protein, with a low dielectric. In the solvent area, the Poisson-Boltzmann equation applies (used in linearized form), with a high dielectric corresponding to water. The electric potential can be described by the following system of partial differential equations, with interface conditions on the potential and electric displacement,

$$\begin{aligned} \nabla^2\phi_1(\mathbf{r}) &= -\sum_i \frac{q_i}{\epsilon_1}\delta(\mathbf{r},\mathbf{r}_i) \quad \text{in solute } (\Omega_1) \\ \nabla^2\phi_2(\mathbf{r}) &= \kappa^2\phi_2(\mathbf{r}) \quad \text{in solvent } (\Omega_2) \\ \phi_1 &= \phi_2 \qquad \text{on interface } \Gamma \\ \epsilon_1\frac{\partial\phi_1}{\partial\mathbf{n}} &= \epsilon_2\frac{\partial\phi_2}{\partial\mathbf{n}}, \end{aligned} \tag{1}$$

where κ is the reciprocal of the Debye length, indicating the strength of the screening effect.

The solvent-excluded area is obtained by rolling a spherical probe of the size of a water molecule all around the protein, defining the closest that a solvent molecule can get to it.

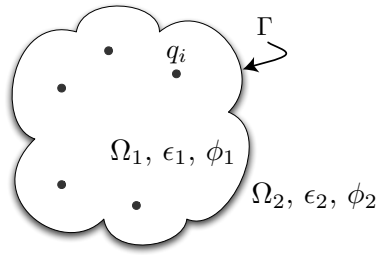


Figure 1: Sketch of the semi-macroscopic electrostatic model of a biomolecule in solution.

ϕ_1 and ϕ_2 are the potentials and ϵ_1 and ϵ_2 the dielectric constants for the regions inside the protein and in the solvent, respectively; q_i represents the charge of atom i of the protein, located at x_i. Commonly used values of ϵ_1 range between 2 and 4, whereas the solvent has $\epsilon_2 \approx 80$.

One of the most important quantities obtained using this model is the solvation free energy, ΔG_{solv}. It corresponds to the difference in free energy between two states: the protein in vacuum, where there is only a Coulombic field due to the point charges; and in the solvent, where the polarization due to the presence of the protein generates a different total potential field. The solvation free energy is generated by the difference in potential between these two states, called reaction potential and given by

$$\phi_{\text{reaction}} = \phi_1 - \phi_{\text{Coulomb}} \tag{2}$$

The energy due to ϕ_{reaction} yields ΔG_{solv} as follows:

$$\Delta G_{\text{solv}} = \frac{1}{2}\sum_{k=1}^{N_c} q_k\phi_{\text{reaction}}(\mathbf{r}_k), \tag{3}$$

ϕ_{reaction} is the difference in potential between the vacuum state and the solvated state (the potential due to polarization of the solvent), ϕ_1 is the total potential in Ω_1 and ϕ_{Coulomb} the potential generated by the point charges in the locations of the atoms; N_c is the number of point charges or atoms.

Equation (3) is the result of the convolution of ϕ_{reaction} and the charge field, which can be analytically integrated as the charge field is essentially the sum of delta functions.

Boundary integral formulation

We use a boundary integral formulation of the system (1), presented by Yoon and Lenhoff.[3] By taking the convolution of the first two equations in (1) with the free-space Green's function of the Laplace and linearized Poisson-Boltzmann equations, respectively, then evaluating the resulting equations on the interface and applying the interface conditions, the following set of boundary integral equations is obtained, valid for the potential ϕ_1 inside the protein:

[3] B J. Yoon and A. M. Lenhoff. A boundary element method for molecular electrostatics with electrolyte effects. *J. Comput. Chem.*, 11(9):1080–1086, 1990

$$2\pi\phi_1(\mathbf{r}) - \oint_\Gamma \frac{\partial\phi_1}{\partial\mathbf{n}}(\mathbf{r}')\frac{1}{|\mathbf{r}-\mathbf{r}'|}\mathrm{d}\Gamma' + \oint_\Gamma \frac{\partial}{\partial\mathbf{n}}\left[\frac{1}{|\mathbf{r}-\mathbf{r}'|}\right]\phi_1(\mathbf{r}')\mathrm{d}\Gamma' = \sum_{i=0}^{N_c}\frac{q_i}{\epsilon_1}\frac{1}{|\mathbf{r}-\mathbf{r}_i|}, \tag{4}$$
$$2\pi\phi_1(\mathbf{r}) + \frac{\epsilon_2}{\epsilon_1}\oint_\Gamma \frac{\partial\phi_1}{\partial\mathbf{n}}(\mathbf{r}')\frac{e^{-\kappa|\mathbf{r}-\mathbf{r}'|}}{|\mathbf{r}-\mathbf{r}'|}\mathrm{d}\Gamma' - \oint_\Gamma \frac{\partial}{\partial\mathbf{n}}\left[\frac{e^{-\kappa|\mathbf{r}-\mathbf{r}'|}}{|\mathbf{r}-\mathbf{r}'|}\right]\phi_1(\mathbf{r}')\mathrm{d}\Gamma' = 0,$$

$G_1(\mathbf{r},\mathbf{r}') = \frac{1}{|\mathbf{r}-\mathbf{r}'|}$ and $G_2(\mathbf{r},\mathbf{r}') = \frac{e^{-\kappa|\mathbf{r}-\mathbf{r}'|}}{|\mathbf{r}-\mathbf{r}'|}$ are the free-space Green's functions for the Laplace and linearized Poisson-Boltzmann operators, respectively. Vector $\mathbf{n} = \mathbf{n}(\mathbf{r}')$ varies with the integration variable and points outwards of the surface of the molecule.

Once the potential on the interface is known, the reaction potential at the locations of the point charges, needed in (3), is calculated by:

$$\phi_{\text{reaction}}(\mathbf{r}) = \oint_\Gamma \frac{\partial\phi_1(\mathbf{r}')}{\partial\mathbf{n}(\mathbf{r}')}G_1(\mathbf{r},\mathbf{r}')\mathrm{d}\Gamma' - \oint_\Gamma \frac{\partial G_1(\mathbf{r},\mathbf{r}')}{\partial\mathbf{n}(\mathbf{r}')}\phi_1(\mathbf{r}')\mathrm{d}\Gamma'. \tag{5}$$

Equation (5) is obtained in a similar fashion to Equation (4). In this case, the Coulombic contribution is subtracted as shown in Equation (2), and the equation is not evaluated at the interface, but at the locations of the point charges.

Multi-surface formulation

In cases where there are more than two dielectric regions, one-surface models are not capable of performing the simulation. This situation arises, for example, if the simulation involves two or more interacting proteins, the protein region has solvent-filled cavities, or when we want to consider steric effects using Stern layers. All of these situations require more than one surface to be modeled, and those surfaces interact with each other depending on their relative location.

The Stern layer is a 1–2 Å-thick layer around the surface of the molecule where there are no ions or atoms. This region is governed by the Poisson equation with no source term (Laplace equation). It accounts for the fact that the concentration of ions near charged surfaces is overestimated by the Poisson-Boltzmann equation due to steric effects.

We can extend this formulation to account for more than one surface using the guidelines detailed in the work by Altman and co-workers.[4]

[4] M. D. Altman, J. P. Bardhan, J. K. White, and B. Tidor. Accurate solution of multi-region continuum electrostatic problems using the linearized Poisson–Boltzmann equation and curved boundary elements. *J. Comput. Chem.*, 30:132–153, 2009

Numerical method

*The boundary element method (*BEM*)*

To solve the system of boundary integral equations in (4), we use a boundary element method (BEM). We discretize the protein-solvent

interface in panels or boundary elements, assume a distribution of ϕ and $\frac{\partial \phi}{\partial \mathbf{n}}$ over each panel, and then use either collocation or a Galerkin approach to generate a linear system. Currently, we use flat triangular panels with a constant distribution of the variables, and use collocation for the linear system. For N_p panels, the discretized form of Equation (4) is

$$2\pi\phi_1(\mathbf{r}_i) - \sum_{j=1}^{N_p} \frac{\partial \phi_1}{\partial \mathbf{n}}(\mathbf{r}_j) \int_{\Gamma_j} \frac{1}{|\mathbf{r}_i - \mathbf{r}'|} d\Gamma' + \sum_{j=1}^{N_p} \phi_1(\mathbf{r}_j) \int_{\Gamma_j} \frac{\partial}{\partial \mathbf{n}} \left[\frac{1}{|\mathbf{r}_i - \mathbf{r}'|} \right] d\Gamma' = \sum_{k=0}^{N_c} \frac{q_k}{\epsilon_1} \frac{1}{|\mathbf{r}_i - \mathbf{r}_k|} \tag{6}$$
$$2\pi\phi_1(\mathbf{r}_i) + \sum_{j=1}^{N_p} \frac{\partial \phi_1}{\partial \mathbf{n}}(\mathbf{r}_j) \frac{\epsilon_1}{\epsilon_2} \int_{\Gamma_j} \frac{e^{-\kappa|\mathbf{r}_i - \mathbf{r}'|}}{|\mathbf{r}_i - \mathbf{r}'|} d\Gamma' - \sum_{j=1}^{N_p} \phi_1(\mathbf{r}_j) \int_{\Gamma_j} \frac{\partial}{\partial \mathbf{n}} \left[\frac{e^{-\kappa|\mathbf{r}_i - \mathbf{r}'|}}{|\mathbf{r}_i - \mathbf{r}'|} \right] d\Gamma' = 0.$$

$\mathbf{r}_j$ is the center of panel Γ_j, $\mathbf{r}_i$ denotes the collocation points, and $\mathbf{r}_k$ the locations of the atoms inside the protein. The variables ϕ_1 and $\frac{\partial \phi_1}{\partial \mathbf{n}}$ are constant on each panel, so they are taken out of the integral over panel Γ_j.

Collocating Equation (6) at the center of each panel, we obtain the following $2N \times 2N$ system of equations:

$$\begin{bmatrix} 2\pi I + A & -B \\ 2\pi I - C & \frac{\epsilon_1}{\epsilon_2} D \end{bmatrix} \begin{bmatrix} \phi_1 \\ \frac{\partial \phi_1}{\partial \mathbf{n}} \end{bmatrix} = \begin{bmatrix} Q \\ 0 \end{bmatrix} \tag{7}$$

A, B, C and D are $N \times N$ matrices, I the identity matrix and Q a length-N vector. The elements of A, B, C and D and the Q vector are:

$$A_{ij} = \int_{\Gamma_j} \frac{\partial}{\partial \mathbf{n}} \left(\frac{1}{|\mathbf{r}_{ij}|} \right) d\Gamma \qquad B_{ij} = \int_{\Gamma_j} \frac{1}{|\mathbf{r}_{ij}|} d\Gamma$$
$$C_{ij} = \int_{\Gamma_j} \frac{\partial}{\partial \mathbf{n}} \left(\frac{\exp(-\kappa|\mathbf{r}_{ij}|)}{|\mathbf{r}_{ij}|} \right) d\Gamma \qquad D_{ij} = \int_{\Gamma_j} \frac{\exp(-\kappa|\mathbf{r}_{ij}|)}{|\mathbf{r}_{ij}|} d\Gamma$$
$$Q_i = \sum_{k=0}^{N_c} \frac{q_k}{\epsilon_1} \frac{1}{|\mathbf{r}_{ik}|} \tag{8}$$

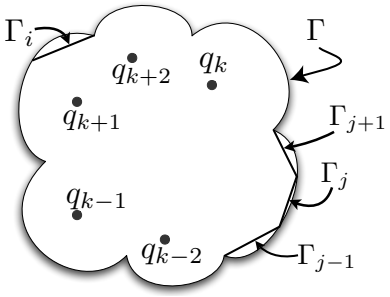


Figure 2: BEM discretization.

$\mathbf{r}_{ij}$ is the distance vector between the collocation point i and panel j; A and C are the double-layer operators, whereas B and D are the single-layer operators.

The integrands in (8) have singularities at the collocation point (when $|\mathbf{r}_{ij}| = 0$), near which standard numerical integration techniques struggle to be accurate. For integrals with a singularity, we use a semi-analytical integration technique[5] and elsewhere the integration is performed with Gauss quadrature rules, using more Gauss points for elements close to the singularity.

[5] Z. Zhu, J. Huang, B. Song, and J. White. Improving the robustness of a surface integral formulation for wideband impedance extraction of 3D structures. In *Proceedings of the 2001 IEEE/ACM Int. Conf. on Computer-Aided Design*, pages 592–597, 2001

The Treecode

The treecode is a fast-summation algorithm capable of performing in $\mathcal{O}(N \log N)$ operations the following $\mathcal{O}(N^2)$ computational pattern:

q_j is the weight, ψ the kernel, $\mathbf{y}_j$ the location of sources and $\mathbf{x}_i$ the location of targets.

$$V(\mathbf{x}_i) = \sum_{j=1}^{N} q_j \psi(\mathbf{x}_i, \mathbf{y}_j) \tag{9}$$

In the treecode, we group the sources $\mathbf{y}_j$ according to their spatial positions into cells or boxes of an octree. Having this, we calculate nearby contributions directly, and approximate distant contributions by writing series expansions representing the sources, around the box centers $\mathbf{y}_c$. Usually, expansions are based either on Taylor series or spherical harmonics. In this work, we used the simpler Taylor expansions, such as the following approximation for ψ:

$$\psi(\mathbf{x}_i, \mathbf{y}_j) = \sum_{||\mathbf{k}||=0}^{P} \frac{1}{\mathbf{k}!} D_{\mathbf{y}}^{\mathbf{k}} (\mathbf{x}_i, \mathbf{y}_j)(\mathbf{y}_j - \mathbf{y}_c)^{\mathbf{k}} \tag{10}$$

$\mathbf{y}_c$ is the position of the cell or box; P is the order of the expansion; $D_{\mathbf{y}}^{\mathbf{k}} = D_{y_1}^{k_1} D_{y_2}^{k_2} D_{y_3}^{k_3}$ the derivative operator; $\mathbf{k} = (k_1, k_2, k_3)$, $\mathbf{k}! = k_1!k_2!k_3!$ and $\mathbf{y}^{\mathbf{k}} = y_1^{k_1} y_2^{k_2} y_3^{k_3}$.

Rearranging the terms in (10), we find the following expression:

$$V(\mathbf{x}_i) = \sum_{||\mathbf{k}||=0}^{P} \frac{1}{\mathbf{k}!} D_{\mathbf{y}}^{\mathbf{k}} \psi(\mathbf{x}_i, \mathbf{y}_c) \sum_j q_j (\mathbf{y}_j - \mathbf{y}_c)^{\mathbf{k}}, \tag{11}$$

In (11), the calculation of multipoles (12) can be done independent of $\mathbf{x}_i$, thus eliminating the $O(N^2)$ scaling.

where

$$a^{\mathbf{k}}(\mathbf{x}_i, \mathbf{y}_j) = D_{\mathbf{y}}^{\mathbf{k}} \psi(\mathbf{x}_i, \mathbf{y}_c) \quad \text{and} \quad m_c^{\mathbf{k}} = \sum_j q_j (\mathbf{y}_j - \mathbf{y}_c)^{\mathbf{k}} \tag{12}$$

are the coefficients of the Taylor expansion and the multipoles, respectively.

For each target point, we decide if a box of sources is distant enough for the Taylor expansion approximation to be acceptable. If it is, we use Equation (11); if not, we ask the same question for the child boxes. We perform this process recursively until a lowest-level box is reached, and we compute the influence of those sources directly. This decision is based on the multipole acceptance criterion θ (MAC), which must be larger than the ratio of the size of the box and the distance between the target and the box center.

Multipole acceptance criterion: $\frac{r_b}{r} < \theta$. Common values for θ are $\frac{1}{2}$ and $\frac{2}{3}$

After the tree structure has been constructed, the treecode algorithm can be divided into four parts:

- ▷ Particle-to-multipole: multipoles in (12) are calculated for the lowest-level boxes.
- ▷ Multipole-to-multipole: multipoles in (12) are calculated for the remaining boxes.
- ▷ Multipole-to-particle: Equation (11) is calculated for the particle-box interactions that satisfy the θ criterion.
- ▷ Particle-to-particle: Direct pairwise interactions between particles that are too close together to satisfy the θ criterion.

Treecode-accelerated BEM

There are several options for standard linear solvers to compute ϕ_1 and $\frac{\partial \phi_1}{\partial \mathbf{n}}$ from Equation (7). We used GMRES, as it accepts non-symmetric matrices. GMRES is dominated by matrix vector products —an $\mathcal{O}(N^2)$ computational pattern—, which are performed n times (usually $n << N$); on the other hand, direct solvers scale as $\mathcal{O}(N^3)$.

Since we evaluate most integrals with Gauss quadrature rules, each matrix-vector product in the GMRES iteration is effectively an N-body problem, of the form of Equation (9). We can use the treecode algorithm to accelerate it, which means that there is no need to explicitly store the matrix and computations scale as $\mathcal{O}(N \log N)$ instead of $\mathcal{O}(N^2)$.

To calculate the derivatives of ψ needed for the coefficients of the Taylor expansion in (12) we use recursive relations available in the literature for Laplace [6] and Yukawa (LPB) potentials.[7]

In the treecode-accelerated BEM, the sources $\mathbf{y}_j$ in (9) are the Gauss points, targets $\mathbf{x}_i$ correspond to the collocation points and the kernel ψ is the free-space Green's function for the Laplace and the linearized Poisson-Boltzmann equations, respectively.

In this N-body problem, the weight q_j in (9) has contributions from the vector involved in the matrix-vector product, the Gauss integration weights and the normal vectors (in the case of $\frac{\partial G}{\partial \mathbf{n}}$).

[6] Z.-H. Duan and R. Krasny. An adaptive treecode for computing nonbonded potential energy in classical molecular systems. *J. Comp. Chem.*, 22(2):184–195, 2001

[7] P. Li, H. Johnston, and R. Krasny. A Cartesian treecode for screened Coulomb interactions. *J. Comp. Phys.*, 228:3858–3868, 2009

Validation

In this section, we present the results of a validation exercise with the `PyGBe` code. Four test cases are presented:

- ▷ Kirkwood sphere with an off-center charge and no Stern layer
- ▷ Kirkwood sphere with an off-center charge and a Stern layer
- ▷ Two separate spheres with a centered charge
- ▷ Lysozyme molecule

We compare our simulations to analytical solutions available in the literature for the first three test cases .[8,9] For more complex geometries, like the Lysozyme, there are no analytical solutions and we contrast our results to previous numerical solutions from Altman and co-workers.[10] We validated both a CPU-only and a GPU version of the code. The timings include I/O to read the mesh files.

In these tests, ϵ is the dielectric constant, κ the inverse of the Debye length, K the number of Gauss points per element, P the order of the expansion, θ the multipole acceptance criterion.$\text{GMRES}_{\text{tol}}$ is the tolerance of the GMRES solver and $\text{thres}_{\text{near}}$ is the threshold distance from a singularity to end of the high-accuracy region, in which 19 Gauss points are used; L is the characteristic length of the integrated triangle.

[8] J. G. Kirkwood. Theory of solutions of molecules containing widely separated charges with special application to zwitterions. *J. Chem. Phys.*, 2, 1934

[9] B. Kim and X. Song. Calculations of the second virial coefficients of protein solutions with an extended fast multipole method. *Phys. Rev. E*, 83, 2011

[10] M. D. Altman, J. P. Bardhan, B. Tidor, and J. K. White. FFTSVD: A fast multiscale boundary-element method solver suitable for BioMEMS and biomolecule simulation. *IEEE Trans. Comput. Aid. D.*, 25:274–284, 2006. DOI: 10.1109/TCAD.2005.855946

Kirkwood sphere with an off-center charge and no Stern layer

We consider a spherical molecule of radius 4Å, with a charge of $1e^-$ located at a distance of 2Å from the center, at $(1, 1, \sqrt{2})$. We calculated the solvation energy and compared it to the analytical solution by Kirkwood. Figure 3 is a sketch of the situation.

The error plot in Figure 4(a) shows that the simulation is converging with the expected $\frac{1}{N}$ rate. The time to solution plot, (b), shows that the runtime scales slightly worse than $O(N \log N)$; this is due to the fact that in this formulation the condition number depends on the size of the system; thus, as the number of elements increases, the GMRES solver needs more iterations to converge; number of iterations is shown in (d). An important result is that the time per iteration does scale as $O(N \log N)$, showing that it is the treecode which dominates the calculation runtime. The plots in Figure 4 include results for CPU-only runs, both to compare timings and to check that both CPU and GPU codes converge in the expected way.

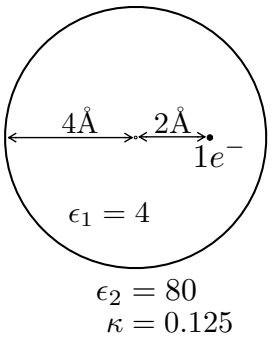
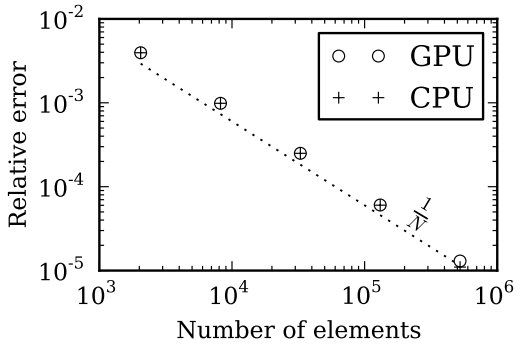


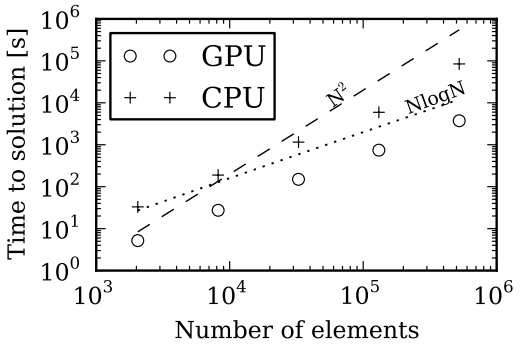
Figure 3: Sketch of the Kirkwood sphere.

ϵ_1	ϵ_2	κ	K	P	θ	$\text{GMRES}_{\text{tol}}$	$\text{thres}_{\text{near}}$
4	80	0.125	3	12	0.5	10^{-6}	$2L$

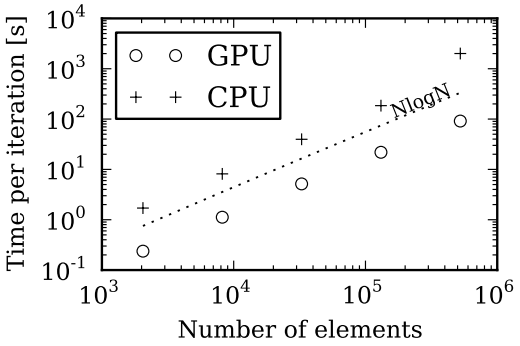
Table 1: Simulation parameters for Kirkwood sphere without Stern layer.



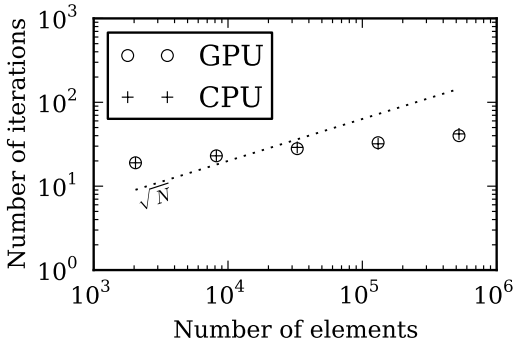
(a) Relative error convergence.



(b) Time to solution.



(c) Average time per iteration.



(d) Number of iterations.

Figure 4: Results for Kirkwood sphere without Stern layer using 2048, 8192, 32768, 131072 and 524288 boundary elements.

Kirkwood sphere with an off-center charge and a Stern layer

A natural extension from the test using a Kirkwood sphere with one surface is to include a Stern layer. This simulation requires two surfaces, as shown in Figure 5, where one surface interfaces a spherical region of radius $R = 4$Å with the Stern layer — having the same dielectric constant as the solvent but no ions — and the second one limits the Stern layer and the solvent. We placed an off-center charge of $1e^-$ at $(1, 1, \sqrt{2})$ in the innermost region, then calculated the solvation energy and compared it to the analytical solution by Kirkwood.

The results shown in Figure 6 are very similar to those in Figure 4. Again, the error decreases as $\frac{1}{N}$ and the time to solution is dominated by the matrix-vector product computed with the treecode. As shown in Figure 6(b), the runtime scales as $\mathcal{O}(N \log N)$.

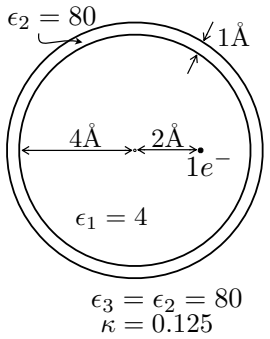
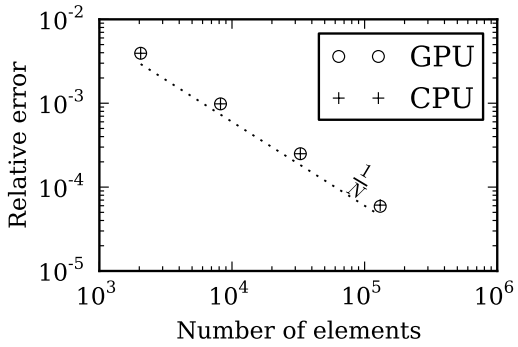


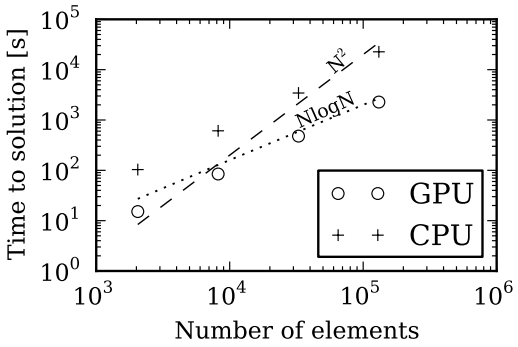
Figure 5: Sketch of the Kirkwood sphere with a Stern layer and an off-center charge.

ϵ_1	ϵ_2	ϵ_3	κ	K	P	θ	GMRES$_{\text{tol}}$	thres$_{\text{near}}$
4	80	80	0.125	3	12	0.5	10^{-6}	$2L$

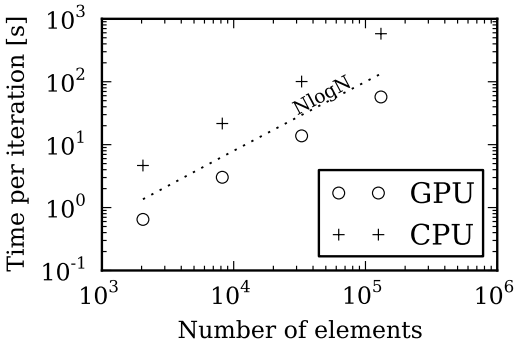
Table 2: Simulation parameters for Kirkwood sphere with Stern layer.



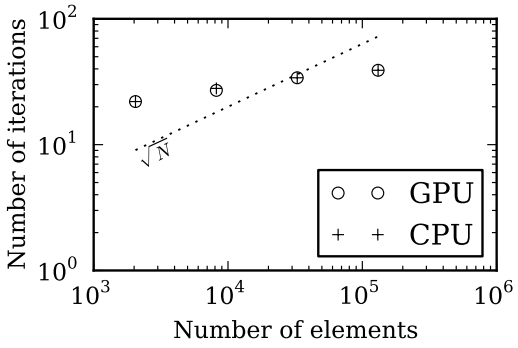
(a) Relative error convergence.



(b) Time to solution.



(c) Average time per iteration.



(d) Number of iterations.

Figure 6: Results for for Kirkwood sphere with Stern layer using 2048, 8192, 32768 and 131072 boundary elements per surface.

Two separate spheres with a centered charge

In order to show the capability of the code to calculate interaction energies, we simulated two spherical molecules of radius $R = 4$Å separated a distance $D = 20$Å, each with a centered charge of $1e^-$. We calculated free energy that arises from the difference in potential between one molecule alone in vacuum and both molecules immersed in the solvent, and compared it to the analytical solution in the work by Kim and Song. The test parameters are shown in Table 3.

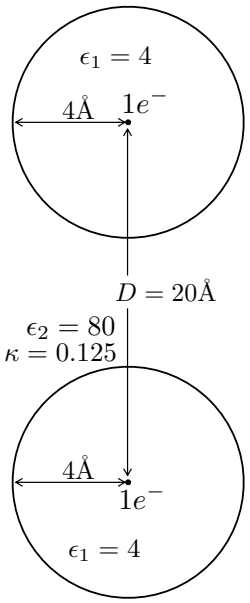


Figure 7: Sketch of two spherical molecules in the same solvent.

Lysozyme

As a more realistic demo, we computed the solvation energy for a Lysozyme molecule and compare it to the FFTSVD code from Altman and co-workers. The molecule data was downloaded from the Protein Data Bank (PDB code 1HEL) and processed with MSMS to obtain the solvent-excluded area discretized in flat triangles. The Lysozyme molecule contains 1323 atoms.

As shown in Table 4, we used coarser parameters for the Lysozyme than for the sphere case, because in the sphere simulations we needed very high accuracy in order to show convergence to 10^{-5} error. In order to ensure that there were no sources of errors higher than the discretization error, we needed to use more Gauss points per element, a higher value of P —the order of the Taylor expansion—, and a tighter GMRES tolerance. In this case, which is more biologically relevant than the Kirkwood sphere, the complicated geometry generates a higher discretization error and we do not need to have such fine parameters.

ϵ_1	ϵ_2	κ	K	P	θ	$\text{GMRES}_{\text{tol}}$	$\text{thres}_{\text{near}}$
1	10	0.125	3	10	0.5	10^{-6}	$2L$

Table 3: Simulation parameters for the two-sphere problem.

ϵ_1	ϵ_2	κ	K	P	θ	$\text{GMRES}_{\text{tol}}$	$\text{thres}_{\text{near}}$
4	80	0	1	2	0.6	10^{-4}	$1.25L$

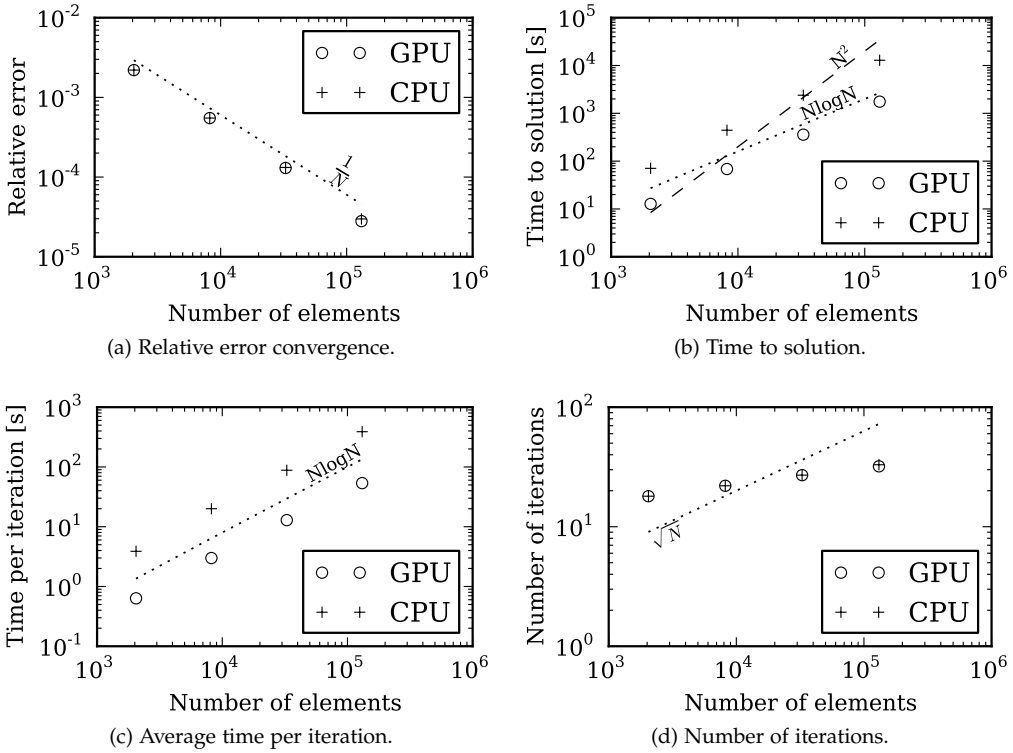
Table 4: Simulation parameters for Lysozyme simulation.

Implementation details

All of the user-visible code in `PyGBe` is `Python`, and we interface to `Cuda` via `PyCuda` [11] for the most computationally intensive parts of the algorithm. The parts that run on GPU are the following:

- ▷ Generation of the right hand side;

[11] Andreas Klöckner, Nicolas Pinto, Yunsup Lee, B. Catanzaro, Paul Ivanov, and Ahmed Fasih. PyCUDA and PyOpenCL: A scripting-based approach to GPU run-time code generation. *Parallel Computing*, 38(3):157–174, 2012



(a) Relative error convergence.

(b) Time to solution.

(c) Average time per iteration.

(d) Number of iterations.

Figure 8: Results for two spherical molecules in the same solvent.

- ▷ multipole-to-particle interactions in the treecode;
- ▷ particle-to-particle interactions in the treecode;
- ▷ calculation of the reaction potential at the locations of point charges.

To improve runtime efficiency, some parts of `PyGBe` were written in `C++` and wrapped in `Python` (specifically, the calculation of the multipoles). Also, in the CPU version of the code, the multipole-to-particle and particle-to-particle interactions of the treecode were written in `C++` and wrapped in `Python`. To interface between `C++` and `Python`, `PyGBe` uses the `swig` library.[12]

The code calculates the right-hand side and the reaction potential by directly computing all pairwise interactions, even though they could be done with a treecode. For the problems tested so far, this is the fastest way to perform these calculations. However, in the future, we can consider also using a treecode in these parts of the algorithm so that they scale better to larger problems.

All tests presented in this document were done in double precision on a single INTEL Xeon (2.67 GHz) CPU core with a NVIDIA Tesla C2075 GPU.

[12] `http://www.swig.org`

	PyGBe		
N	GPU	CPU	*FFTSVD*
14398	−573.8	−573.7	−572.8
21944	−516.2	−516.9	−515.6
52780	−498.8	−499.0	−498.0
101382	−492.4	−492.8	−491.6
214618	−490.9	−491.3	–

Table 5: Solvation energy comparison between PyGBe and FFTSVD for the Lysozyme case.

We release the PyGBe code under the MIT License, and maintain a Bitbucket repository at https://bitbucket.org/cdcooper/pygbe. The repository also includes instructions for running the test cases described in this document.

Acknowledgement Jaydeep Bardhan aided us throughout this work, including with the mathematical derivations, getting through stumbling blocks in the process of obtaining code correctness, providing a reference implementation of the semi-analytical integrals (according to Zhu et al., 2001) and supplying the initial mesh files for the lysozyme test. He also providedthe fftsvd code used for the comparison in Table 5.

References

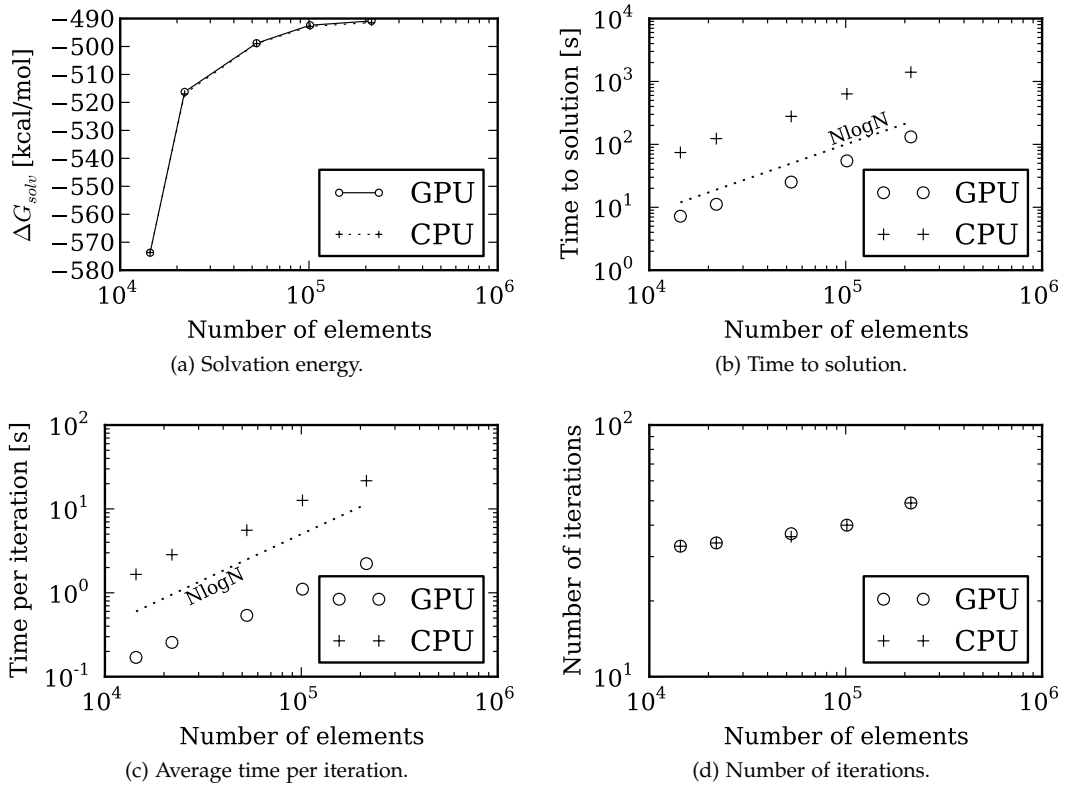
M. D. Altman, J. P. Bardhan, B. Tidor, and J. K. White. FFTSVD: A fast multiscale boundary-element method solver suitable for BioMEMS and biomolecule simulation. *IEEE Trans. Comput. Aid. D.*, 25:274–284, 2006. doi: 10.1109/TCAD.2005.855946.

M. D. Altman, J. P. Bardhan, J. K. White, and B. Tidor. Accurate solution of multi-region continuum electrostatic problems using the linearized Poisson–Boltzmann equation and curved boundary elements. *J. Comput. Chem.*, 30:132–153, 2009.

Z.-H. Duan and R. Krasny. An adaptive treecode for computing nonbonded potential energy in classical molecular systems. *J. Comp. Chem.*, 22(2):184–195, 2001.

B. Kim and X. Song. Calculations of the second virial coefficients of protein solutions with an extended fast multipole method. *Phys. Rev. E*, 83, 2011.

J. G. Kirkwood. Theory of solutions of molecules containing widely separated charges with special application to zwitterions. *J. Chem. Phys.*, 2, 1934.



(a) Solvation energy.

(b) Time to solution.

(c) Average time per iteration.

(d) Number of iterations.

Figure 9: Results for the Lysozyme molecule for 14398, 21944, 52780, 131382 and 214618 elements.

Andreas Klöckner, Nicolas Pinto, Yunsup Lee, B. Catanzaro, Paul Ivanov, and Ahmed Fasih. PyCUDA and PyOpenCL: A scripting-based approach to GPU run-time code generation. *Parallel Computing*, 38(3):157–174, 2012.

P. Li, H. Johnston, and R. Krasny. A Cartesian treecode for screened Coulomb interactions. *J. Comp. Phys.*, 228:3858–3868, 2009.

A. Warshel, P. K. Sharma, M. Kato, and W. W. Parson. Modeling electrostatic effects in proteins. *Biochim. Biophys. Acta*, 1764:1647–1676, 2006.

B J. Yoon and A. M. Lenhoff. A boundary element method for molecular electrostatics with electrolyte effects. *J. Comput. Chem.*, 11 (9):1080–1086, 1990.

Z. Zhu, J. Huang, B. Song, and J. White. Improving the robustness of a surface integral formulation for wideband impedance extraction of 3D structures. In *Proceedings of the 2001 IEEE/ACM Int. Conf. on Computer-Aided Design*, pages 592–597, 2001.

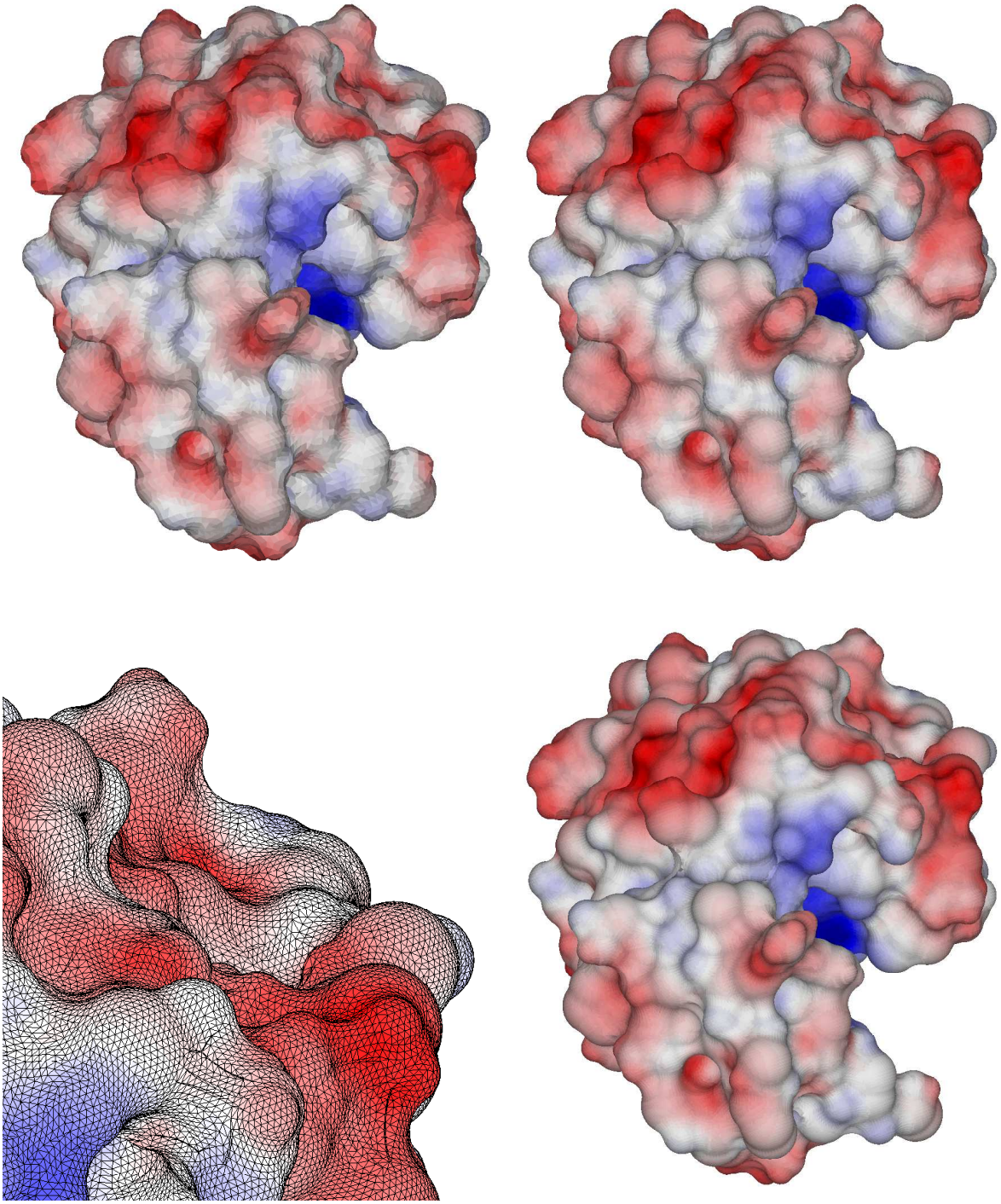


Figure 10: Electric potential on the surface of the Lysozyme molecule for 52780 (top left), 131382 (top right) and 214618 (bottom right) elements. The bottom left figure is a close-up view of the mesh for the case with 214618 elements.