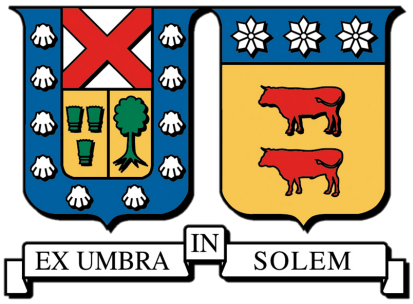
UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INGENIERÍA MECÁNICA
VALPARAÍSO, CHILE



COMPARACIÓN DE FORMULACIONES DE LA ECUACIÓN DE POISSON-BOLTZMANN APLICANDO EL MÉTODO DE ELEMENTOS DE FRONTERA

STEFAN DAVID SEARCH TOLOZA

MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL MECÁNICO

Profesor Guía: PhD Christopher Cooper Villagrán

Profesor Correferente: PhD Elwin van't Wout

Marzo - 2021

Acknowledgements

Firstly, I would like to thank my family whom have always encouraged me to explore how day to day objects work and function. From mending our garden machinery to pottering around around at the factory, the place in which I got my first taste of industrial machinery. For having always provided me with everything I have needed for my education and to get to this point in life.

To all the friends I have made during the last 7 years, the good times we've had and the many things we've done. The times spent talking in the "Salita", swimming in the pool at uni, relaxing in the "Patio del Cañón", or going to eat and drink. These were much needed times of relaxation, reflection and to let off a bit of steam.

To those who have been close to me over this last year, through what have been strange and uncertain times, you have encouraged me and helped me to keep focused. For always being there to message if I needed something or to video chat just to talk and relax a bit.

Thank you to Prof. Christopher Cooper who has helped me complete this thesis, encouraging me with his research ideas and his vision to connect different lines of his student's investigation into a larger collective. For always having his door open to talk about this work and other broader points of academia. Thank you to Prof. Elwin van't Wout who showed great interest in my work, providing me with very useful information and comments that helped me greatly.

Finally, to the university staff, for the help over these years, and to the campus "Casa Central", with its green spaces, beautiful buildings and spectacular view, having been my home away from home and a place I will miss dearly.

Abstract

Electrostatic biomolecular interactions are an important area of study to help understand many biological processes. The use of an implicit solvent model can be used to simulate different situations. In this work the Poisson-Boltzmann implicit solvent model was used to calculate the solvation energy of different biomolecules.

The Poisson-Boltzmann model can be solved via application of the Boundary Element Method (BEM). Various formulations can be derived, such as the direct and Juffer formulations. Preconditioners can also be applied to these formulations in order to improve the conditioning of the resulting system. The objective of this study was to compare, from a computational point of view, the use of different formulations and preconditioners in the calculation of solvation energy.

Using the Poisson-Boltzmann model and applying BEM, different formulations were derived, including a generalisation of Juffer, this was given the name alpha-beta. Calculations of the solvation energy for different biomolecules were performed with various combinations of preconditioner, formulation and discrete form. Several parameters were compared including total time, memory usage, GMRES time and GMRES iterations.

The results showed that in the case of the 1BPI protein the best combination was weak form discretisation $\alpha = 1$ / $\beta = 1$ with block diagonal preconditioning. Calderón preconditioning performed very well in some cases, lowering greatly the iterations required, however, the total time was never the lowest. It was also seen that there appeared to be a dependency between the molecule size and the formulation with the overall lowest total time. This was seen in both simplified spheres and to a point using real life molecules. Further research is required to understand this relation and possibly find a rule to aid in the selection of the combination to be used given a particular biomolecule, in order to obtain the lowest total time.

Resumen

Las interacciones biomoleculares electrostáticas son un área de estudio importante, lo cual ayuda a comprender muchos procesos biológicos. Se puede utilizar un modelo de solvente implícito para simular diferentes situaciones. En este trabajo se utilizó el modelo de solvente implícito de Poisson-Boltzmann para calcular la energía de solvatación de diferentes biomoléculas.

El modelo de Poisson-Boltzmann se puede resolver mediante la aplicación del método de elementos de frontera (BEM por sus siglas en inglés). Se puede derivar varias formulaciones, tales como la formulación directa y Juffer. También se puede aplicar precondicionadores a estas formulaciones para mejorar el acondicionamiento del sistema resultante. El objetivo de este estudio fue comparar, desde un punto de vista computacional, el uso de diferentes formulaciones y precondicionadores en el cálculo de la energía de solvatación.

Usando el modelo de Poisson-Boltzmann y aplicando el método de elementos de frontera, se derivaron diferentes formulaciones, incluyendo una generalización de Juffer, a esto se le dio el nombre de alfa-beta. Los cálculos de la energía de solvatación para diferentes biomoléculas se realizaron con varias combinaciones de precondicionador, formulación y forma discreta. Se compararon varios parámetros, incluido el tiempo total, el uso de memoria, el tiempo de GMRES y las iteraciones de GMRES.

Los resultados mostraron que en el caso de la proteína 1BPI la mejor combinación fue la discretización de forma débil $\alpha = 1$ / $\beta = 1$ con precondicionamiento “Block-Diagonal”. El precondicionamiento de Calderón funcionó muy bien en algunos casos, reduciendo en gran medida las iteraciones requeridas, sin embargo, el tiempo total nunca fue el más bajo. También se vio que parecía haber una dependencia entre el tamaño de la molécula y la formulación con el tiempo total más bajo. Esto se vio tanto en esferas simplificadas como en el uso de moléculas reales. Se requiere más investigación para comprender esta relación, y posiblemente encontrar

una regla que ayude en la selección de la combinación a utilizar dada una biomolécula en particular, con el fin de obtener el tiempo total más bajo.

Contents

1 Introduction

The interactions between biomolecules is an important area of study, given that these govern many phenomenons in the biological world. There are different elements that play a role in these interactions, a major one of these being electrostatics. These processes often occur in an environment containing a mixture of water and salt.

There is a variety of ways to computationally analyse these systems, each with there own advantages and disadvantages. One of the most exact methods is using molecular dynamics (MD), were by each of the molecules of water is treated explicitly. This can be useful in some applications but can take prohibitively long depending on the problem to resolve. If knowledge of the dynamics of the system is not required and the parameter of study is at equilibrium, then a more rapid alternative can be the use of continuum electrostatics, with an implicit solvent model applied to describe the aqueous region.

A well known implicit solvent model is the Poisson-Boltzmann model, which couples the Poisson and Poisson-Boltzmann equations. This model is widely used with a variety of different software packages that come with incorporated solvers. It can be resolved utilising both volumetric and boundary methods, each having there own advantages and disadvantages. In this work we focused on its resolution using boundary integral methods. Having transformed the problem into a boundary integral problem one can then use the boundary element method (BEM) to solve the problem numerically. Different BEM formulations have been proposed previously, however the most known and investigated are the Direct formulation shown by Yoon and Lenhoff [1], and the Juffer formulation proposed by Juffer and coworkers. [2].

It can noted that the Juffer formulation appears to be mathematically similar in form to some problems solved in the field of acoustics. In this field more generalised formulations exist, so part of this work focused on seeing if these

ideas could be applied to electrostatics. Another idea, inspired from acoustics, is the proposed use of Calderón preconditioning. This did not appear to have been tried for the case of electrostatics so part of this work was to test this type of preconditioning. There has also been published a preconditioning technique for the direct formulation but information about its use with Juffer could not be found. What was clear was that there exists various different formulations and preconditioning methods, however these had not been tested or studied together under the same circumstances.

The main objective of this work was to compare these different formulations and preconditioning techniques in order to understand which is best, based on a series of different parameters such as computational time and GMRES iterations required. This was done by implementing the different formulations and preconditioners using a the open-source software bempp-cl [3], which allows for quick and easy implementation of the different formulations.

The specific goals of this work were:

- To study the mathematical model that describes the electrostatic interactions between molecules and an ionic solution.
- To apply the boundary element method (BEM) to the case of electrostatics in biomolecules.
- To carry out an analysis to compare the different formulations using factors such as calculation time, iterations necessary to solve the system of equations and required computational memory.
- Define the optimal formulation for the calculation of the solvation energy of different biomolecules.

2 Theory

2.1 Implicit Solvent Model

2.1.1 Background

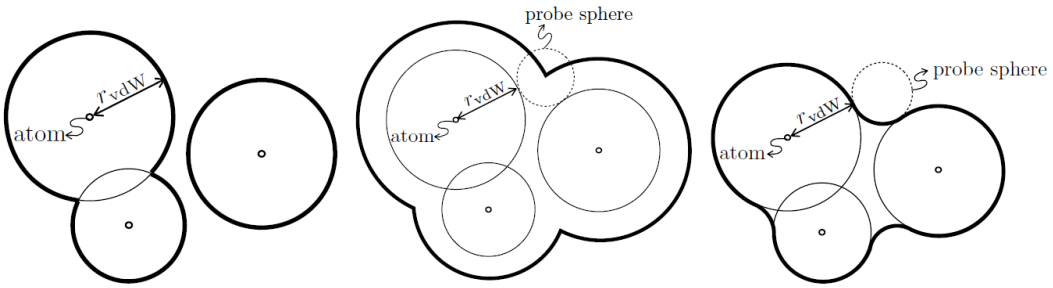
Solvent models are used to make simulations and thermodynamic calculations of reactions and processes which occur in solution. These models can either be explicit or implicit. In the first case, each molecule of the solvent is treated explicitly, taking into account the position and degrees of freedom of each. On the other hand, implicit models treat the solvent as a continuous medium surrounding the solute. Each has their advantages and disadvantages, explicit models are able to reproduce some phenomena not possible with implicit models, however they require far more computational resources.

2.1.2 Poisson-Boltzmann Implicit Solvent Model

A well known and widely used implicit solvent model in biomolecular systems is the Poisson-Boltzmann model, which separates the problem into two distinct domains, the solute and the solvent. Continuum electrostatic theory in a dielectric medium and a distribution of single point charges in a dielectric medium, are used to represent the solvent and solute regions, respectively. In biomolecular systems the solvent is normally considered to be a solution of water and salt, with a particular concentration, whereas the solute is a protein or some other biomolecule.

The interface between these domains can be defined in different ways. First there is the Van der Waals surface, which considers the solute region to be anything within the Van der Waals radius of each atom. Next, there is the solvent accessible surface (SAS), which is formed by tracing the centre of a spherical probe the size of a water molecule (1.4 Å), as it rolls around the solute defined by the Van der Waals surface. Finally there is the solvent excluded surface (SES), which

represents the closest a water molecule can get to the solute. This is formed by tracing the inner edge of the same spherical probe as used to form the SAS. The model in this work considered the SES as the boundary between the domains. The following figure (1) shows a representation of the different surfaces.



(a) van de Waals surface (b) Solvent-accessible surface (c) Solvent-excluded surface

Figure 1: Different surface definitions [4]

In the case of the solute, we know that the electrostatic potential can be described by the macroscopic Poisson equation of electrostatics,

$$\nabla^2\phi = -\frac{\rho}{\epsilon}, \tag{1}$$

where ϕ is the electrostatic potential, ρ is the charge density distribution and ϵ is the permittivity of the domain. As the solute is represented by a distribution of point charges, the charge density distribution can be described with the following:

$$\rho = q\delta(r') = \sum_{k}^{N} q_k \delta(r, r_k), \tag{2}$$

where N is the total number of atoms, q_k is the charge associated to atom k, δ is the Dirac delta function, r is position of evaluation and r_k the position of the atom k. Replacing this in equation (1) we obtain

$$\nabla^2\phi = -\sum_{k}^{N} \frac{q_k}{\epsilon}\delta(r, r_k). \tag{3}$$

In the case of the solvent, for most biomolecular systems this consists of water

and salt. The salt when dissolved in the solution produces ions that contribute to the electric field and which are free to move. When an electric field is applied the spatial distribution of the salt ions will be changed, this means that the charge density distribution will be dependent on the electrostatic potential ($\rho = \rho(\phi)$). There has been previous analysis of this type of system (Hill, An introduction to statistical thermodynamics [5]), here follows a brief outline of the result of this analysis.

Assuming that the ions are hard spheres and the solvent has no molecular behaviour, we can describe the distribution of salt ions using the radial distribution function. Substituting the charge density distribution in the Poisson equation (1) we obtain

$$\nabla^2\phi = -\frac{\rho}{\epsilon} = -\frac{1}{\epsilon}\sum_{i}^{N_s} q_i c_i(r) = -\frac{1}{\epsilon}\sum_{i}^{N_s} q_i c_{0i} e^{\frac{-\phi q_i}{k_B T}}, \tag{4}$$

where N_s is the number of species of ions, q_i and c_{0i} are the charge and mean number density of ion species i, k_B is the Boltzmann constant, and T is the absolute temperature of the solvent. Salt (sodium chloride) when dissolved produces two types of ions, the chlorine ion with charge e^- and the sodium ion with charge e^+. Assuming that they have the same mean number of density, equation (4) becomes

$$\nabla^2\phi = -\frac{e^+ n_0}{\epsilon}\left[e^{\frac{-\phi e^+}{k_B T}} - e^{\frac{\phi e^+}{k_B T}}\right] = \frac{2n_0 e^+}{\epsilon} sinh\left(\frac{\phi e^+}{k_b T}\right). \tag{5}$$

For cases where $\phi q << k_B T$ the first order approximation of the exponential is

$$e^{\frac{-\phi q}{k_B T}} \approx 1 + \frac{\phi q}{k_B T}, \tag{6}$$

Thus, the linearised version of equation (4) reads

$$\nabla^2\phi = \frac{\phi}{\epsilon k_B T}\sum_i^{N_s} n_{0i}q_i = \kappa^2\phi, \tag{7}$$

$$\nabla^2\phi - \kappa^2\phi = 0. \tag{8}$$

This is known as the linearised Poisson-Boltzmann equation. Here κ is the inverse of the Debye length, which in the case of salt, as mentioned above, corresponds to

$$\kappa^2 = \frac{2n_0e^+}{\epsilon k_B T}. \tag{9}$$

It has units of L^{-1} and in this application is commonly expressed in Angstroms Å. We also have ϕ, which is the electrostatic potential and has units of $ML^2T^{-3}I^{-1}$, and is commonly measured in *volts*.

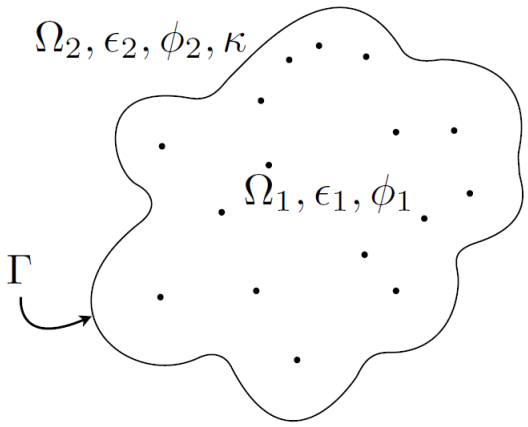


Figure 2: Depiction of dissolved protein. [4]

Figure (2) shows a dissolved protein, with the solute region (Ω_1) and the solvent region (Ω_2) separated by the boundary (Γ). The dots in the protein region portray the point charges at the location of the atoms. The solute domain has a permittivity of ϵ_1 and the electrostatic potential is represented by ϕ_1. Meanwhile,

the solvent region has a permittivity of ϵ_2, an inverse Debye length of κ and the electrostatic potential is represented by ϕ_2.

At the interface between the two domains suitable boundary conditions must be applied. We know from electrostatic theory that both the electrostatic potential (ϕ) and normal electric displacement ($\epsilon\frac{\partial\phi}{\partial n}$) must be continuous across the boundary. Taking the equations for each region presented above and adding these boundary conditions, we obtain the following system of equations:

$$\begin{cases} \nabla^2\phi_1 = -\sum_k \frac{q_k}{\epsilon_1}\delta(r, r_k) & \text{in } \Omega_1, \\ \nabla^2\phi_2 - \kappa^2\phi_2 = 0 & \text{in } \Omega_2, \\ \phi_1 = \phi_2 & \text{on } \Gamma, \\ \epsilon_1\frac{\partial\phi_1}{\partial n} = \epsilon_2\frac{\partial\phi_2}{\partial n} & \text{on } \Gamma. \end{cases} \tag{10}$$

The coupling of these two equations is of much interest in the biophysics community and there are many widely used software packages available for performing calculations. The equation system can be solved numerically using volumetric methods, such as finite elements or finite difference, which is the most popular method and for which there exists many software packages. This work, however, utilised boundary integral methods, which have also been successfully used for this application, shown in work by Yoon and Lenhoff [1] and also by Juffer and coworkers [2]. Software packages that include solvers based on these formulations include PyGBe [6] and TABI [7].

2.2 Solvation Energy

The solvation energy is a parameter of interest in biomolecular systems and can be calculated from the electrostatic potential. In the case of one dissolved protein, it corresponds to the difference in energy between the protein in a vacuum and

the protein in a dissolved state. It can be calculated as

$$E_{solv} = \frac{1}{2}\int_{\Omega} \rho\phi_{\text{reac}} = \sum_{k=0}^{N_q} q_k \phi_{\text{reac}}(x_k), \tag{11}$$

where by ϕ_{reac} is the reaction potential and ρ is the charge density. In the case of the implicit solvent model, the charge distribution is represented with point charges and so the integral becomes a sum, with N_q being the number of point charges that make up the protein and q_k the charge of each point charge k. The reaction potential is given by

$$\phi_{\text{reac}} = \phi_{\text{total}} - \phi_{\text{Coul}}, \tag{12}$$

where ϕ_{Coul} is the Coulomb potential and ϕ_{total} is the total potential.

2.3 Green's Function

A Green's function is a useful technique for the resolution of differential equations, including ordinary differential equations with initial values or boundary value conditions, or inhomogeneous partial differential equations with boundary conditions. A Green's function is defined as

$$\mathcal{L}G(x, y) = \delta(x - y), \tag{13}$$

where $\mathcal{L}$ is an arbitrary linear differential operator and $\delta(x - y)$ is the Dirac delta function. x and y are both points in the space. As an example, the three-dimensional Green's function for the Laplace equation is

$$G_L(x, y) = \frac{1}{4\pi|x - y|}. \tag{14}$$

2.4 Boundary Element Method (BEM)

The boundary element method (BEM) is a numerical method used for solving linear PDEs. It involves converting the system to the form of a boundary integral equations (BIE), so can only be used in cases when this is possible, that is, the Green's function of the PDE must be known. BEM has many advantages over finite element methods (FEM), particuly because only the boundary must be discretised, and not the entire domain. This means that BEM can be used to easily treat problems where the exterior domain is unbounded, and also has the added advantage that the mesh is of smaller size. The disadvantage of BEM, however, is that knowledge of the Green's function for the PDE is required, something that is not the case for FEM.

2.4.1 Laplace Example

The Poisson equation governs many phenomena in the physical world and can be solved through the application of BEM. Making the simplification whereby the function on the right hand side is zero, it becomes the Laplace equation (15), which is also solvable with the application of BEM. Taking the case of an interior Laplace problem in the domain Ω^-, surrounded by the boundary Γ, we have

$$\nabla^2\phi = 0 \qquad \text{in } \Omega^-. \tag{15}$$

Taking this equation and multiplying by an arbitrary function (ω), we obtain

$$\omega\left(\nabla^2\phi\right) = 0. \tag{16}$$

The two following equations are vector derivative identities:

$$\nabla\cdot(\omega\nabla\phi) = (\nabla\omega)\cdot(\nabla\phi) + \left(\omega\nabla^2\phi\right), \tag{17}$$

$$\nabla \cdot (\phi \nabla \omega) = (\nabla \omega) \cdot (\nabla \phi) + \left(\phi \nabla^2 \omega\right). \tag{18}$$

Subtracting equation (18) from (17) gives

$$\nabla \cdot (w \nabla \phi - \phi \nabla w) = w \nabla^2 \phi - \phi \nabla^2 w. \tag{19}$$

Divergence theorem states that

$$\int_\Omega (\nabla \cdot F)\, d\Omega = \int_\Gamma F \cdot d\Gamma. \tag{20}$$

Applying this to equation (19), and knowing that $\nabla^2 \phi = 0$ we obtain

$$\int_\Omega \left(-\phi \nabla^2 w\right) d\Omega = \int_\Gamma (w \nabla \phi - \phi \nabla w) \cdot d\Gamma, \tag{21}$$

$$\int_\Gamma (w \nabla \phi) \cdot d\Gamma - \int_\Gamma (\phi \nabla w) \cdot d\Gamma - \int_\Omega \left(\phi \nabla^2 w\right) d\Omega = 0. \tag{22}$$

If we apply a Green's function to resolve the volume integral. This is done by making the arbitrary function w the corresponding Green's function, giving

$$\int_\Gamma (G(x,y) \nabla \phi) \cdot d\Gamma - \int_\Gamma (\phi \nabla G(x,y)) \cdot d\Gamma - \int_\Omega \left(\phi \nabla^2 G(x,y)\right) d\Omega = 0. \tag{23}$$

As ∇^2 is a linear differential operator we know that $\nabla^2 G(x,y) = \delta(x,y)$. This can be substituted, yielding

$$\int_\Gamma (G(x,y) \nabla \phi) \cdot d\Gamma - \int_\Gamma (\phi \nabla G(x,y)) \cdot d\Gamma - \int_\Omega (\phi \delta(x,y))\, d\Omega = 0, \tag{24}$$

$$\int_\Gamma \frac{\partial G(x,y)}{\partial n} \phi(y) d\Gamma(y) - \int_\Gamma G(x,y) \frac{\partial \phi(y)}{\partial n} d\Gamma(y) + \phi_{(x)} = 0. \tag{25}$$

The Green's function for use in this case depends on the number of dimensions. For two dimensions it is

$$G(x,y) = -\frac{1}{2\pi} log|x-y|, \tag{26}$$

and in the case of three dimensions it reads

$$G(x,y) = \frac{1}{4\pi|x-y|}. \tag{27}$$

In the case that x is equal to y there is a singularity and this must be taken into account and treated. In three dimensions, assuming that the boundary is smooth, this can be done by placing a semi-sphere around the point of the singularity and taking the limit when the radius tends to zero, shown as follows:

$$\int_\Gamma \frac{\partial G(x,y)}{\partial n}\phi(y)d\Gamma = \lim_{\epsilon\to 0}\int_\Gamma -\left(\frac{1}{4\pi\epsilon^2}\right)\phi(y)d\Gamma = \lim_{\epsilon\to 0}\left(\frac{2\pi\epsilon^2}{4\pi\epsilon^2}\phi(y)\right) = -\frac{\phi(y)}{2}, \tag{28}$$

$$\int_\Gamma G(x,y)\frac{\partial \phi(y)}{\partial n}d\Gamma = \lim_{\epsilon\to 0}\int_\Gamma \left(\frac{1}{4\pi\epsilon}\right)\frac{\partial\phi(y)}{\partial n}d\Gamma = \lim_{\epsilon\to 0}\left(\frac{2\pi\epsilon^2}{4\pi\epsilon}\frac{\partial\phi(y)}{\partial n}\right) = 0. \tag{29}$$

Taking these results and substituting into the previous equation we end up with

$$\frac{\phi_{(x)}}{2} + \int_\Gamma \frac{\partial G(x,y)}{\partial n}\phi(y)d\Gamma = \int_\Gamma G(x,y)\frac{\partial\phi(y)}{\partial n}d\Gamma. \tag{30}$$

After obtaining the results on the boundary, to be able to calculate the internal values one must apply the following equation:

$$\phi_{(x)} = \int_\Gamma G(x,y)\frac{\partial\phi(y)}{\partial n}d\Gamma - \int_\Gamma \frac{\partial G(x,y)}{\partial n}\phi(y)d\Gamma, \tag{31}$$

which does not have treatment for singularities as these do not exist.

2.4.2 Trace, Potential and Boundary Operators

Operators can be used to simplify the notation and here follows the definition of these operators.

Taking the interior bounded domain $\Omega^- \subset \mathbb{R}^3$ and the unbounded external domain Ω^+, separated by the smooth interface Γ, we define the Dirichlet and Neumann traces operators as follows:

$$\gamma_D^{\pm} f(x) = \lim_{\Omega^{\pm} \ni y \to x} f(y) \qquad \text{for } x \in \Gamma, \tag{32}$$

$$\gamma_N^{\pm} f(x) = \lim_{\Omega^{\pm} \ni y \to x} \nabla f(y) \cdot \hat{n}(x) \qquad \text{for } x \in \Gamma. \tag{33}$$

It is important to note that both the interior and exterior Neumann traces $\gamma_N^{\pm}$ are based on the outward facing normal $\hat{n}$. From these traces we can also define the Cauchy trace as $\gamma^{\pm} = \begin{bmatrix} \gamma_D^{\pm} & \gamma_N^{\pm} \end{bmatrix}^T$ which can be used to shorten and simplify notation.

As we know the Green's function can be used to solve differential equations, and that of the Laplace equation is given in equation (14). With this we can define the potential integral operators $\mathcal{V}$ and $\mathcal{K}$, which are known as the single-layer and double-layer potential operator, respectively:

$$[\mathcal{V}_n \mu](x) = \int_\Gamma G_n(x, y) \mu(y) d\Gamma(y) \qquad \text{for } x \in \Omega^{\pm}, \tag{34}$$

$$[\mathcal{K}_n \nu](x) = \int_\Gamma \frac{\partial G_n(x, y)}{\partial n(y)} \nu(y) d\Gamma(y) \qquad \text{for } x \in \Omega^{\pm}, \tag{35}$$

where μ and ν are arbitrary potential functions and $G_n(x, y)$ is the Green's function. The sub-index n is used to define the Green's function, as this depends

on the particular problem. These operators map from functions on the interface to the volume.

We can also define the boundary integral operators, which map from a function on the interface to another function also on the interface:

$$[V_n\mu](x) = \int_\Gamma G_n(x,y)\mu(y)d\Gamma(y) \qquad \text{for } x \in \Gamma, \tag{36}$$

$$[K_n\nu](x) = \int_\Gamma \frac{\partial G_n(x,y)}{\partial n(y)}\nu(y)d\Gamma(y) \qquad \text{for } x \in \Gamma, \tag{37}$$

$$[T_n\mu](x) = \frac{\partial}{\partial n(x)}\int_\Gamma G_n(x,y)\mu(y)d\Gamma(y) \qquad \text{for } x \in \Gamma, \tag{38}$$

$$[D_n\nu](x) = -\frac{\partial}{\partial n(x)}\int_\Gamma \frac{\partial G_n(x,y)}{\partial n(y)}\nu(y)d\Gamma(y) \qquad \text{for } x \in \Gamma. \tag{39}$$

These are known as the single-layer, double-layer, adjoint double-layer and hypersingular boundary operators, respectively. Again the sub-index n indicates the Green's function used.

2.4.3 Calderón Operator and Projector

The Calderón, or multitrace, operator is a blocked operator made up of 4 operators used in BEM formulations. We define it as

$$A_n = \begin{bmatrix} -K_n & V_n \\ \\ D_n & T_n \end{bmatrix}, \tag{40}$$

where by V_n, K_n, T_n and D_n are the single-layer, double-layer, adjoint double-layer and hypersingular boundary operators, respectively, as defined in section (2.4.2)

The Calderón projector is a pseudo-differential operator. It can be expressed

as

$$C_n^{\pm} = \begin{bmatrix} \frac{1}{2}I \pm K_n & \mp V_n \\ \mp D_n & \frac{1}{2}I \mp T_n \end{bmatrix} = \frac{1}{2}I \mp A_n. \tag{41}$$

The sign to be chosen depends on whether is is the interior $(-)$ or exterior $(+)$ problem. It can be shown that

$$(C^+)^2 = C^+ \qquad \text{and} \qquad (C^-)^2 = C^-, \tag{42}$$

also

$$A^2 = \frac{1}{4}I, \tag{43}$$

and

$$\begin{bmatrix} -K & V \\ D & T \end{bmatrix} \begin{bmatrix} -K & V \\ D & T \end{bmatrix} = \begin{bmatrix} K^2 + VD & VT - KV \\ TD - KD & T^2 + DV \end{bmatrix} = \begin{bmatrix} \frac{1}{4} & 0 \\ 0 & \frac{1}{4} \end{bmatrix}. \tag{44}$$

These identities make both the Calderón projector and operator very useful for preconditioning of some BEM systems and were tested for some of the formulations in this work.

2.4.4 Laplace Example Utilising Operators

The trace, potential and boundary integral operators can be used to simplify the notation of the BEM formulation. This will be presented with an example using the Laplace equation. Recalling equation (25) from the previous example, this can be expressed using the single layer and double layer potential integral operators, that is,

$$\int_\Gamma \frac{\partial G(x,y)}{\partial n}\phi(y)d\Gamma(y) - \int_\Gamma G(x,y)\frac{\partial \phi(y)}{\partial n}d\Gamma(y) + \phi_{(x)} = 0 = \mathcal{K}(\nu) - \mathcal{V}(\mu) + \phi(x),$$

$$\phi(x) = \mathcal{V}(\mu(\phi(x))) - \mathcal{K}(\nu(\phi(x))). \tag{45}$$

This is known as the representation formula, where the potentials ν and μ are given by the jumps across the boundary, these are,

$$\nu(y) = \gamma_D^-\phi - \gamma_D^+\phi \qquad \text{and} \qquad \mu(y) = \gamma_N^-\phi - \gamma_N^+\phi. \tag{46}$$

The boundary integral operators can be related to the traces of the potential integral operators with the following jump relations:

$$V\mu = \gamma_D^-(\mathcal{V}\mu) = \gamma_D^+(\mathcal{V}\mu), \tag{47}$$

$$K\nu = \gamma_D^-(\mathcal{K}\nu) + \frac{1}{2}\nu = \gamma_D^+(\mathcal{K}\nu) - \frac{1}{2}\nu, \tag{48}$$

$$T\mu = \gamma_N^-(\mathcal{V}\mu) - \frac{1}{2}\mu = \gamma_N^+(\mathcal{V}\mu) + \frac{1}{2}\mu, \tag{49}$$

$$D\nu = -\gamma_N^-(\mathcal{K}\nu) = -\gamma_N^+(\mathcal{K}\nu). \tag{50}$$

As can be seen not all of the traces of the potential integral operators are continuous across the boundary, that is they differ between the interior and exterior domains.

Taking the interior Dirichlet trace of the representation formula gives

$$\gamma_D^-\phi(x) = \gamma_D^-(\mathcal{V}(\mu)) - \gamma_D^-(\mathcal{K}(\nu)),$$

$$\phi(x) = V\mu - K\nu + \frac{1}{2}\nu. \tag{51}$$

As this is the case of an internal Laplace problem, the exterior potential is assumed to be zero. Given this the potentials, as defined in equation (46), are $\nu = \gamma_D^- \phi$ and $\mu = \gamma_N^- \phi$ which we can substitute this into (51), obtaining

$$\phi(x) = V\frac{\partial \phi(x)}{\partial n} - K\phi(x) + \frac{1}{2}\phi(x). \tag{52}$$

This can be extended to the exterior case, and also to include the Neumann trace, giving

$$\begin{bmatrix} \gamma_D^{\pm}\phi \\ \gamma_N^{\pm}\phi \end{bmatrix} = \begin{bmatrix} -\gamma_D^{\pm}\mathcal{K} & \gamma_D^{\pm}\mathcal{V} \\ -\gamma_N^{\pm}\mathcal{K} & \gamma_N^{\pm}\mathcal{V} \end{bmatrix} \begin{bmatrix} \nu \\ \mu \end{bmatrix}. \tag{53}$$

This is known as the Calderón system. Separating into two cases, one with the interior trace and the other the exterior trace, plus substituting in the boundary operators, we obtain

$$\begin{bmatrix} \gamma_D^{-}\phi \\ \gamma_N^{-}\phi \end{bmatrix} = \begin{bmatrix} \frac{1}{2}I - K & V \\ D & \frac{1}{2}I + T \end{bmatrix} \begin{bmatrix} \nu \\ \mu \end{bmatrix}, \tag{54}$$

$$\begin{bmatrix} \gamma_D^{+}\phi \\ \gamma_N^{+}\phi \end{bmatrix} = \begin{bmatrix} -\frac{1}{2}I - K & V \\ D & -\frac{1}{2}I + T \end{bmatrix} \begin{bmatrix} \nu \\ \mu \end{bmatrix}. \tag{55}$$

The discontinuities across the boundary, seen in equations (47 - 50), are present here in the form of the identity operator I. Utilising the Calderón operator A,

defined in section (2.4.3), it is possible to simplify the notation,

$$\begin{bmatrix} \gamma_D^{\pm}\phi \\ \\ \gamma_N^{\pm}\phi \end{bmatrix} = \left(\mp \frac{1}{2} I + A \right) \begin{bmatrix} \nu \\ \\ \mu \end{bmatrix}. \tag{56}$$

2.5 Discretisation

The continuous operators described above must first be discretised, in order for the problem to be solved numerically. The use of BEM means that only a surface mesh is required and this is generally made up of flat triangular panels.

With the space discretised the values of the variables to be calculated, in this case the potential ϕ and its normal derivative $\frac{\partial \phi}{\partial n}$, can be represented utilising constant values across each element, or continuous or discontinuous polynomials of a certain order. These are called the function spaces.

3 Methodology

The biomolecular system studied in this work was that of a single biomolecule dissolved in a saline solvent, as depicted in figure 3. This simplified system only contains two regions, that of the solute and the other containing the solvent. Because of this, notation changes can be made for simplification and easier understanding. The domain in the interior of the protein will be referred to as the interior domain $(-, int)$, and that of the solvent as the exterior domain $(+, ext)$. This is shown in the revised figure of the problem shown bellow:

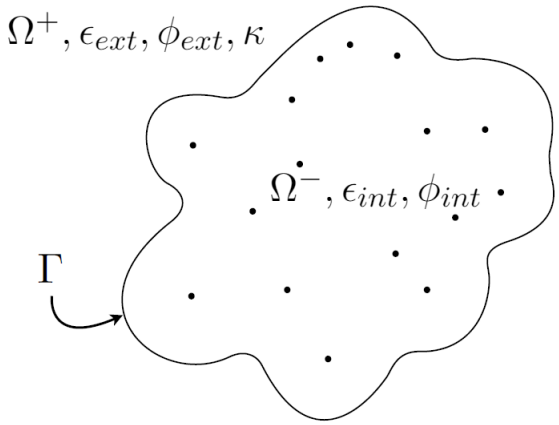


Figure 3: Depiction of single dissolved protein.

Taking the system of equations (10) from the PB implicit solvent model described in section (2.1), and making notational changes to account for the presence of only two domains results in the following:

$$\begin{cases} \nabla^2\phi_{ext}(x) - \kappa^2\phi_{ext}(x) = 0 & \text{in } \Omega^+, \\ \nabla^2\phi_{int}(x) = -\sum_k \frac{q_k}{\epsilon_{int}}\delta(x, x_k) & \text{in } \Omega^-, \\ \gamma_D^-\phi_{int}(x) = \gamma_D^+\phi_{ext}(x) & \text{on } \Gamma, \\ \gamma_N^-\phi_{int}(x) = \epsilon\gamma_N^+\phi_{ext}(x) & \text{on } \Gamma, \end{cases} \tag{57}$$

where ϵ is the ratio between the dielectric constants of the two domains $\epsilon = \frac{\epsilon_{ext}}{\epsilon_{int}}$.

3.1 Integral Formulations

To be able to apply BEM the PDEs in the system of equations (57) must first be transformed into integral equations. As shown in section (2.4.4), using Green's representation theorem, the solutions of the internal and external potentials can be written as

$$\phi_{int} = \mathcal{V}_{int}\mu_{int} - \mathcal{K}_{int}\nu_{int} + \sum_k \frac{q_k}{\epsilon_{int}}\delta(x, x_k), \tag{58}$$

$$\phi_{ext} = \mathcal{V}_{ext}\mu_{ext} - \mathcal{K}_{ext}\nu_{ext}, \tag{59}$$

where the potentials are defined as the jumps across the boundary, these are

$$\nu_{int} = \gamma_D^- \phi_{int} - \gamma_D^+ \phi_{int}, \qquad \mu_{int} = \gamma_N^- \phi_{int} - \gamma_N^+ \phi_{int},$$
$$\nu_{ext} = \gamma_D^- \phi_{ext} - \gamma_D^+ \phi_{ext}, \qquad \mu_{ext} = \gamma_N^- \phi_{ext} - \gamma_N^+ \phi_{ext},$$

And the single layer and double layer potential operators are dependant on the domain and are defined as follows:

$$[\mathcal{V}_{int}\mu](x) = \int_\Gamma G_{int}(x, y)\mu(y)d\Gamma(y) \qquad \text{for } x \in \Omega^-, \tag{60}$$

$$[\mathcal{V}_{ext}\mu](x) = \int_\Gamma G_{ext}(x, y)\mu(y)d\Gamma(y) \qquad \text{for } x \in \Omega^+, \tag{61}$$

$$[\mathcal{K}_{int}\nu](x) = \int_\Gamma \frac{\partial G_{int}(x, y)}{\partial n(y)}\nu(y)d\Gamma(y) \qquad \text{for } x \in \Omega^-, \tag{62}$$

$$[\mathcal{K}_{ext}\nu](x) = \int_\Gamma \frac{\partial G_{ext}(x, y)}{\partial n(y)}\nu(y)d\Gamma(y) \qquad \text{for } x \in \Omega^+. \tag{63}$$

G_{int} and G_{ext} are the Green's function of the internal and external domains, respectively. The Green's function for the internal domain is that of Laplace, and

for the external domain it is Modified Helmholtz or Yukawa Potential. These are

$$G_{int}(x,y) = \frac{1}{4\pi|x-y|}, \tag{64}$$

$$G_{ext}(x,y) = \frac{e^{-\kappa|x-y|}}{4\pi|x-y|}. \tag{65}$$

Defining the interior and exterior fields for the problem we have

$$\phi_{int} = \begin{cases} 0 & \text{in } \Omega^+, \\ \phi_{int} & \text{in } \Omega^-, \end{cases} \tag{66}$$

$$\phi_{ext} = \begin{cases} \phi_{ext} & \text{in } \Omega^+, \\ 0 & \text{in } \Omega^-. \end{cases} \tag{67}$$

From this we can get the value for each potential:

$$\nu_{int} = \gamma_D^- \phi_{int}, \tag{68}$$

$$\mu_{int} = \gamma_N^- \phi_{int}, \tag{69}$$

$$\nu_{ext} = -\gamma_D^+ \phi_{ext}, \tag{70}$$

$$\mu_{ext} = -\gamma_N^+ \phi_{ext}. \tag{71}$$

Taking the Dirchlet and Neumann traces of the representation formulas gives

$$\begin{bmatrix} \gamma_D^- \phi_{int} \\ \gamma_N^- \phi_{int} \end{bmatrix} = \left(\frac{1}{2} I + A_{int} \right) \begin{bmatrix} \nu_{int} \\ \mu_{int} \end{bmatrix} + \begin{bmatrix} \frac{1}{4\pi\epsilon_{int}} \sum_k \frac{q_k}{|x-x_k|} \\ \frac{\partial}{\partial n} \left(\frac{1}{4\pi\epsilon_{int}} \sum_k \frac{q_k}{|x-x_k|} \right) \end{bmatrix}, \tag{72}$$

$$\begin{bmatrix} \gamma_D^+ \phi_{ext} \\ \\ \gamma_N^+ \phi_{ext} \end{bmatrix} = \left(-\frac{1}{2} I + A_{ext} \right) \begin{bmatrix} \nu_{ext} \\ \\ \mu_{ext} \end{bmatrix}, \tag{73}$$

where A_{int} and A_{ext} are Calderón operators as defined in (40). The boundary operators that make up these Calderón operators use the corresponding Green's function for the domain, so A_{int} uses G_{int} and A_{ext} uses G_{ext}. Now, replacing the potentials ν and μ, with the identities in equations (68 - 71), and making use of the Cauchy trace ($\gamma^{\pm} = \begin{bmatrix} \gamma_D^{\pm} & \gamma_N^{\pm} \end{bmatrix}^T$) gives

$$\gamma^- \phi_{int} = \left(\frac{1}{2} I + A_{int} \right) \gamma^- \phi_{int} + \gamma^- \left(\frac{1}{4\pi\epsilon_{int}} \sum_k \frac{q_k}{|x - x_k|} \right), \tag{74}$$

$$\gamma^+ \phi_{ext} = \left(-\frac{1}{2} I + A_{ext} \right) \begin{bmatrix} -\gamma_D^+ \phi_{ext} \\ \\ -\gamma_N^+ \phi_{ext} \end{bmatrix} = \left(\frac{1}{2} I - A_{ext} \right) \gamma^+ \phi_{ext}. \tag{75}$$

$$\gamma_D^- + \epsilon \tag{76}$$

Next, the boundary conditions defined in (57) are applied, and all of the potential parameters in the equations can be left in function of only the internal potential ϕ_{int}. To do this we define $E = \begin{bmatrix} I & 0 \\ 0 & I/\epsilon \end{bmatrix}$, now the boundary conditions can be expressed as $\gamma^+ \phi_{ext} = E\, \gamma^- \phi_{int}$. Substituting into equation (75) results in

$$E\, \gamma^- \phi_{int} = \left(\frac{1}{2} I - A_{ext} \right) E\, \gamma^- \phi_{int}. \tag{77}$$

In the case that we multiply both sides by E^{-1} this simplifies to

$$\gamma^- \phi_{int} = \left(\frac{1}{2} I - \tilde{A}_{ext} \right) \gamma^- \phi_{int}, \tag{78}$$

where by $\tilde{A}_{ext} = E^{-1}\, A_{ext}\, E$, which is known as the scaled exterior Calderón

operator. We define this as

$$\tilde{A}_{ext} = \begin{bmatrix} -K_{ext} & \frac{1}{\epsilon}V_{ext} \\ \epsilon D_{ext} & T_{ext} \end{bmatrix}. \tag{79}$$

3.1.1 Direct Formulation

Following the work done by Yoon and Lenhoff [1] we obtain our first formulation for the system of equations (57). This will be referred to as the direct formulation. Taking only the Dirichlet trace part of equations (74) and (77) gives

$$\gamma_D^- \phi_{int} = \left(\frac{1}{2} - K_{int}\right)\gamma_D^- \phi_{int} + V_{int}\gamma_N^- \phi_{int} + \gamma_D^- \left(\frac{1}{4\pi\epsilon_{int}} \sum_k \frac{q_k}{|x - x_k|}\right),$$
$$\gamma_D^- \phi_{int} = \left(\frac{1}{2} + K_{ext}\right)\gamma_D^- \phi_{int} - \left(\frac{1}{\epsilon}V_{ext}\right)\gamma_N^- \phi_{int},$$

$$\left(\frac{1}{2} + K_{int}\right)\phi_{int}(x) - V_{int}\frac{\partial \phi_{int}(x)}{\partial n} = \sum_k \frac{q_k}{\epsilon_{int}}\delta(x, x_k), \tag{80}$$
$$\left(\frac{1}{2} - K_{ext}\right)\phi_{int}(x) + \frac{1}{\epsilon}V_{ext}\frac{\partial \phi_{int}(x)}{\partial n} = 0. \tag{81}$$

This can also be expressed in a matrix based notation,

$$\begin{bmatrix} \frac{1}{2}I + K_{int} & -V_{int} \\ \frac{1}{2} - K_{ext} & \frac{1}{\epsilon}V_{ext} \end{bmatrix} \begin{bmatrix} \phi_{int} \\ \frac{\partial \phi_{int}}{\partial n} \end{bmatrix} = \begin{bmatrix} \sum_k \frac{q_k}{\epsilon_{int}}\delta(x, x_k) \\ 0 \end{bmatrix}. \tag{82}$$

3.1.2 Juffer Formulation

Another well known formulation of the problem is that presented by Juffer and coworkers [2], and which is used in the TABI code developed by Geng and Krasny [7]. This formulation is well conditioned and makes use of the normal derivative of the equations of the direct method. Taking (74) and adding $\left[\begin{smallmatrix}\epsilon & 0\\ 0 & I\end{smallmatrix}\right]$ times (77) results in

$$\begin{bmatrix}\gamma_D^-\phi_{int}\\ \\ \gamma_N^-\phi_{int}\end{bmatrix} + \begin{bmatrix}\epsilon & 0\\ \\ 0 & I\end{bmatrix}\begin{bmatrix}\gamma_D^-\phi_{int}\\ \\ \frac{1}{\epsilon}\gamma_N^-\phi_{int}\end{bmatrix} = \left(\frac{1}{2}I + A_{int}\right)\begin{bmatrix}\gamma_D^-\phi_{int}\\ \\ \gamma_N^-\phi_{int}\end{bmatrix} + \gamma^-\left(\frac{1}{4\pi\epsilon_{int}}\sum_k \frac{q_k}{|x-x_k|}\right) + \begin{bmatrix}\epsilon & 0\\ \\ 0 & 1\end{bmatrix}\left(\frac{1}{2}I - A_{ext}\right)\begin{bmatrix}\gamma_D^-\phi_{int}\\ \\ \frac{1}{\epsilon}\gamma_N^-\phi_{int}\end{bmatrix}. \quad (83)$$

Splitting this into two separate equations, one containing the Dirichlet trace and the other the Neumann trace, we obtain the Juffer formulation, that is,

$$\frac{1}{2}\left(1+\epsilon\right)\phi_{int} = \left(\epsilon K_{ext} - K_{int}\right)\phi_{int} + \left(V_{int} - V_{ext}\right)\frac{\partial\phi_{int}}{\partial n} + \frac{1}{4\pi\epsilon_{int}}\sum_k \frac{q_k}{|x-x_k|}, \quad (84)$$

$$\frac{1}{2}\left(1+\frac{1}{\epsilon}\right)\frac{\partial\phi_{int}}{\partial n} = \left(D_{int} - D_{ext}\right)\phi_{int} + \left(T_{int} - \frac{1}{\epsilon}T_{ext}\right)\frac{\partial\phi_{int}}{\partial n} + \frac{\partial}{\partial n}\left(\frac{1}{4\pi\epsilon_{int}}\sum_k \frac{q_k}{|x-x_k|}\right). \quad (85)$$

We can pass this to a matrix based notation giving

$$\begin{bmatrix} \frac{1}{2}I\left(1+\epsilon\right) \\ \frac{1}{2}I\left(1+\frac{1}{\epsilon}\right) \end{bmatrix} \begin{bmatrix} \phi_{int} \\ \frac{\partial\phi_{int}}{\partial n} \end{bmatrix} = \begin{bmatrix} \epsilon K_{ext} - K_{int} & V_{int} - V_{ext} \\ D_{int} - D_{ext} & T_{int} - \frac{1}{\epsilon}T_{ext} \end{bmatrix} \begin{bmatrix} \phi_{int} \\ \frac{\partial\phi_{int}}{\partial n} \end{bmatrix} + \begin{bmatrix} \frac{1}{4\pi\epsilon_{int}} \sum_k \frac{q_k}{|x-x_k|} \\ \frac{\partial}{\partial n}\left(\frac{1}{4\pi\epsilon_{int}} \sum_k \frac{q_k}{|x-x_k|}\right) \end{bmatrix}. \quad (86)$$

Rearranging gives the $Ax = b$ form that was used in this work,

$$\begin{bmatrix} \frac{1}{2}I\left(1+\epsilon\right) - \left(\epsilon K_{ext} - K_{int}\right) & -\left(V_{int} - V_{ext}\right) \\ -\left(D_{int} - D_{ext}\right) & \frac{1}{2}I\left(1+\frac{1}{\epsilon}\right) - \left(T_{int} - \frac{1}{\epsilon}T_{ext}\right) \end{bmatrix} \begin{bmatrix} \phi_{int} \\ \frac{\partial\phi_{int}}{\partial n} \end{bmatrix} = \begin{bmatrix} \frac{1}{4\pi\epsilon_{int}} \sum_k \frac{q_k}{|x-x_k|} \\ \frac{\partial}{\partial n}\left(\frac{1}{4\pi\epsilon_{int}} \sum_k \frac{q_k}{|x-x_k|}\right) \end{bmatrix}. \quad (87)$$

3.1.3 Alpha-Beta Formulation

As with the Juffer formulation, we take equations (74) and (77) and add them. However, this time multiplying (77) by matrix $D = \left[\begin{smallmatrix} \alpha & 0 \\ 0 & \beta \end{smallmatrix}\right]$, this gives

$$\left(I + DE\right)\gamma^{-}\phi_{int} = \left(\left(\frac{1}{2}I + A_{int}\right) + D\left(\frac{1}{2}I - A_{ext}\right)E\right)\gamma^{-}\phi_{int} + \gamma^{-}\left(\frac{1}{4\pi\epsilon_{int}} \sum_k \frac{q_k}{|x-x_k|}\right). \quad (88)$$

Taking α and β to be real numbers, this becomes a series of linear combinations of the interior and exterior representation formulas. Restructuring (88) we obtain

the final linear system that must be solved as

$$\left(\left(\frac{1}{2}I + A_{int}\right) + D\left(\frac{1}{2}I - A_{ext}\right)E - (I+F)\right)\begin{bmatrix}\phi_{int}\\ \\ \frac{\partial\phi_{int}}{\partial n}\end{bmatrix} = -\begin{bmatrix}\frac{1}{4\pi\epsilon_{int}}\sum_k \frac{q_k}{|x-x_k|}\\ \\ \frac{\partial}{\partial n}\left(\frac{1}{4\pi\epsilon_{int}}\sum_k \frac{q_k}{|x-x_k|}\right)\end{bmatrix}. \quad (89)$$

Where D, E and F are defined as

$$D = \begin{bmatrix}\alpha & 0\\ \\ 0 & \beta\end{bmatrix}, \qquad E = \begin{bmatrix}I & 0\\ \\ 0 & \frac{I}{\epsilon}\end{bmatrix}, \qquad F = DE = \begin{bmatrix}\alpha & 0\\ \\ 0 & \frac{\beta}{\epsilon}\end{bmatrix}. \quad (90)$$

This can also be expressed as

$$\left(\frac{1}{2}I + A_{int} + \left(D\cdot\frac{1}{2}I\cdot E\right) - (D\cdot A_{ext}\cdot E) - I - F\right)\begin{bmatrix}\phi_{int}\\ \\ \frac{\partial\phi_{int}}{\partial n}\end{bmatrix} = -\begin{bmatrix}\frac{1}{4\pi\epsilon_{int}}\sum_k \frac{q_k}{|x-x_k|}\\ \\ \frac{\partial}{\partial n}\left(\frac{1}{4\pi\epsilon_{int}}\sum_k \frac{q_k}{|x-x_k|}\right)\end{bmatrix},$$

$$\left(\frac{1}{2}I + \begin{bmatrix} -K_{int} & V_{int} \\ D_{int} & T_{int} \end{bmatrix} + \frac{1}{2}I \begin{bmatrix} \alpha & 0 \\ 0 & \frac{\beta}{\epsilon} \end{bmatrix} - \begin{bmatrix} -\alpha K_{ext} & \frac{\alpha V_{ext}}{\epsilon} \\ \beta D_{ext} & \frac{\beta T_{ext}}{\epsilon} \end{bmatrix} - I - \begin{bmatrix} \alpha & 0 \\ 0 & \frac{\beta}{\epsilon} \end{bmatrix}\right) \begin{bmatrix} \phi_{int} \\ \frac{\partial \phi_{int}}{\partial n} \end{bmatrix} = - \begin{bmatrix} \frac{1}{4\pi\epsilon_{int}} \sum_k \frac{q_k}{|x-x_k|} \\ \frac{\partial}{\partial n}\left(\frac{1}{4\pi\epsilon_{int}} \sum_k \frac{q_k}{|x-x_k|}\right) \end{bmatrix},$$

$$\left(\begin{bmatrix} -K_{int} + \alpha K_{ext} & V_{int} - \frac{\alpha V_{ext}}{\epsilon} \\ D_{int} - \beta D_{ext} & T_{int} - \frac{\beta T_{ext}}{\epsilon} \end{bmatrix} - \frac{1}{2}I \begin{bmatrix} 1+\alpha & 0 \\ 0 & 1+\frac{\beta}{\epsilon} \end{bmatrix}\right) \begin{bmatrix} \phi_{int} \\ \frac{\partial \phi_{int}}{\partial n} \end{bmatrix} = - \begin{bmatrix} \frac{1}{4\pi\epsilon_{int}} \sum_k \frac{q_k}{|x-x_k|} \\ \frac{\partial}{\partial n}\left(\frac{1}{4\pi\epsilon_{int}} \sum_k \frac{q_k}{|x-x_k|}\right) \end{bmatrix}. \quad (91)$$

It is apparent that this is a generalisation of the Juffer formulation, which is a sub-case when $\alpha = \epsilon$ and $\beta = 1$. This is shown by substituting the values into equation (89), giving

$$\left(\left(\frac{1}{2}I + A_{int}\right) + \begin{bmatrix} \epsilon & 0 \\ 0 & I \end{bmatrix}\left(\frac{1}{2}I - A_{ext}\right) \begin{bmatrix} I & 0 \\ 0 & \frac{I}{\epsilon} \end{bmatrix} - I - \begin{bmatrix} \epsilon & 0 \\ 0 & \frac{I}{\epsilon} \end{bmatrix}\right) \begin{bmatrix} \phi_{int} \\ \frac{\partial \phi_{int}}{\partial n} \end{bmatrix} = - \begin{bmatrix} \frac{1}{4\pi\epsilon_{int}} \sum_k \frac{q_k}{|x-x_k|} \\ \frac{\partial}{\partial n}\left(\frac{1}{4\pi\epsilon_{int}} \sum_k \frac{q_k}{|x-x_k|}\right) \end{bmatrix},$$

$$\left(A_{int} - \begin{bmatrix} -\epsilon K_{ext} & V_{ext} \\ D_{ext} & \frac{1}{\epsilon} T_{ext} \end{bmatrix} - \frac{1}{2} I - \begin{bmatrix} \frac{\epsilon}{2} & 0 \\ 0 & \frac{I}{2\epsilon} \end{bmatrix} \right) \begin{bmatrix} \phi_{int} \\ \frac{\partial \phi_{int}}{\partial n} \end{bmatrix}$$

$$= - \begin{bmatrix} \frac{1}{4\pi\epsilon_{int}} \sum_k \frac{q_k}{|x - x_k|} \\ \frac{\partial}{\partial n} \left(\frac{1}{4\pi\epsilon_{int}} \sum_k \frac{q_k}{|x - x_k|} \right) \end{bmatrix},$$

$$\left(\begin{bmatrix} -K_{int} + \epsilon K_{ext} & V_{int} - V_{ext} \\ D_{int} - D_{ext} & T_{int} - \frac{1}{\epsilon} T_{ext} \end{bmatrix} - \frac{1}{2} I \begin{bmatrix} 1 + \epsilon & 0 \\ 0 & 1 + \frac{1}{\epsilon} \end{bmatrix} \right) \begin{bmatrix} \phi_{int} \\ \frac{\partial \phi_{int}}{\partial n} \end{bmatrix}$$

$$= - \begin{bmatrix} \frac{1}{4\pi\epsilon_{int}} \sum_k \frac{q_k}{|x - x_k|} \\ \frac{\partial}{\partial n} \left(\frac{1}{4\pi\epsilon_{int}} \sum_k \frac{q_k}{|x - x_k|} \right) \end{bmatrix}. \quad (92)$$

This is equivalent to the Juffer system shown in equation (87), and provides an easy method to try different combinations of the representation formulas, simply by changing the values of α and β.

In this work, along with $\alpha = \epsilon$ and $\beta = 1$ (Juffer formulation), three other cases of α and β were assessed to see how they compare in terms of their convergence and use of computational resources. The first case was selected as a case presenting perhaps the simplest values of α and β, both being 1. The other two cases were selected because of the form these present, comprising of the addition and subtraction of Calderón operators and the identity matrix, appearing similar to formulations used in acoustics and electromagnetism. Here follows a simplified notation of these three cases.

Case 1: ($\alpha = 1$ and $\beta = 1$)

$$\left(\left(\frac{1}{2}I + A_{int}\right) + \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix} \left(\frac{1}{2}I - A_{ext}\right) \begin{bmatrix} I & 0 \\ 0 & \frac{I}{\epsilon} \end{bmatrix} - I - \begin{bmatrix} I & 0 \\ 0 & \frac{I}{\epsilon} \end{bmatrix} \right) \begin{bmatrix} \phi_{int} \\ \frac{\partial \phi_{int}}{\partial n} \end{bmatrix}$$
$$= - \begin{bmatrix} \frac{1}{4\pi\epsilon_{int}} \sum_k \frac{q_k}{|x - x_k|} \\ \frac{\partial}{\partial n} \left(\frac{1}{4\pi\epsilon_{int}} \sum_k \frac{q_k}{|x - x_k|} \right) \end{bmatrix}$$

$$\left(A_{int} - \begin{bmatrix} -K_{ext} & \frac{1}{\epsilon}V_{ext} \\ D_{ext} & \frac{1}{\epsilon}T_{ext} \end{bmatrix} - \frac{1}{2}I \begin{bmatrix} 2 & 0 \\ 0 & 1 + \frac{1}{\epsilon} \end{bmatrix} \right) \begin{bmatrix} \phi_{int} \\ \frac{\partial \phi_{int}}{\partial n} \end{bmatrix}$$
$$= - \begin{bmatrix} \frac{1}{4\pi\epsilon_{int}} \sum_k \frac{q_k}{|x - x_k|} \\ \frac{\partial}{\partial n} \left(\frac{1}{4\pi\epsilon_{int}} \sum_k \frac{q_k}{|x - x_k|} \right) \end{bmatrix} \quad (93)$$

Case 2: ($\alpha = 1$ and $\beta = \epsilon$)

$$\left(\left(\frac{1}{2}I + A_{int}\right) + \begin{bmatrix} I & 0 \\ 0 & \epsilon \end{bmatrix} \left(\frac{1}{2}I - A_{ext}\right) \begin{bmatrix} I & 0 \\ 0 & \frac{I}{\epsilon} \end{bmatrix} - I - \begin{bmatrix} I & 0 \\ 0 & \frac{\epsilon}{\epsilon} \end{bmatrix} \right) \begin{bmatrix} \phi_{int} \\ \frac{\partial \phi_{int}}{\partial n} \end{bmatrix}$$
$$= - \begin{bmatrix} \frac{1}{4\pi\epsilon_{int}} \sum_k \frac{q_k}{|x - x_k|} \\ \frac{\partial}{\partial n} \left(\frac{1}{4\pi\epsilon_{int}} \sum_k \frac{q_k}{|x - x_k|} \right) \end{bmatrix}$$

$$\left(A_{int} - \tilde{A}_{ext} - I\right) \begin{bmatrix} \phi_{int} \\ \\ \frac{\partial \phi_{int}}{\partial n} \end{bmatrix} = - \begin{bmatrix} \frac{1}{4\pi\epsilon_{int}} \sum_k \frac{q_k}{|x - x_k|} \\ \\ \frac{\partial}{\partial n} \left(\frac{1}{4\pi\epsilon_{int}} \sum_k \frac{q_k}{|x - x_k|} \right) \end{bmatrix} \tag{94}$$

Where $\tilde{A}_{ext} = E^{-1}\, A_{ext}\, E$ is the scaled exterior Calderón operator, as defined in equation (79).

Case 3: ($\alpha = -1$ and $\beta = -\epsilon$)

$$\left(\left(\frac{1}{2} I + A_{int}\right) + \begin{bmatrix} -I & 0 \\ \\ 0 & -\epsilon \end{bmatrix} \left(\frac{1}{2} I - A_{ext}\right) \begin{bmatrix} I & 0 \\ \\ 0 & \frac{I}{\epsilon} \end{bmatrix} - I - \begin{bmatrix} -I & 0 \\ \\ 0 & \frac{-\epsilon}{\epsilon} \end{bmatrix} \right) \begin{bmatrix} \phi_{int} \\ \\ \frac{\partial \phi_{int}}{\partial n} \end{bmatrix}$$

$$= - \begin{bmatrix} \frac{1}{4\pi\epsilon_{int}} \sum_k \frac{q_k}{|x - x_k|} \\ \\ \frac{\partial}{\partial n} \left(\frac{1}{4\pi\epsilon_{int}} \sum_k \frac{q_k}{|x - x_k|} \right) \end{bmatrix}$$

$$\left(A_{int} + \tilde{A}_{ext}\right) \begin{bmatrix} \phi_{int} \\ \\ \frac{\partial \phi_{int}}{\partial n} \end{bmatrix} = - \begin{bmatrix} \frac{1}{4\pi\epsilon_{int}} \sum_k \frac{q_k}{|x - x_k|} \\ \\ \frac{\partial}{\partial n} \left(\frac{1}{4\pi\epsilon_{int}} \sum_k \frac{q_k}{|x - x_k|} \right) \end{bmatrix} \tag{95}$$

3.2 Calculation of Solvation Energy

To calculate the solvation energy of the biomolecule we must first calculate the reaction potential as shown in equation (12). This can be done by taking equation (58) and subtracting the Coulomb effect, giving

$$\phi_{reac} = \mathcal{V}_{int} \frac{\partial \phi_{int}}{\partial n} - \mathcal{K}_{int} \phi_{int}. \tag{96}$$

With this calculation of the solvation energy simply requires the evaluation of

equation (11).

3.3 Analytical solution

Generally the geometry of biomolecules is not regular and so the electrostatic potential can not be calculated by analytical means. However, with simpler forms it is possible to obtain an analytical solution. For the case of a sphere with radius R and single point charge at its centre, shown in figure (4), the solution can be easily calculated. Based on the analytical solution presented by Kirkwood [8] the following is obtained.

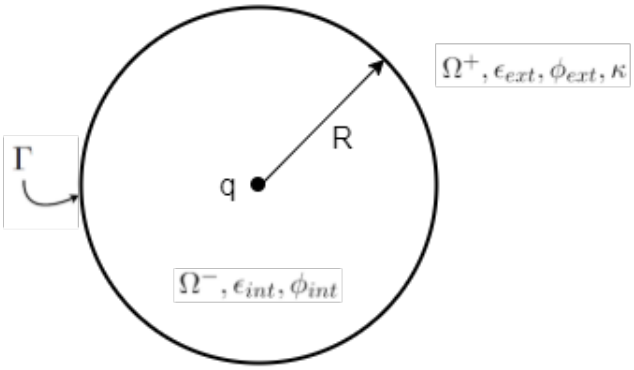


Figure 4: Representation of problem with simplified geometry

Using an expansion with Legendre Polynomials (P_n) the internal and external potentials can be expressed as

$$\phi_{int}(r) = \frac{q}{4\pi\epsilon_{int}r} + \sum_{n=0}^{\infty} C_n r^n P_n(cos(\theta)), \tag{97}$$

$$\phi_{ext}(r) = \sum_{n=0}^{\infty} a_n k_n(\kappa r) P_n(cos(\theta)). \tag{98}$$

Given that there is only single point charge at the centre of the sphere, the only term that remains is $n = 0$. It is also known that $P_0(x) = 1$ so the expressions

become

$$\phi_{int}(r) = \frac{q}{4\pi\epsilon_{int}r} + C_0, \tag{99}$$

$$\phi_{ext}(r) = a_0 k_0(\kappa r). \tag{100}$$

k_n is the modified spherical Bessel function of the second kind and for $n = 0$ results in

$$k_0(x) = \frac{e^{-x}}{x}, \tag{101}$$

$$\frac{dk_0(x)}{dx} = -\frac{e^{-x}}{x} - \frac{e^{-x}}{x^2} = -\frac{e^{-x}}{x}\left(1 + \frac{1}{x}\right). \tag{102}$$

On the boundary ($r = R$) are the boundary conditions stated in (57). Applying these we obtain

$$\frac{q}{4\pi\epsilon_{int}R} + C_0 = a_0 k_0(\kappa r), \tag{103}$$

$$-\epsilon_{int}\frac{q}{4\pi\epsilon_{int}R^2} = \epsilon_{ext}a_0\kappa k_0'(\kappa r). \tag{104}$$

Using (104) the expression of a_0 can be determined to be

$$-\frac{q}{4\pi R^2} = -\epsilon_{ext}a_0\kappa\frac{e^{-\kappa R}}{\kappa R}\left(1 + \frac{1}{\kappa R}\right), \tag{105}$$

$$a_0 = \frac{q\kappa}{4\epsilon_{ext}\pi e^{-\kappa R}\left(1 + \kappa R\right)}. \tag{106}$$

Substituting back into equation (100) we obtain the expression for the external

potential,

$$\phi_{ext} = a_0 k_0(\kappa r) = \frac{q\kappa}{4\epsilon_{ext}\pi e^{-\kappa R}\left(1+\kappa R\right)}\frac{e^{-\kappa r}}{\kappa r} = \frac{q}{4\epsilon_{ext}\pi\left(1+\kappa R\right)}\frac{e^{-\kappa(r-R)}}{r}. \quad (107)$$

Taking the potential on the boundary ($r = R$) gives

$$\phi_{ext} = \frac{q}{4\pi\epsilon_{ext}\left(1+\kappa R\right)}\frac{e^{-\kappa(R-R)}}{R} = \frac{q}{4\pi\epsilon_{ext}\left(1+\kappa R\right)R}. \quad (108)$$

Given that the internal in external potentials are the same on the boundary, ϕ_{ext} can be used to calculate the reaction potential as shown in (12). The Coulomb potential in this case can be calculated as

$$\phi_{Coul} = \frac{1}{4\pi\epsilon_{int}}\frac{q}{R}. \quad (109)$$

The reaction potential thus becomes

$$\phi_{reac} = \phi_{ext} - \phi_{Coul} = \frac{q}{4\pi\epsilon_{ext}\left(1+\kappa R\right)R} - \frac{1}{4\pi\epsilon_{int}}\frac{q}{R}. \quad (110)$$

And finally, to calculate the solvation energy in $kcal/mol$,

$$E_{sol} = 2\pi \times 332.064 \times q\phi_{reac}. \quad (111)$$

3.4 Preconditioning Methods

The different formulations shown in this work all result in the need to solve a system of linear equations. In this case the iterative algorithm GMRES was used to resolve the system. The linear system can be expressed as $A\mathbf{x} = \mathbf{b}$, and the number of iterations required to solve the system is linked to the condition number of the matrix A. In the case of the direct formulation (3.1.1), it is known that the condition number increases in function of the number of elements in the mesh

[9]. To combat this problem, and lower the number of iterations required for convergence of other formulations, preconditioning techniques can be used.

A preconditioner is an operator, which when applied correctly is capable of lowering the condition number of the matrix A, thus the system can be resolved in less iterations. A very common type of preconditioning is the left preconditioner, which was used in this work. In this case the preconditioner takes the form of a matrix $\mathbb{P}$ and is multiplied from the left on both sides of the linear system. This results in

$$\mathbb{P}A\mathbf{x} = \mathbb{P}\mathbf{b}. \tag{112}$$

Two types of preconditoner $\mathbb{P}$ were tested in this work, they are described bellow.

3.4.1 Block-diagonal preconditioning

One of the preconditioners tested in this work was the block-diagonal, as described by Altman and coworkers [10]. As pointed out in their work, a good value for $\mathbb{P}$ is that $\mathbb{P} \approx A^{-1}$. They also state that the dominant entries of A tend to be the self influence terms. From this, they proposed that $\mathbb{P}$ be the inverse of a sparse matrix consisting of only the self terms. This is given the name block-diagonal preconditioner as it takes the form of a 2×2 block structure, whereby each block is a diagonal matrix. This preconditioner was tested on all of the presented formulations to see if there was an improvement on the number of iterations to convergence.

All of the formulations presented in this work result in a 2×2 block matrix

A.

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \tag{113}$$

Taking only the diagonal of each block we define M as

$$M = \begin{bmatrix} diag(A_{11}) & diag(A_{12}) \\ diag(A_{21}) & diag(A_{22}) \end{bmatrix}. \tag{114}$$

Taking the inverse of M we obtain our preconditioning matrix $\mathbb{P}$, that is,

$$\mathbb{P} = M^{-1}. \tag{115}$$

Given that M is a diagonal matrix, calculating it's inverse can be done easily via LU factorisation and computationally is relatively cheap.

3.4.2 Calderon preconditioning

In the case of the formulations which can be expressed with use of the Calderón operator and projector (Juffer and Alpha-Beta), a method of preconditioning can be applied arising from the identities in equations (42) and (43). This preconditioning is performed by multiplying the operators before discretising them, and so receives the name of operator preconditioning. Four different schemes based on the different Calderón operators and projectors were tested, these were as follows.

First we have the entire matrix A, which as the linear combination of the internal and external Calderón operators conserves the identifies. So the final system to be solved in this case has the following appearance:

$$\mathbb{P} = A \qquad \Longrightarrow \qquad \mathbb{P}A\mathbf{x} = \mathbb{P}\mathbf{b} \qquad \Longrightarrow \qquad A^2\mathbf{x} = A\mathbf{b}. \tag{116}$$

Two other options are to apply either the interior or exterior Calderón operators (A_{int} or A_{ext}) instead of the entire matrix A. This leads to

$$A_{int}A\mathbf{x} = A_{int}\mathbf{b}, \tag{117}$$

$$A_{ext}A\mathbf{x} = A_{ext}\mathbf{b}. \tag{118}$$

Lastly, the Calderón projector as defined in equation (41) and it's properties can make it useful as a preconditioner. In this work a variation of the Calderón projector was tested for two of the Alpha-Beta formulations presented in (3.1.3), these were for cases 2 and 3 when $\alpha = 1$, $\beta = \epsilon$ and $\alpha = -1$, $\beta = -\epsilon$, respectfully. In the general formulation, shown in equation (89), appears the term $D\left(\frac{1}{2}I - A_{ext}\right)E$. In the cases mentioned, matrix D is either the inverse or minus the inverse of matrix E, and in this context the term $E^{-1}\left(\frac{1}{2}I - A_{ext}\right)E$ is known as the scaled exterior projector. Applying this preconditioner gives

$$\left[D\left(\frac{1}{2}I - A_{ext}\right)E\right]A\mathbf{x} = \left[D\left(\frac{1}{2}I - A_{ext}\right)E\right]\mathbf{b}. \tag{119}$$

3.5 Bempp Library

Bempp [3] is an open-source computational boundary element library for solving electrostatic, acoustic and electromagnetic problems. The library has passed through various versions and rewrites over the years. The current version (bempp-cl) is written almost entirely in python, with the use of PyOpenCL for just-in-time calculation, and it also supports CPU and GPU parallelisation. The library uses operator representation, making implementation of different formulations easy. It

also supports both dense and fast multipole method (FMM) operator assembly, giving the ability to use one or the other depending on the problem to be calculated. The dense assembler is directly implemented in the bempp library, where as the FMM assembler utilises the ExaFMM [11] python library.

The previous version of the bempp (3.3.4) was written in a mixture of Python and C++, and also included hierarchical matrix (H-matrix) techniques for the assembly of operators. This is currently not available in bempp-cl, however there are plans for it to be passed into a feature release.

3.5.1 Notes on Discretisation

Bempp allows the use of a variety of different types of function spaces, for use depending on the problem to be simulated. In the case of this work, the functions spaces of both the potential (ϕ) and its normal derivative ($\frac{\partial \phi}{\partial n}$) were set to be continuous polynomials of order 1.

The bempp library uses Galerkin discretisation. This converts the continues boundary integral problem into a discrete problem, allowing numerical calculations to be performed. Due to this, the operators can be discretised into either the weak or strong form.

Defining the discrete weak form of the operator A as A, the discrete strong form (A^S) is given by

$$A^S = M_A^{-1} A. \tag{120}$$

Here M_A is the mass matrix of the operator A, which is the discrete map from the range space to the dual space of A. Applying this to the linear system $A\mathbf{x} = \mathbf{b}$, the corresponding strong form would be

$$M^{-1} A\mathbf{x} = M^{-1}\mathbf{b}. \tag{121}$$

This is sometimes referred to as mass matrix preconditioning and has been

shown to improve convergence when using the PMCHWT formulation for electromagnetic scattering [12]. The different formulations presented in this work were tested using both the weak and strong forms. For more information on the discretisation used, and the discrete operator algebra implemented in bempp, please refer to [13] and [12].

3.6 Bem_electrostatics Python Library

A python library was written to reduce the lines of code needed to perform a calculation with the methods outlined in this work, obtaining as a result the surface potential and solvation energy. The library automates almost all the steps in section (3.7), preprocessing the data needed for the calculations. It also performs all the steps outlined in section (3.8) to calculate the results. The user only needs to write a few lines of code and supply either a PDB or PQR file. Different options are available for debugging, timing different formulations, or extracting data for other purposes. The library brings together features of many different software packages (bempp-cl, NanoShaper, pdb2pqr, etc.) to perform the calculations in a way that is far more user friendly and streamline. It allows for easy selection of the formulation to be used, changing of options such as the dielectric constants and GMRES tolerance, and the output of timings of the different steps. The library is open source and can be found on GitHub [14].

3.7 Structure Preparation

Before the calculations can be performed data must be obtained about the molecule's structure and then processed, finally, producing the surface mesh to be used. These steps are described in more detail bellow.

3.7.1 Protein Data Bank

Information about the structure of the biomolecule to be used as the solute must first be obtained. The crystal structures of many folded molecules are available for free from the Protein Data Bank, an online information portal and data archive, which provides access to 3D structure data for large biological molecules. The information for the structures is obtained through methods such as X-ray crystallography, NMR Spectroscopy and cryo-electron microscopy. These methods help determine the type of atoms that form the molecules, and their positions relative to one another. The information can be downloaded in the PDB format, which contains a list of the molecule's atoms and their positions, as well as other information about the molecule and the conditions under which the data was gathered.

3.7.2 PDB2PQR and Force Fields

With the PDB information, next a force fields model must be applied to calculate the van der Waals radii and charge at the location of each atom. During this step, checks are performed to make sure that all of the atoms which would be expected from a physical point of view are present, as the methods used for producing the molecule structure data do not always capture all of the smaller atoms. This was done utilising the free pdb2pqr [15] software. The final data obtained after applying the force fields model is the position, van der Waals radii and charge of each of the atoms in the molecule. This information is stored in a PQR file.

3.7.3 Generation of Surface Mesh

From the data in the PQR file the surface mesh of the molecule can be constructed. The definition of the surface interface used in this work was the solvent excluded surface (SES), as described in the implicit solvent model (2.1). To generate surface meshes there exist various different freely available software packages, including

MSMS [16] and NanoShaper [17]. All of the calculations in this work were made using meshes generated with NanoShaper.

3.8 Calculation Code Details

The calculations presented in this work were performed utilising the bempp-cl library, with the bem_electrostatics library hiding much of the required code from the user, providing an easier to use interface for this application. Here is briefly detailed the processes used to perform the calculations with the different formulations mentioned previously. For more detailed examples on how to use the library, and information on all of the different parameters available, please refer to the page on GitHub [14].

3.8.1 Protein Object Initialisation

The first step in performing a calculation, utilising the bem_electrostatics library, consists in the creation of a solute object. This holds all the information about the solute, the parameters used and the results obtained. During the initialisation of this solute object, the required preparation steps (3.7) are performed depending on the parameters given. Either a PDB or PQR file must be specified, but other optional parameters can be specified, these include, mesh density, whether to save files generated during mesh generation, use of a pregenerated mesh, the force field model to be used, amongst others. The surface mesh is imported into the bempp-cl library and the x, y, z positions and charge of each point atom are saved to the solute object. Here follows a simple example, creating the object named `protein`.

```
import bem_electrostatics
protein = bem_electrostatics.solute('1bpi.pdb',
                                    mesh_density = 8,
                                    mesh_probe_radius = 1.4,
```

```
                                    mesh_generator = 'nanoshaper',
                                    print_times = True,
                                    save_mesh_build_files = True,
                                    force_field = 'parse')
```

Next, the necessary changes can be made to the parameters used in the calculations. In the PB implict-solvent model (2.1) there are 3 parameters that one may wish to alter, as these describe physical properties of the solvent and solute. They are the dielectric constants and the inverse Debye length. These have default values set in the bem_electrostatics library but can be changed at this stage. Other parameters must also be set depending on the formulation and preconditioning scheme to be used. The following list shows these parameters and their default values.

```
# The formulation to be used is set with the following attribute:
protein.pb_formulation = "direct"

# Attributes for the internal and external dielectric constants
# and inverse Debye length (kappa)
protein.ep_in = 4.0
protein.ep_ex = 80.0
protein.kappa = 0.125

# Attributes for the aplha and beta values to be used in the case
# that this formulation is selected
protein.pb_formulation_alpha = 1.0
protein.pb_formulation_beta = protein.ep_ex/protein.ep_in
```

```
# Whether to apply or not calderon preconditioning (only if using
# alpha_beta), and which to apply (squared, interior or exterior)
protein.pb_formulation_preconditioning = False
protein.pb_formulation_preconditioning_type = "calderon_squared"

# Select which discrete form to use (strong or weak)
protein.discrete_form_type = "strong"

# Attributes that can be changed for the GMRES solver
protein.gmres_tolerance = 1e-5
protein.gmres_max_iterations = 1000)
```

3.8.2 Calculation of the Surface Potential

When all of the different parameters have been set to the desired values, the process of calculating the surface potential can be started. This is done by calling the solute object's `calculate_potential()` function, for example,

`protein.calculate_potential()`.

This function performs the following steps to calculate the potential on the boundary, utilising the formulations presented previously.

In the first step the function spaces for the problem are created. These represent the type of approximation to be used in the discretisation of the continuous space. In bempp-cl many types are implemented, both scalar and vector spaces, which can be used depending on the problem to be solved. In the case of this work both the Dirichlet and Neumann spaces were discretised using a continuous polynomial of first order.

Next, the necessary boundary operators are generated depending on the formulation, and with these the matrix of the left hand side is initialised. This is

followed by the creation of the vectors for the right hand side of the equation. These vectors are generated through use of grid functions, which are used to represent a function on the mesh.

Now the desired preconditioning scheme is applied. In the case that this be Calderón operator preconditioning, as mentioned in section (3.4.2), the preconditioner is created and applied to the system in it's operator form. However, if the block-diagonal preconditioner is desired this is created but saved and passed as an argument to the GMRES solver later on.

Now the system of operators must be discretised. As mentioned above, bempp-cl uses Galerkin discretisation, meaning either the weak or strong form may be used depending on the situation, this is set via the `discrete_form_type` parameter of the solute object. The final matrix A and rhs vector, with preconditioning applied if desired, are passed to the discrete form set in the parameter.

Next the Scipy python library's GMRES function is used to resolve the system of equations iteratively. The variables passed to the function are, the discretised matrix and RHS vector, the desired tolerance, maximum number of iterations to perform and, in the case that block-diagonal preconditioning is used, the block-diagonal matrix. The default tolerance, which was used for all calculations in this work is $1e-5$. The maximum number of iterations was set to 200 and the default restart of 20 iterations was left unchanged.

Once the result has been computed, the solution is split and the Dirichlet and Neumann parts extracted. This data, along with the number of iterations, is saved to the solute object.

3.8.3 Calculation of Solvation Energy

In this work the solvation energy was used to compare the convergence of the different formulations. This parameter, which is also of much interest for biomolecular systems, can be calculated from the surface potential, as described in section (3.2). Here follows the steps that the bem_electrostatics library performs to

calculate the energy. In this case, the corresponding solute object function is `calculate_solvation_energy()`.

First the library checks to see if the surface potential for the solute object has been calculated, if not it first calls the `calculate_potential()` function. Now that the solute object will contain the information used and result of the calculation of the surface potential, equation (11) can be applied.

Using the same function spaces as the calculation of the surface potential, the appropriate potential operators are generated with the required evaluation points at the locations of the point charges. Utilising these operators and the solution of the surface potential, equation (96) is evaluated to compute the reaction potential at the location of the point charges. Knowing the reaction potential, equation (11) is applied, giving as a result the solvation energy. Finally, this is converted to the units $\frac{kcal}{mol}$ and saved to the solute object.

3.9 Richardson Extrapolation

In numerical computing Richardson extrapolation can be used to estimate the exact solution to a problem. This is done using a series of computations, in which the resolution of the mesh used is refined. In the case of this work Richardson extrapolation was used to analyse the order of convergence of each method used and determine and approximate exact solution. To use this method various calculations must be made with meshes of increasing fineness, and with these results the order of convergence can be calculated. The order of convergence (p), can be calculated with

$$p = \frac{log\left(\frac{f_3 - f_2}{f_2 - f_1}\right)}{log(r)}. \tag{122}$$

Here r is the mesh refinement ratio, this is calculated as $\frac{h_2}{h_1} = \frac{h_3}{h_2}$, where h may be in spacing, area or volume. f_1, f_2 and f_3 are the results of the computations for each mesh resolution. If the calculated order of convergence matches the

expected value, one can say that the numerical calculations for f_1, f_2 and f_3 are in the asymptotic range and thus it is possible to estimate the exact solution, with the following equation,

$$f_{exact} \approx f_1 + \frac{f_1 - f_2}{r^p - 1}. \tag{123}$$

4 Results and Discussion

Unless otherwise stipulated, the results bellow were calculated using the following hardware, software versions and parameters. The calculations were performed on a computational workstation equipped with two Intel Xeon E5-2680 v3 CPUs @ 2.50GHz (total of 24 physical cores and 48 threads) and 94 GB of RAM. The values used for the interior and exterior dielectric constants and inverse Debye length were $\epsilon_{int} = 4$, $\epsilon_{int} = 80$ and $\kappa = 0.125$, respectively. While the GMRES tolerance was set to 10^{-5}.

The bem_electrostatics library was used to simplify the code required in this particular application of BEM. The main code for the BEM calculations, called from bem_electrostatics, was bempp-cl version 0.2.2. Given the number of elements in the meshes utilised and the RAM resources of the workstation, it was necessary to use Fast Multiple Method (FMM) evaluation, which is done by bempp-cl using the ExaFMM library. The ExaFMM version was 0.1.0. For the generation of surface meshes for the different biomolecules, NanoShaper [17] version 0.7.8 was utilised. As explained in section (3.7.2), pdb2pqr was used to calculate the Van der Waals radii, with the PARSE force field option.

The ExaFMM library utilises two parameters to create the FMM structure which can be modified by the user, these are the ncrit and the expansion order. The best values for these parameters depends on the hardware used and the desired level of precision. To decide what values to use for these parameters a set of calculations was performed using one biomolecule and changing the values of the ncrit and expansion order. The results were then studied and the best values identified. For this case they were determined to be an $ncrit = 100$ and an $expansion\_order = 5$.

4.1 Mesh Convergence

First it was tested to see that when the mesh was refined, the different formulations converged in the expected manner. To do this, meshes of varying densities were produced for both the 1BPI biomolecule and a simple sphere of fixed radius. The solvation energy was then calculated using the 4 main formulations tested during this work. The calculations were performed using the strong form discretisation (mass matrix preconditioning) and no extra preconditioning methods.

Below are the results for the calculations using the sphere. The radius was of 20[Å] and it contained a single point charge at its centre, with a value of $10[e^{-}]$. The other parameters were kept with the values presented at the beginning of the results section.

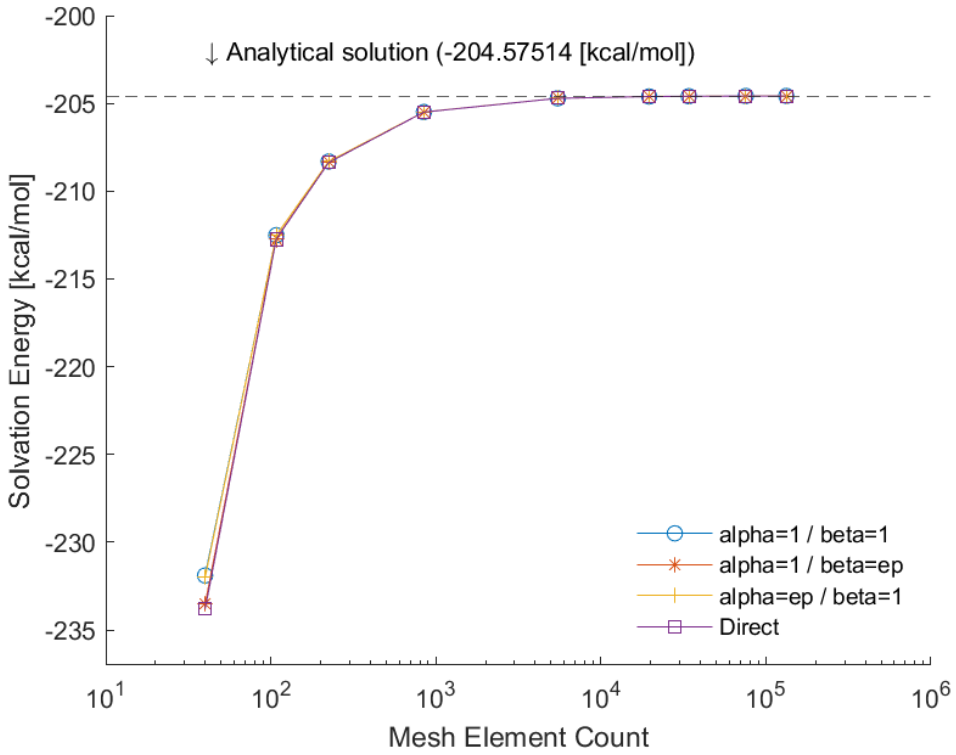


Figure 5: Solvation energy convergence with mesh refinement for spherical mesh.

For this simplified problem, with a spherical mesh and single point charge,

the solution can be computed analytically, as shown in section 3.3. The result for this analytical solution of the solvation energy, with the parameters defined here, gave -204.57514 [kcal/mol] and is plotted with the other results. From the results it can be seen that there is very low variability between the solutions of the different formulations, converging at the same rate and to very similar values. The difference between the numerical and analytical solutions is within less than 0.5 % from 846 mesh elements onwards.

Next, the same procedure was performed using the biomolecule 1BPI, for which the results are presented in figure (6). Richardson extrapolation (Sec. 3.9) was used to obtain an estimated exact solution of the solvation energy. This was done for each of the 4 different formulations and an average of these was plotted with a dotted line. The individual results of the extrapolation are shown in table (1).

Formulation used	Refinement ratio r	Observed order of convergence p	Extrapolated exact solvation energy
alpha=1 / beta=1	0.5	-0.985015	-357.0760127
alpha=1 / beta=ep	0.5	-0.997566	-357.0924383
alpha=ep / beta=1	0.5	-0.990138	-357.0820445
Direct	0.5	-0.998473	-357.0927437
		Average estimated exact solution	-357.0858098

Table 1: Richardson extrapolation results.

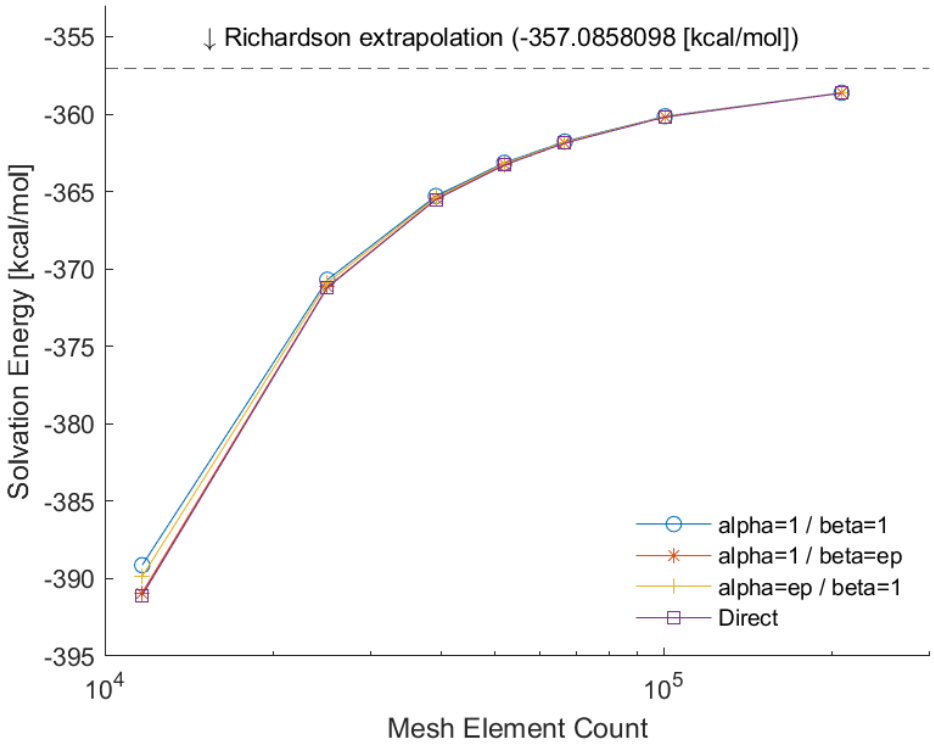


Figure 6: Solvation energy convergence with mesh refinement for 1BPI biomolecule.

From the results it can be seen that the convergence of the different formulations, as the mesh was refined, was in accordance with what is expected for these numerical calculations. The results converge toward the calculated Richardson extrapolation in an asymptotic fashion. From these results it was decided that a density of 10 $[\frac{vertices}{\text{Å}^2}]$ is sufficiently refined, this given that the results in this case present an error of between 1.31% and 1.34%, when compared with the estimated exact solutions.

4.2 Preconditioning and formulation comparison

A series of different formulation and preconditioner combinations were tested, calculating the solvation energy of the 1BPI molecule. The objective of this was to compare the different combinations using a variety of parameters, in order to determine which of these is the most convenient to use. These calculations were

performed using a single mesh density, this was chosen, for the reasons given in the previous section, to be 10 $[\frac{vertices}{\text{Å}^2}]$. This mesh contained 66596 triangular elements and 33302 vertices. For these calculations the GMRES max iteration limit was lowered to 200.

Various parameters of the calculations were measured in order to make comparisons between the different schemes tested, these included, total calculation time, the time taken to resolve the linear system using GMRES, the number of iterations taken and RAM usage. The RAM usage was done utilising a python library to monitor the maximum usage during the calculations. Using the number of iterations and GMRES time, the approximate average time per iteration was also calculated. The difference between the total time and the GMRES time includes things such as the operator assembly, steps required for preconditioning, and the calculation of the solvation energy, but, as can be seen in the results, the dominating time is that of the GMRES solver.

A total of 7 different combinations were tested, the results are presented in the following tables.

Memory Usage [GB]	Total Time [s]	GMRES Time [s]	Number of GMRES Iterations	Aprox. Time per Iteration [s]	Solvation Energy [kcal/mol]	Formulation Type
2.165	419.9	355.3	28	12.7	-361.775	alpha=1 / beta=1
2.143	2459.8	2395.7	>200	12.0	-361.933	alpha=1 / beta=ep
2.132	2509.8	2443.9	>200	12.2	-361.890	alpha=-1 / beta=-ep
2.121	1317.0	1252.1	101	12.4	-361.818	alpha=ep / beta=1
2.127	2488.9	2424.2	>200	12.1	-361.870	alpha=-ep / beta=-1
1.667	993.2	957.1	>200	4.8	-361.817	Direct

Table 2: Weak Form discretisation - No preconditioning

Memory Usage [GB]	Total Time [s]	GMRES Time [s]	Number of GMRES Iterations	Aprox. Time per Iteration [s]	Solvation Energy [kcal/mol]	Formulation Type
2.127	305.3	228.3	17	13.4	-361.775	alpha=1 / beta=1
2.232	2566.0	2486.2	>200	12.4	-363.127	alpha=1 / beta=ep
2.131	2585.9	2505.9	>200	12.5	-362.100	alpha=-1 / beta=-ep
2.191	365.2	283.9	21	13.5	-361.817	alpha=ep / beta=1
2.167	2562.3	2482.9	>200	12.4	-361.818	alpha=-ep / beta=-1
1.706	460.3	416.4	81	5.1	-361.874	Direct

Table 3: Weak Form discretisation - Block Diagonal preconditioning

Memory Usage [GB]	Total Time [s]	GMRES Time [s]	Number of GMRES Iterations	Aprox. Time per Iteration [s]	Solvation Energy [kcal/mol]	Formulation Type
2.129	301.5	236.0	18	13.1	-361.775	alpha=1 / beta=1
2.100	321.8	255.4	20	12.8	-361.865	alpha=1 / beta=ep
2.171	350.8	287.6	22	13.1	-361.880	alpha=-1 / beta=-ep
2.119	834.2	769.7	62	12.4	-361.815	alpha=ep / beta=1
2.133	1801.5	1734.9	139	12.5	-361.820	alpha=-ep / beta=-1
1.717	982.3	943.5	187	5.0	-361.874	Direct

Table 4: Strong Form discretisation - No preconditioning

Memory Usage [GB]	Total Time [s]	GMRES Time [s]	Number of GMRES Iterations	Aprox. Time per Iteration [s]	Solvation Energy [kcal/mol]	Formulation Type
2.121	955.7	878.6	35	25.1	-361.734	alpha=1 / beta=1
2.159	1579.8	1504.4	60	25.1	-361.864	alpha=1 / beta=ep
2.169	410.7	334.1	12	27.8	-361.880	alpha=-1 / beta=-ep
2.175	4999.6	4919.9	>200	24.6	-360.672	alpha=ep / beta=1
2.197	5000.3	4923.1	>200	24.6	-361.615	alpha=-ep / beta=-1

Table 5: Strong Form discretisation - Squared Calderón preconditioning

Memory Usage [GB]	Total Time [s]	GMRES Time [s]	Number of GMRES Iterations	Aprox. Time per Iteration [s]	Solvation Energy [kcal/mol]	Formulation Type
2.203	754.5	677.9	31	21.9	-361.776	alpha=1 / beta=1
2.175	1061.1	981.5	45	21.8	-361.863	alpha=1 / beta=ep
2.245	641.1	565.7	26	21.8	-361.880	alpha=-1 / beta=-ep
2.168	4229.5	4153.1	>200	20.8	-361.803	alpha=ep / beta=1
2.219	4230.3	4156.1	>200	20.8	-361.752	alpha=-ep / beta=-1

Table 6: Strong Form discretisation - Exterior Calderón Operator preconditioning

Memory Usage [GB]	Total Time [s]	GMRES Time [s]	Number of GMRES Iterations	Aprox. Time per Iteration [s]	Solvation Energy [kcal/mol]	Formulation Type
2.124	586.7	517.5	33	15.7	-361.776	alpha=1 / beta=1
2.143	591.8	521.2	33	15.8	-361.854	alpha=1 / beta=ep
2.153	385.9	315.3	20	15.8	-361.870	alpha=-1 / beta=-ep
2.158	3065.6	2993.8	>200	15.0	-361.767	alpha=ep / beta=1
2.094	3105.5	3036.2	>200	15.2	-361.623	alpha=-ep / beta=-1

Table 7: Strong Form discretisation - Interior Calderón Operator preconditioning

Memory Usage [GB]	Total Time [s]	GMRES Time [s]	Number of GMRES Iterations	Aprox. Time per Iteration [s]	Solvation Energy [kcal/mol]	Formulation Type
2.192	335.5	259.1	11	23.6	-361.876	alpha=1 / beta=ep
2.154	331.4	257.0	11	23.4	-361.869	alpha=-1 / beta=-ep

Table 8: Strong Form discretisation - Exterior Calderón Projector preconditioning

These results contain lots of information that can be used to compare the different combinations tested. It is not immediately obvious which is the best combination and a variety of factors must be taken into account.

For the base case, with weak form discretisation and no preconditioning, it can be seen that out of the 6 formulations, 4 of these reached the 200 maximum iteration limit before the desired tolerance was obtained, and so the timings of these results should not be used for detailed comparisons. The two other formulations converge before the limit, however, they performed poorly when compared with other combinations, taking both longer in terms of total time and having a higher GMRES iteration count.

The formulations $\alpha = 1$ / $\beta = \epsilon$ and $\alpha = -1$ / $\beta = -\epsilon$ performed very well with the different Calderón preconditioning types, with a lower number of GM-

RES iterations required for convergence. This can be explained by studying these formulations in (94) and (95). The left hand sides of these equations consist in the addition or subtraction of the interior Calderón, the scaled exterior Calderón and identity operators. For this reason, and given the identities presented in section 2.4.3, it can be expected that the application of the different Calderón preconditioners would perform well. In particular, the use of exterior Calderón projector preconditioning greatly improved the convergence, requiring just 11 iterations for both the mentioned formulations. This is substantially lower than the 17 of the next lowest combination ($\alpha = 1$ / $\beta = 1$ - weak form with block diagonal preconditioning). However, the problem, as is the case when using any of the Calderón preconditioning methods, is that each GMRES iteration took longer, given that it requires more matrix-vector operations. This can be seen by looking at the calculated approximate time per iteration. When using strong form discretisation without any preconditioning, the time per iteration for all the alpha-beta type formulations was ≈ 13 [s]. If squared Calderón preconditioning was applied, the time per iteration was approximately double ($\approx$ 25-27 [s]). This is expected given that twice as many matrix-vector operations must be performed. For the other Calderón preconditioning schemes this time was somewhere in between. In all the cases tested any reduction in the number of GMRES iterations required appears to be offset, given that these take longer, and in most cases the total time was longer than if preconditioning had not been applied. It may be possible to apply Calderón type preconditioning with the use of a sparser matrix, which would mean that the increase in iteration time may be less, and the overall time be lower. This point was not investigated further for this work and remains a possible area of future study.

The formulations with a fixed value of $\beta = 1$ and different positive values of α, appeared to perform well under the application of block diagonal preconditioning. There was an improvement in the number of GMRES iterations required in the case of both the $\alpha = 1$ / $\beta = 1$ and $\alpha = \epsilon$ / $\beta = 1$ formulations. The combination

with the lowest number of iterations utilised this type of preconditioning, and one of its great advantages is that the increase in the time per iteration was only small, given that the preconditioning matrix is sparse. It is seen that the block diagonal preconditioning lowered more greatly the iterations required in the case of $\alpha = \epsilon$ / $\beta = 1$ (Juffer) over the $\alpha = 1$ / $\beta = 1$ formulation. This is an interesting observation and raises the question of what occurs for other values of α.

The direct formulation results show a few advantages over the other combinations. It used much less memory and it also had a much lower time per iteration. These two advantages can be explained because the formulation uses half the number of boundary operators in comparison to the alpha-beta type formulations. However, convergence was obtained with an elevated number of iterations, when compared to other combinations. This means that although the time per iteration is lower, the best total time was 52.7 % greater than that of the fastest overall combination. This best time was obtained with the use of the direct formulation with weak form discretisation and block diagonal preconditioning. The application of this preconditioning type improved greatly the convergence, taking just 87 iterations. On the other hand, if no preconditioning was applied then the 200 iteration limit was reached before the desired tolerance could be obtained.

Memory usage is very close for all alpha-beta type formulations, with only a low variation between them. In the case of the direct method memory usage is noticeably lower, which is to be expected given that it only requires two boundary operators, where as the other formulations have 4. As a comparison, some calculations were performed using different alpha-beta formulations with dense operator assembly, the memory usage in these cases was ≈ 60 [GB], much higher than was used with FMM assembly. The results above do not give information about the change in RAM usage as the number of elements increases. However, this shows the use of FMM is very economical on RAM usage, even with the number of elements tested, and that the workstation used should be able to perform calculations with meshes of much greater number of elements with ease. The

extra usage of RAM by the alpha-beta formulations over direct is of no concern at this number of elements and is almost a depreciable effect, given current computational workstation specifications. In order to study memory usage depending on the number of elements in the mesh, the same procedure as used to study the mesh convergence was performed again, measuring the memory usage of each mesh density. The results are shown in the following figure.

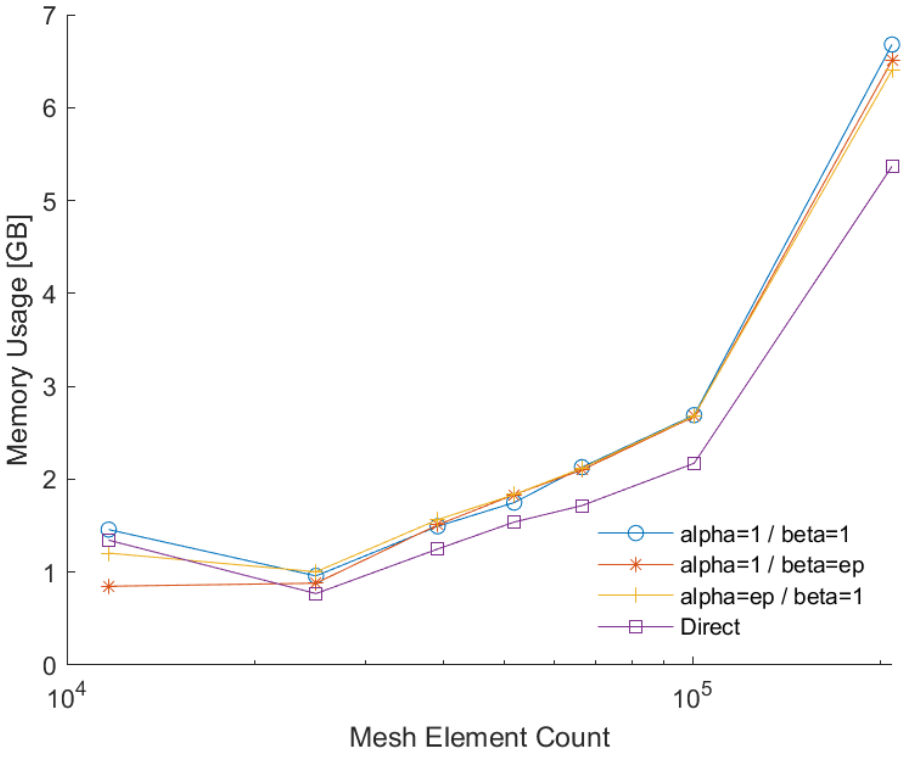


Figure 7: Memory usage for different formulations and mesh densities

From these results it can be seen that for lower mesh densities there is a greater disparity between the memory used by the different formulation, however, the total usage of them all is relatively low. For higher mesh densities the usage is more stable between the different formulations. The direct formulation used about 20% less memory than the formulation with the highest usage. Even with about 200,000 elements in the mesh the highest RAM usage was approximately 6.7$[GB]$, not a very high value for modern computational hardware. Thus the

point can still be made that although the direct formulation uses less memory, it is not in an amount that would make its usage immediately desirable over the other formulations.

From all of these observations it was decided to further investigate the combination of weak form discretisation with block diagonal preconditioning of the alpha-beta formulation.

4.3 Sphere results

After the previous results showed that the combination with the lowest number of iterations was weak form discretisation with block diagonal preconditioning of the $\alpha = 1$ / $\beta = 1$ formulation. Also, the block diagonal preconditioning had a great effect on the Juffer ($\alpha = \epsilon$ / $\beta = 1$) formulation, lowering the iterations to 21, from the next best of 62. For these reasons it was decided to further test these two combinations, this time using a sphere with a single point charge at its centre. To test if the value of the charge, κ or the size of the sphere had an effect on the conditioning of the system, the potential was calculated using 3 different sized spherical meshes with roughly the same number of elements. In one case the value of the point charge was changed, and in the other the value of κ. The results are presented in the following two tables.

Charge = 1.0

alpha = ep / beta = 1 (Juffer)					alpha = 1 / beta = 1				
Sphere Radius [Å]	Total Time [s]	GMRES Time [s]	Number of GMRES Iterations	Solvation Energy [kcal/mol]	Sphere Radius [Å]	Total Time [s]	GMRES Time [s]	Number of GMRES Iterations	Solvation Energy [kcal/mol]
15	144	105.5	5	-2.7193	15	144	105.5	5	-2.7192
35	157	118.5	5	-1.1750	35	173	133.8	6	-1.1750
68	154	115.1	5	-0.6072	68	206	167.8	8	-0.6072

Charge = 10.0

alpha = ep / beta = 1 (Juffer)					alpha = 1 / beta = 1				
Sphere Radius [Å]	Total Time [s]	GMRES Time [s]	Number of GMRES Iterations	Solvation Energy [kcal/mol]	Sphere Radius [Å]	Total Time [s]	GMRES Time [s]	Number of GMRES Iterations	Solvation Energy [kcal/mol]
15	142	104.8	5	-271.9251	15	144	106.9	5	-271.9244
35	161	119.4	5	-117.4985	35	176	136.4	6	-117.4989
68	157	115.7	5	-60.7238	68	206	166.4	8	-60.7242

Charge = 100.0

alpha = ep / beta = 1 (Juffer)					alpha = 1 / beta = 1				
Sphere Radius [Å]	Total Time [s]	GMRES Time [s]	Number of GMRES Iterations	Solvation Energy [kcal/mol]	Sphere Radius [Å]	Total Time [s]	GMRES Time [s]	Number of GMRES Iterations	Solvation Energy [kcal/mol]
15	143	105.2	5	-27192.5114	15	142	104.1	5	-27192.4440
35	157	118.3	5	-11749.8452	35	173	134.0	6	-11749.8897
68	158	117.2	5	-6072.3819	68	208	167.3	8	-6072.4160

Table 9: Results of testing effect of different charge values on convergence.

Changes to the value of the point charge appear to have no effect on the convergence of the system, as this required the same number of iterations for the 3 different values tested. However, it can be noted that the $\alpha = 1$ / $\beta = 1$ formulation presented a varying number of iterations necessary for convergence, depending on the radius of the sphere.

κ = 0.0625

alpha = ep / beta = 1 (Juffer)					alpha = 1 / beta = 1				
Sphere Radius [Å]	Total Time [s]	GMRES Time [s]	Number of GMRES Iterations	Solvation Energy [kcal/mol]	Sphere Radius [Å]	Total Time [s]	GMRES Time [s]	Number of GMRES Iterations	Solvation Energy [kcal/mol]
15	143	102.9	5	-269.5963	15	143	104.9	5	-269.5941
35	155	115.4	5	-116.7412	35	174	134.7	6	-116.7410
68	139	99.8	4	-60.4636	68	173	134.2	6	-60.4638

κ = 0.125

alpha = ep / beta = 1 (Juffer)					alpha = 1 / beta = 1				
Sphere Radius [Å]	Total Time [s]	GMRES Time [s]	Number of GMRES Iterations	Solvation Energy [kcal/mol]	Sphere Radius [Å]	Total Time [s]	GMRES Time [s]	Number of GMRES Iterations	Solvation Energy [kcal/mol]
15	142	104.8	5	-271.9251	15	144	106.9	5	-271.9244
35	161	119.4	5	-117.4985	35	176	136.4	6	-117.4989
68	157	115.7	5	-60.7238	68	206	166.4	8	-60.7242

κ = 0.250

alpha = ep / beta = 1 (Juffer)					alpha = 1 / beta = 1				
Sphere Radius [Å]	Total Time [s]	GMRES Time [s]	Number of GMRES Iterations	Solvation Energy [kcal/mol]	Sphere Radius [Å]	Total Time [s]	GMRES Time [s]	Number of GMRES Iterations	Solvation Energy [kcal/mol]
15	145	106.4	5	-273.8252	15	163	125.0	6	-273.8260
35	160	120.7	5	-117.9937	35	208	169.0	8	-117.9944
68	172	132.8	6	-60.8757	68	236	197.2	10	-60.8760

Table 10: Results of testing effect of different κ values on convergence.

In this case a change to the value of kappa appeared to have an influence on the convergence of both the tested formulations. Again, a dependency of the radius of sphere on the number of iterations taken is also visible.

The linear system to be resolved in this problem can be written as $Ax = b$, where the condition number of matrix A is what determines the amount of iterations needed to reach the desired tolerance. Matrix A consists of 4 different blocks and one would expect the top left and bottom right blocks to play an important part in the conditioning of the entire matrix, this given that they contain the elements that make up the diagonal of A. The top left block of the alpha-beta formulation consists of

$$-K_{int} + \alpha K_{ext} - \frac{1}{2} I(1 + \alpha), \tag{124}$$

and is derived from equation (91). As for the bottom right block we have

$$T_{int} - \frac{\beta}{\epsilon} T_{ext} - \frac{1}{2} I(1 + \frac{\beta}{\epsilon}). \tag{125}$$

Taking the double layer (K) boundary operator, defined in section 2.4.2, and simplifying for the case of a sphere of radius R with single centred point charge, omitting the potential, yields

$$K_{int} = -\frac{1}{4\pi R^2} \int_{\Gamma} ds = -\frac{1}{4\pi R^2} 4\pi R^2 = -1, \tag{126}$$

$$K_{ext} = -\frac{e^{-\kappa R}(\kappa R + 1)}{4\pi R^2} \int_{\Gamma} ds = -\frac{e^{-\kappa R}(\kappa R + 1)}{4\pi R^2} 4\pi R^2 = -e^{-\kappa R}(\kappa R + 1). \tag{127}$$

It can be seen that although internal operator is constant, the exterior operator has a dependency on κR factor. This could explain why changes in these values affect the conditioning of the whole system. This same dependency is mirrored with the internal and external adjoint double layer operators, with T_{int} being constant and T_{ext} depending on κR.

The value of κ is a constant that reflects a physical property (the concentration of salt in the exterior domain) and so often does not vary far from the base value (0.125 [Å^{-1}]) used for these calculations. It was decided to focus on how changes in the radius of the sphere and the value of α used, affect the conditioning of the system. Using the same combination of formulation and preconditioner (Block-Diagonal with alpha-beta) the solvation energy of spheres with increasing radii was calculated, this was done for a varity of values of α. These spheres contained a single point charge at their centres with a value of 10 [e^-]. The spheres were of radii ranging from 15 to 240 [Å]. The number of elements of the generated meshes was kept constant at 8988 elements and 4496 vertices. The value of β was fixed at $\beta = 1$ and α took a series of values between 1 and 100.

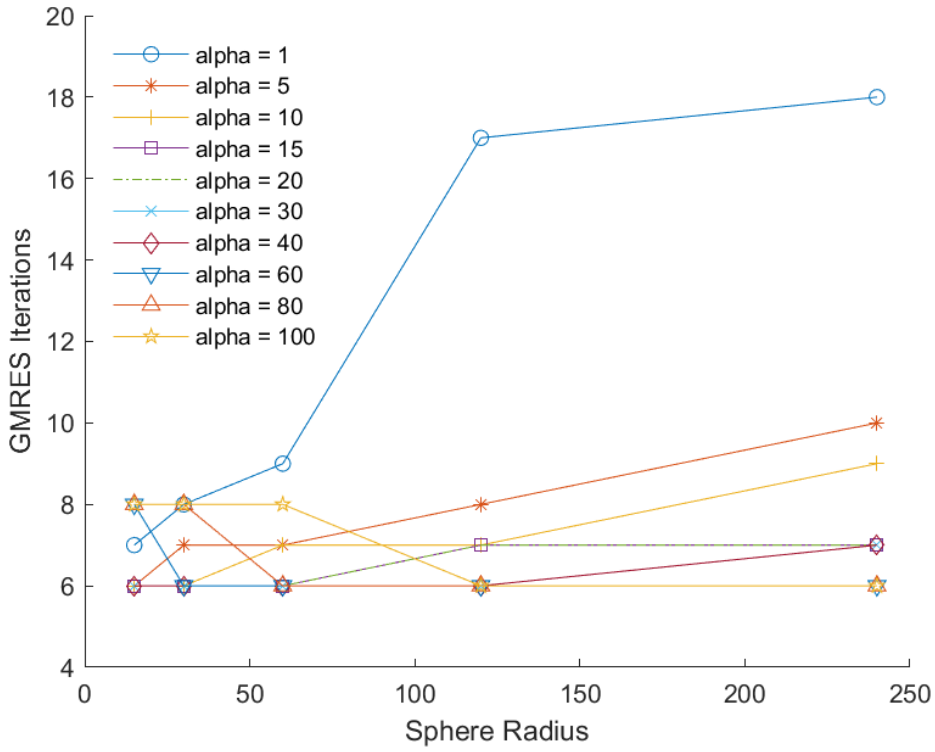


Figure 8: Converge of sphere with single point charge of different radii [Å] for a variety of values of α

From the graph it can be seen that with higher values of alpha, the radius for which the least number of GMRES iterations increases. This is an interesting indication, as it suggests that the size of the biomolecule for which one wishes to calculate the solvation energy, could have an effect on which is the best formulation to use. For example, when $\alpha = 1$ the lowest number of GMRES iterations occurs for a radius of 15, while for $\alpha = 100$ the lowest iterations are for radii 120 and 240. Another observation is that the Juffer formulation ($\alpha = 20$) the lowest iterations occur for the radii 15, 30 and 60 [Å], this shows a larger range of radii where it performs well in combination with the block diagonal preconditioner.

Looking at equations (124) and (125) it can been seen that the α and β parameters have an effect on the diagonal dominance of the corresponding blocks. In the case of the results above, when the value of α is increased the dominance of

the diagonal of the top left block becomes greater, and this seems to aid converge of larger spheres. This altering of the diagonals could explain the different convergence behaviour as the radius increases, scaling the blocks in a way that maintains the conditioning of the entire matrix. To further understand this, greater analysis of the matrix is required, possibly through the calculation of the condition number for a problem with few elements, or the study of the singular values and their grouping, something that has been commented on in other publications [18].

4.4 Increasing molecule size

To get data on the convergence of other real molecules with different values of α, the solvation energy was calculated for 6 proteins of different sizes. The proteins used were 1BPI, 1LYZ, 1A7M, 1X1Z, 4LGP and 1IGT, and their surface meshes are shown in the following figure.

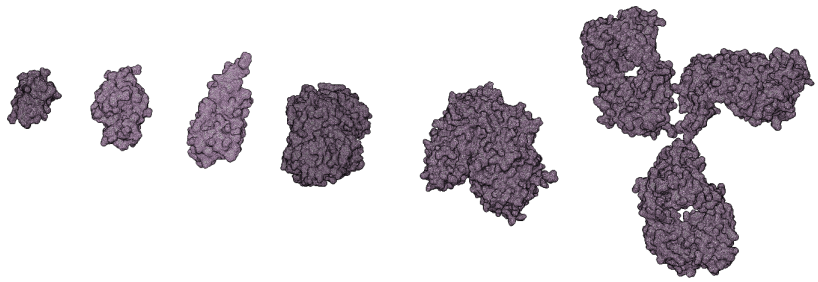


Figure 9: Surface meshes of the 6 different proteins that were used. From left to right these are, 1BPI, 1LYZ, 1A7M, 1X1Z, 4LGP and 1IGT

Again this was using the weak form - block diagonal preconditioning combination that has shown to use the least iterations (other than exterior Calderón projector) and low total time. Meshes were produced for each of the proteins such that the number elements of each mesh was between 100,000 and 111,000. The solvation energy was then calculated for each molecule 4 times, using different values of α. Data on the total time taken, GMRES time, GMRES iterations and the calculated energy was collected, and is presented in the following tables.

1BPI					
Number of Mesh Elements	Total Time [s]	GMRES Time [s]	Number of GMRES Iterations	Solvation Energy [kcal/mol]	Alpha Value
100616	402	290	16	-360.142841	1
100616	513	398	21	-360.165934	20
100616	539	429	23	-360.166574	40
100616	812	699	39	-360.167131	80

1LYZ					
Number of Mesh Elements	Total Time [s]	GMRES Time [s]	Number of GMRES Iterations	Solvation Energy [kcal/mol]	Alpha Value
107432	469	339	17	-570.282628	1
107432	565	435	22	-570.365256	20
107432	661	532	27	-570.368163	40
107432	811	680	35	-570.369565	80

1A7M					
Number of Mesh Elements	Total Time [s]	GMRES Time [s]	Number of GMRES Iterations	Solvation Energy [kcal/mol]	Alpha Value
107436	523	376	18	-630.481876	1
107436	616	473	23	-630.614399	20
107436	703	564	28	-630.619064	40
107436	858	718	37	-630.621495	80

Table 11: Calculations of solvation energy using larger real molecules and alpha-beta formulation.

1X1Z					
Number of Mesh Elements	Total Time [s]	GMRES Time [s]	Number of GMRES Iterations	Solvation Energy [kcal/mol]	Alpha Value
110924	647	420	20	-1670.12656	1
110924	722	503	24	-1670.9009	20
110924	818	594	30	-1670.92112	40
110924	1064	839	42	-1670.93151	80

4LGP					
Number of Mesh Elements	Total Time [s]	GMRES Time [s]	Number of GMRES Iterations	Solvation Energy [kcal/mol]	Alpha Value
108632	852	548	26	-2321.32631	1
108632	772	482	23	-2327.40112	20
108632	958	658	32	-2327.65179	40
108632	1084	792	40	-2327.78149	80

1IGT					
Number of Mesh Elements	Total Time [s]	GMRES Time [s]	Number of GMRES Iterations	Solvation Energy [kcal/mol]	Alpha Value
110200	1082	642	28	-5103.59537	1
110200	933	516	23	-5133.9429	20
110200	1079	671	30	-5135.20348	40
110200	1285	871	40	-5135.84608	80

Table 12: Calculations of solvation energy using larger real molecules and alpha-beta formulation. (Cont.)

It can be seen that for the 4 smaller molecules the value of α with the lowest time and iteration count is $\alpha = 1$, but for the two larger molecules this occurs for $\alpha = 20$. In this case, given the parameters used, $\alpha = 20$ corresponds to the Juffer formulation. These results are very interesting as they also seem to suggest that the value of α can be fine tuned, depending on the size of the biomolecule, in order to obtain a more well-conditioned system, that can be resolved in less time and iterations. The results do not show whether values higher than $\alpha = 20$ work better in any cases, this may be because the molecules tested do not have the size to show this, or that the Juffer formulation is best for all larger molecules.

Biomolecules have shapes that are rarely spherical and so it is difficult to compare their size to the spheres tested. Two methods were used in order to estimate a characteristic radius of the molecules and compare their size and convergence pattern to the sphere case. First, taking the surface area of the molecules and assuming that they were a sphere, the radius was calculated as $R = \sqrt{\frac{\text{area of molecule}}{4\pi}}$. Secondly, again assuming the molecule to be a sphere but this time using its volume, obtaining $R = \sqrt[3]{\frac{3 \text{ volume of molecule}}{4\pi}}$. The calculated values for each of the biomolecules is presented below.

Molecule	Surface area [Å^2]	Radius calculated from surface area [Å]	Volume [Å^3]	Radius calculated from volume [Å]
1BPI	3246.4	16.1	7109.4	11.9
1LYZ	5652.7	21.2	15924.5	15.6
1A7M	7984.2	25.2	21598.0	17.3
1X1Z	14054.5	33.4	54713.8	23.6
4LGP	28450.9	47.6	96024.8	28.4
1IGT	53263.8	65.1	168135.0	34.2

Table 13: Estimated radii of biomolecules, based on assuming them to be sphere like.

There are large differences between the estimated radii given by the two methods. This does not give a clear picture to compare the convergence of the spheres in the previous section with these real molecules. However, it can be seen that these molecules are not of a particularly "large" size, and so the possible cross over to a higher value of α performing better than the Juffer formulation may require the use of a larger biomolecule to be seen. Further testing would be required on larger bodies in order to see if the trend observed with the spheres also occurs for real molecules. However, these results along with those performed on the sphere suggest that better convergence can be achieved by fine tuning the value of α.

5 Conclusions

In this work we studied the implicit solvent Poisson-Boltzmann model and applied it to the case of electrostatic interactions of biomolecules in an ionic solution (water with salt). The boundary element method (BEM) was then applied to the resulting system of equations, obtaining the internal and external representation formulas. From these different formulations were obtained, two of these (direct and Juffer) having been previously documented. A more generalised version of Juffer was derived, based on the parameters α and β. Along with this, different possible preconditioners were identified from previous work.

Various tests were performed with combinations of the preconditioners and formulations. Computational parameters, such as total time taken, RAM used and iterations required, were recorded and then used to determine the pros and cons of each combination. At this stage the best combination appeared to be weak form $\alpha = 1$ / $\beta = 1$ with block diagonal preconditioning, taking the least amount of iterations and almost the lowest total time. Calderón preconditioning, although producing the combinations with the lowest iteration counts, caused large increases in time per iteration which led to longer overall times.

Further tests performed on a simplified sphere problem showed how the generalised alpha-beta formulation reacted to changes in the values of the charge, κ and the radius. This lead to the observation that the radius and κ have an effect on the convergence of the system, and that the value of α can be fine tuned to obtain the quickest convergence, depending on the radius of the sphere.

Finally, the generalised alpha-beta formulation was tested on larger biomolecules. This again showed that the best value for α appears to depend on the size of the solute region. However, more study is needed to understand fully the reason behind this and how best to use it.

Further research is required to find a rule to aid selection of the absolute best combination given a particular biomolcule, perhaps through a characteristic size

or other parameter. However, the best combination for the smaller biomolecules tested in this work was weak form $\alpha = 1$ / $\beta = 1$ with block diagonal preconditioning, and for the case of the larger molecules it was weak form $\alpha = \epsilon$ / $\beta = 1$ (Juffer) with block diagonal preconditioning. Another interesting point, is that if a method could be determined to lower the increase in time per iteration of the Caldéron preconditioning, it could be a faster overall combination, given that it can greatly improve the number of iterations required in certain cases.

References

[1] Byung Jun Yoon and A. M. Lenhoff. A boundary element method for molecular electrostatics with electrolyte effects. Journal of Computational Chemistry, 11(9):1080–1086, October 1990.

[2] André H. Juffer, Eugen F.F. Botta, Bert A.M. van Keulen, Auke van der Ploeg, and Herman J.C. Berendsen. The electric potential of a macromolecule in a solvent: A fundamental approach. Journal of Computational Physics, 97(1):144–171, November 1991.

[3] Timo Betcke and Matthew Scroggs. bempp-cl open-source computational boundary element platform. https://bempp.com/, August 2020.

[4] Christopher David Cooper Villagrán. Biomolecular electrostatics with continuum models: a boundary integral implementation and applications to biosensors. PhD thesis, Boston University, Boston, MA, United States, 2015.

[5] T. L. Hill. An Introduction to Statistical Thermodynamics. Addison-Wesley Publishing Company, 1960.

[6] Christopher D. Cooper, Jaydeep P. Bardhan, and L.A. Barba. A biomolecular electrostatics solver using python, gpus and boundary elements that can handle solvent-filled cavities and stern layers. Computer Physics Communications, 185(3):720–729, 2014.

[7] Weihua Geng and Robert Krasny. A treecode-accelerated boundary integral Poisson-Boltzmann solver for electrostatics of solvated biomolecules. Journal of Computational Physics, 247:62–78, August 2013.

[8] John G. Kirkwood. Theory of solutions of molecules containing widely separated charges with special application to zwitterions. The Journal of Chemical Physics, 2:351–361, 7 1934.

[9] J. Liang and S. Subramaniam. Computation of molecular electrostatics with boundary element methods. Biophysical Journal, 73(4):1830 – 1841, 1997.

[10] Michael D. Altman, Jaydeep P. Bardhan, Jacob K. White, and Bruce Tidor. Accurate solution of multi-region continuum biomolecule electrostatic problems using the linearized poisson–boltzmann equation with curved boundary elements. Journal of Computational Chemistry, 30(1):132–153, 2009.

[11] Barba Group. exafmm-t, kernel-independent fast multipole method library. https://github.com/exafmm/exafmm-t, August 2020.

[12] Antigoni Kleanthous, Timo Betcke, David P. Hewett, Matthew W. Scroggs, and Anthony J. Baran. Calderón preconditioning of pmchwt boundary integral equations for scattering by multiple absorbing dielectric particles. Journal of Quantitative Spectroscopy and Radiative Transfer, 224:383–395, Feb 2019.

[13] Timo Betcke, Matthew Scroggs, and Wojciech Smigaj. Product algebras for galerkin discretisations of boundary integral operators and their applications. ACM Transactions on Mathematical Software, 46, 11 2017.

[14] Stefan Search. bem_electrostatics python library. https://github.com/SDSearch/bem_electrostatics, August 2020.

[15] Elizabeth Jurrus, Dave Engel, Keith Star, Kyle Monson, Juan Brandi, Lisa E. Felberg, David H. Brookes, Leighton Wilson, Jiahui Chen, Karina Liles, Minju Chun, Peter Li, David W. Gohara, Todd Dolinsky, Robert Konecny, David R. Koes, Jens Erik Nielsen, Teresa Head-Gordon, Weihua Geng, Robert Krasny, Guo-Wei Wei, Michael J. Holst, J. Andrew McCammon, and Nathan A. Baker. Improvements to the apbs biomolecular solvation software suite. Protein Science, 27(1):112–128, 2018.

[16] Michel F. Sanner, Arthur J. Olson, and Jean-Claude Spehner. Reduced surface: An efficient way to compute molecular surfaces. Biopolymers, 38(3):305–320, 1996.

[17] Sergio Decherchi and Walter Rocchia. A general and robust ray-casting-based algorithm for triangulating surfaces at the nanoscale. PLOS ONE, 8(4):1–15, 04 2013.

[18] Jiahui Chen and Weihua Geng. On preconditioning the treecode-accelerated boundary integral (tabi) poisson–boltzmann solver. Journal of Computational Physics, 373:750 – 762, 2018.