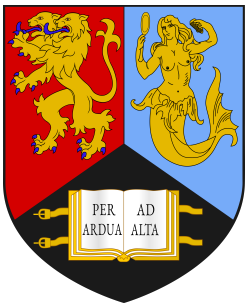
A Coarse approach to the Kernel-Independent Fast Multipole Method for the computation of large regularized Stokeslet systems

Samuel Walker

Student ID : 1888627

Supervisor : D.J Smith

Co-Supervisor : T.D Montenegro-Johnson



School of Mathematics

University of Birmingham

United Kingdom

May 2022

Abstract

ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Acknowledgements

ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Contents

List of Figures

List of Tables

1 Introduction

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u}\right) = \mathbf{F} - \nabla p + \mu\Delta\mathbf{u}$$
$$\nabla\cdot\mathbf{u} = 0 \tag{1.1}$$

where ρ is the fluid density, $\mathbf{u}$ is the velocity of the fluid, $\mathbf{F}$ is the external force, p is the pressure and μ is the fluid viscosity

2 Method of regularised stokeslets

2.1 Stokes flow

When we analyse many physical systems, particularly in microscopic biology, we find that the inertial forces within the fluid are small in comparison to that of the viscous terms. In these cases we can take the limit of the Navier-Stokes equation (eq. (1.1))where $Re = \rho u L/\mu \to 0$ [1]. In this limit, we obtain the steady-state Stokes equations in two or three dimensions as

$$\mu\Delta\boldsymbol{u} = \nabla p - \boldsymbol{F} \tag{2.1a}$$

$$\nabla\cdot\boldsymbol{u} = 0 \tag{2.1b}$$

where $\boldsymbol{u}$ is the velocity of the fluid, $\boldsymbol{F}$ is the external body force, p is the pressure and μ is the fluid viscosity.

Stokes flows occur when either where μ is large or at small length scales where L (typical scale of the system) is very small such as when analysing the flows around cell, microorganism's or small capillaries [2, 3, 4]. Stokes flow is not only useful in modelling flows in microbiology but in non-biological systems such are industrial applications with small scale flows and complex boundaries stokes flow with complex boundaries [5, 6].

When looking for solutions to the the Stokes equation eq. (2.1) there are many different techniques we can use both numerically and analytically we will be focusing on the use of the Green's function of Stokes flow called a *stokeslet*[7, 8] or *Oseen tensor*[9]. The

stokeslet represents a fundamental solution for the velocity of the fluid given that an external force $\boldsymbol{F}$ acts on the fluid at a single point $\boldsymbol{F} = \boldsymbol{f_0}\delta(\boldsymbol{x})$ [8, 10]. Through similar methods to those shown later for deriving the regularised stokeslet Equation, (eq. (2.19a)) we can derive the singular fundamental solutions to the Stokes equations as

$$\begin{aligned} S_{jk}(\boldsymbol{x}, \boldsymbol{y}) &= \frac{\delta_{jk}}{r} + \frac{(x_j - y_j)(x_k - y_k)}{r^3} \\ P_k(\boldsymbol{x}, \boldsymbol{y}) &= \frac{x_k - y_k}{r^3} \\ T_{ijk}(\boldsymbol{x}, \boldsymbol{y}) &= \frac{-6(x_i - y_i)(x_j - y_j)(x_k - y_k)}{r^5} \end{aligned} \tag{2.2}$$

Where $r = |\boldsymbol{x} - \boldsymbol{y}|$. We can then find the velocity $\boldsymbol{u}$ and pressure p at a point $\boldsymbol{x}$ through the equations

$$\boldsymbol{u} = (8\pi\mu)^{-1}(S_{1j}, S_{2j}, S_{3j})\, f_{0,j}, \quad p = (8\pi)^{-1} P_k f_{0,j}$$

where $\boldsymbol{f_0}$ is the force per unit area exerted by the fluid on the surface, concentrated at the point $\boldsymbol{y}$.

2.2 Regularising the stokeslet

While the singular stokeslet provides a useful mechanism to solve boundary integral equations, the solutions contain singular points which prove computationally challenging unless careful consideration of the the quadrature discretization is considered where we don not need to evaluate close to these singular points.

In order to remove these singularities, we use a cutoff function [11, 12] which, instead of approximating the force at a singular point we instead approximate it as a sphere centred at the same point. While the radius of the sphere is often infinite, the cutoff function decays rapidly away from the centre with the largest contribution obtained in the close vicinity of the centre. We introduce a control parameter ϵ, independent of any discretization, to control the rate of decay of the function. The effect of this control parameter can be seen in fig. 1. In order to obtain similar results to that of the singular solutions, we dictate that $\int \phi^{\epsilon}(r)dr = 1$ for all values of ϵ . This allows us to preserve the results obtained for the singular kernels for points far away from where the force

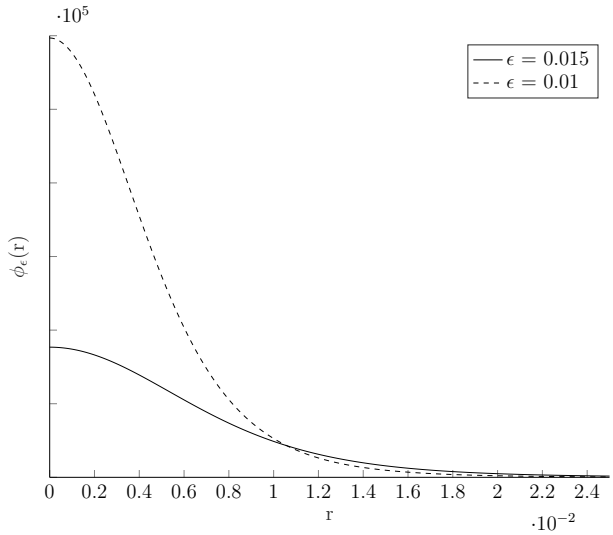


Figure 1: Cutoff function in equation eq. (2.3) for several values of ϵ

is exerted and obtain different results close to the point due to the regularisation error introduced. In order to retain the singular solution, we have the condition that as $\epsilon \to 0$ our cutoff function must tend to the Dirac delta function. For simplicity of the paper and the application to a wider range of problems [13], we will only consider spherically symmetric functions such as those in eqs. (2.3) to (2.5) [13, 14, 15].

$$\phi^\epsilon(r) = \frac{15\epsilon^4}{8\pi\left(r^2+\epsilon^2\right)^{7/2}} \tag{2.3}$$

$$\phi^\epsilon(r) = \frac{15\epsilon^6\left(5-\frac{2r^2}{\epsilon^2}\right)}{16\pi\left(r^2+\epsilon^2\right)^{9/2}} \tag{2.4}$$

$$\phi^\epsilon(r) = \frac{5\epsilon^2-2r^2}{2\pi^{3/2}\epsilon^5}e^{-r^2/\epsilon^2} \tag{2.5}$$

For the derivation of the regularised stokeslets and all further numerical analysis, we will use eq. (2.3) due to its popularity in external literature and the simplicity of the kernel it generates.

2.2.1 Derivation of the regularised stokeslet

The work done in this section is heavily based of that of Cortez, Fauci and Medovikov[11, 12]. By concentrating the force onto a finite area using the cutoff function rather than a

singular point as with the delta function Cortez proposed we convert the stokes equations given in eq. (2.1) to a new set of regularised equations for which we will derive a set of solutions,

$$\mu\Delta\boldsymbol{u} = \nabla p - \phi^{\epsilon}(\boldsymbol{x}-\boldsymbol{y})\boldsymbol{f_0} \tag{2.6a}$$

$$\nabla\cdot\boldsymbol{u} = 0 \tag{2.6b}$$

where $\boldsymbol{f}_0$ is the force per unit area as defined in the previous section. In order to simplify notation from this point onwards, we will use the Einstein summation convention where repeated indices are summed over. We introduce the regularised stokeslet function $S^{\epsilon}(\boldsymbol{x},\boldsymbol{y})$ which is the Green's function for the velocity $u^{\epsilon}(\boldsymbol{x})$. We can now write the solution to eq. (2.6) as

$$u_i(\boldsymbol{x}) = \frac{1}{8\pi\mu}S^{\epsilon}_{ij}(\boldsymbol{x},\boldsymbol{y})f_{0,j} \tag{2.7}$$

The pressure and stress tensor associated with this flow can now be written as

$$p(\boldsymbol{x}) = \frac{1}{8\pi}P^{\epsilon}_{j}(\boldsymbol{x},\boldsymbol{y})f_{0,j} \tag{2.8}$$

$$\sigma_{ik}(\boldsymbol{x}) = \frac{1}{8\pi}T^{\epsilon}_{ijk}(\boldsymbol{x},\boldsymbol{y})f_{0,j} \tag{2.9}$$

By substituting these solutions back into eq. (2.6a) we find that they must obey

$$\Delta S^{\epsilon}_{kj}(\boldsymbol{x},\boldsymbol{y}) = \frac{\partial P^{\epsilon}_{j}(\boldsymbol{x},\boldsymbol{y})}{\partial x_k} - 8\pi\delta_{kj}\phi^{\epsilon}(\boldsymbol{x}-\boldsymbol{y}) \tag{2.10}$$

for all $j = 1,2,3$ and $k = 1,2,4$ with δ_{kj} being the Kronecker delta function. The incompressibility condition eq. (2.6b) also gives us that

$$\frac{\partial S^{\epsilon}_{ij}(\boldsymbol{x},\boldsymbol{y})}{\partial x_i} = 0 \tag{2.11}$$

for all $j = 1,2,3$. We next take the derivative of eq. (2.10) with respect to x_k to get

$$\frac{\partial S^{\epsilon}_{kj}(\boldsymbol{x},\boldsymbol{y})}{\partial x_i\partial x_i\partial x_k} = \frac{\partial^2 P^{\epsilon}_{j}(\boldsymbol{x},\boldsymbol{y})}{\partial x_k^2} - 8\pi\delta_{kj}\frac{\partial\phi^{\epsilon}(\boldsymbol{x}-\boldsymbol{y})}{\partial x_k}$$

Summing over k as per the convention and using eq. (2.11) gives us

$$\Delta P_j^\epsilon(\boldsymbol{x},\boldsymbol{y}) = 8\pi \frac{\partial \phi^\epsilon(\boldsymbol{x}-\boldsymbol{y})}{\partial x_j}. \tag{2.12}$$

In order to solve the regularised Stokes equation we introduce the regularised Laplace's equation eq. (2.13a) which has solution G^ϵ and the Poisson equation with G^ϵ on the right hand side eq. (2.13b). This is also a regularised biharmonic equation $\Delta\Delta B^\epsilon(\boldsymbol{x}-\boldsymbol{y}) = \phi^\epsilon(\boldsymbol{x}-\boldsymbol{y})$.

$$\Delta G^\epsilon(\boldsymbol{x}-\boldsymbol{y}) = \phi^\epsilon(\boldsymbol{x}-\boldsymbol{y}) \tag{2.13a}$$

$$\Delta B^\epsilon(\boldsymbol{x}-\boldsymbol{y}) = G^\epsilon(\boldsymbol{x}-\boldsymbol{y}) \tag{2.13b}$$

In this change of variable we are now working with scalar potentials G^ϵ and b^ϵ instead of the pressure and velocity. This allows us to express the pressure tensor, second rank stokeslet and Stress tensor in terms of these potentials. By our choice of spherically symmetric cutoff function we also obtain easy solutions to both of these equations. Combining eqs. (2.10), (2.12), (2.13a) and (2.13b) we can form the solution to P^ϵ and S_{ij} in terms of G^ϵ and B^ϵ,

$$P_j^\epsilon(\boldsymbol{x},\boldsymbol{y}) f_{0,j} = 8\pi \frac{\partial G^\epsilon(\boldsymbol{x}-\boldsymbol{y})}{\partial x_j} \tag{2.14}$$

and

$$S_{ij}^\epsilon(\boldsymbol{x},\boldsymbol{y}) = 8\pi \left[\frac{\partial^2 B^\epsilon(\boldsymbol{x}-\boldsymbol{y})}{\partial x_i \partial x_j} - \delta_{ij} G^\epsilon(\boldsymbol{x}-\boldsymbol{y}) \right] \tag{2.15}$$

Using the definition of the regularised stress tensor

$$\sigma_{ij}^\epsilon(\boldsymbol{x}) = -\delta_{ik} p^\epsilon(\boldsymbol{x}) + \mu \left(\frac{\partial u_i^\epsilon}{\partial x_k} + \frac{\partial u_k^\epsilon}{\partial x_i} \right) \tag{2.16}$$

and the definition of velocity in terms of the stokeslet eq. (2.7) we find that the double layer potential can be written as

$$T_{ijk}^\epsilon(\boldsymbol{x},\boldsymbol{y}) = -\delta_{ik} P_j^\epsilon(\boldsymbol{x},\boldsymbol{y}) + \mu \left(\frac{\partial S_{ij}^\epsilon(\boldsymbol{x},\boldsymbol{y})}{\partial x_k} + \frac{\partial S_{kj}^\epsilon(\boldsymbol{x},\boldsymbol{y})}{\partial x_i} \right) \tag{2.17}$$

2.2.2 Specific cutoff function

In order to define usable form's for the S^ϵ, P^ϵ and T^ϵ we need to calculate G^ϵ and B^ϵ for our given choice of cutoff function. By solving the Laplace and Possion equation for eq. (2.3) we obtain that

$$G^\varepsilon(\boldsymbol{x}-\boldsymbol{y}) = \frac{-2r^2+3\epsilon^2}{8\pi(r^2+\epsilon^2)^{3/2}} + \frac{3}{8\pi\epsilon} \tag{2.18a}$$

$$B^\varepsilon(\boldsymbol{x}-\boldsymbol{y}) = -\frac{\sqrt{\epsilon^2+r^2}}{8\pi} + \frac{r^2}{16\pi\epsilon} + \frac{\epsilon}{8\pi} \tag{2.18b}$$

where $r = |\boldsymbol{x}-\boldsymbol{y}|$. We now substitute eqs. (2.18a) and (2.18b) into eqs. (2.14), (2.15) and (2.17) to obtain our final kernels which will be used in all further analysis.

$$S_{ij}^\epsilon(\boldsymbol{x},\boldsymbol{y}) = \delta_{ij}\frac{r^2+2\epsilon^2}{(r^2+\epsilon^2)^{3/2}} + \frac{(x_i-y_i)(x_j-y_j)}{(r^2+\epsilon^2)^{3/2}}, \tag{2.19a}$$

$$P_j^\epsilon(\boldsymbol{x},\boldsymbol{y}) = (x_j-y_j)\frac{2r^2+5\epsilon^2}{(r^2+\epsilon^2)^{5/2}} \text{ and} \tag{2.19b}$$

$$\begin{aligned} T_{ijk}^\epsilon(\boldsymbol{x},\boldsymbol{y}) =& \frac{-6(x_i-y_i)(x_j-y_j)(x_k-y_k)}{(r^2+\epsilon^2)^{5/2}} \\ &- \frac{3\epsilon^2[\delta_{jk}(x_i-y_i)+\delta_{ik}(x_j-y_j)+\delta_{ij}(x_k-y_k)]}{(r^2+\epsilon^2)^{5/2}} \end{aligned} \tag{2.19c}$$

We can easily check that these solutions provide results consistent with those found by the singular solutions as in the limit $\epsilon \to 0$ we obtain the same equations stated in eq. (2.2).

2.2.3 Boundary integral equations

In order to use the above relations to compute the flows around solid boundaries or objects we need to consider the effect of an integral over the boundary of surface, computing the effect of stokeslet at each point on a target point. It has been proven that the solution to elliptic PDE's such as the Stokes flow equation can be solved through the integral over the boundary of the object [16, 7]. The most common starting point for deriving the Boundary integral equations is that of the Lorentz reciprocal identity which states for any two non-singular (regular) flows u and u' with stress σ and σ' respectively we have

that the

$$\frac{\partial}{\partial x_k}(u_i'\sigma_{ik} - u_i\sigma_{ik}') = 0$$

We instead present a modified version of the Lorentz reciprocal identity, if we consider the the Stokes equations

$$\begin{aligned} \mu\Delta\boldsymbol{u} &= \nabla p \\ \nabla\cdot\boldsymbol{u} &= 0 \end{aligned} \tag{2.20}$$

and regularised Stokes equation

$$\begin{aligned} \mu\Delta\boldsymbol{u} &= \nabla p - \phi_\epsilon(\boldsymbol{x}-\boldsymbol{x_0})\boldsymbol{f_0} \\ \nabla\cdot\boldsymbol{u} &= 0 \end{aligned} \tag{2.21}$$

the solutions are linked through the cutoff function and obtain the same results in the limit as $\epsilon \to 0$. It is assumed that the regularised solution is a flow generated by a point force of strength $\boldsymbol{f_0}$ located at a point $\boldsymbol{y}$ while the non-regularised solution is absent of all forces. We let D be a rigid body and assume the point $\boldsymbol{x}$ is outside of D. Then we have that $(\boldsymbol{u}, p)$ satisfies eq. (2.20) with

$$\sigma_{ij}(\boldsymbol{x}) = -\delta_{ik}p(\boldsymbol{x}) + \mu\left(\frac{\partial u_i}{\partial x_k} + \frac{\partial u_k}{\partial x_i}\right)$$

and $(\boldsymbol{u^\epsilon}, p)$ satisfies eq. (2.21) with

$$\sigma_{ij}^\epsilon(\boldsymbol{x}) = -\delta_{ik}p^\epsilon(\boldsymbol{x}) + \mu\left(\frac{\partial u_i^\epsilon}{\partial x_k} + \frac{\partial u_k^\epsilon}{\partial x_i}\right).$$

We note that $\partial\sigma_{ik}(\boldsymbol{x})/\partial x_k = 0$ and $\partial\sigma_{ik}^\epsilon(\boldsymbol{x})/\partial x_k = -f_{0,i}\phi^\epsilon(\boldsymbol{x}-\boldsymbol{y})$ through the use of eqs. (2.20) and (2.21) respectively. From these two equations we find that

$$\begin{aligned} \frac{\partial}{\partial x_k}(u_i^\epsilon\sigma_{ik} - u_i\sigma_{ik}^\epsilon) &= \frac{\partial u_i^\epsilon}{\partial x_k}\sigma_{ik} + u_i^\epsilon\frac{\partial\sigma_{ik}}{\partial x_k} - \frac{\partial u_i}{\partial x_k}\sigma_{ik}^\epsilon - u_i\frac{\partial\sigma_{ik}^\epsilon}{\partial x_k} \\ &= -u_i(-f_{0,i})\phi^\epsilon \\ &= u_j f_{0,j}\phi^\epsilon(\boldsymbol{x}-\boldsymbol{y}) \end{aligned}$$

where we have replaced the summation over i with a summation over j without affecting the result. This forms our version of the Lorentz reciprocal identity We now substitute in eqs. (2.7) and (2.9) to obtain

$$\frac{1}{8\pi\mu}\frac{\partial}{\partial x_k}(S^\epsilon_{ij}f_{0,j}\sigma_{ik} - \mu u_i T^\epsilon_{ijk}f_{0,j}) = u_j f_{0,j}\phi_\epsilon(\boldsymbol{x}-\boldsymbol{y})$$

as $f_{0,j}$ is constant we can take it out of the derivative on the left-hand side and note that it is now arbitrary and as such $\boldsymbol{u}$ and p obey the relation

$$\frac{1}{8\pi\mu}\frac{\partial}{\partial x_k}(S^\epsilon_{ij}\sigma_{ik} - \mu u_i T^\epsilon_{ijk}) = u_j\phi_\epsilon(\boldsymbol{x}-\boldsymbol{y}). \tag{2.22}$$

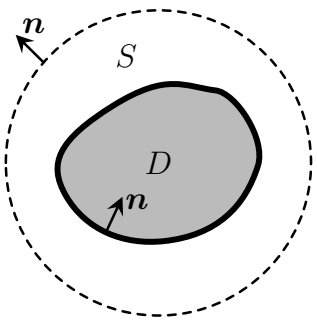


Figure 2: Schematic diagram of the volume used to derive the boundary integral for Stokes flow

Suppose we now let S be the volume between the solid body D and a sphere with a radius such that all of D is contained within. We will denote ∂S to be the surface of S, note that this contains the surface of the sphere and the surface of the solid body ∂D. If we now integrate the above equation over the volume S then we get that

$$\begin{aligned}&\iiint_S \left[\frac{1}{8\pi\mu}\frac{\partial}{\partial x_k}(S^\epsilon_{ij}(\boldsymbol{x},\boldsymbol{y})\,\sigma_{ik}(\boldsymbol{x},\boldsymbol{y}) - \mu u_i(\boldsymbol{x})T^\epsilon_{ijk}(\boldsymbol{x},\boldsymbol{y}))\right] dV(\boldsymbol{x})\\ &= \iiint_S u_j(\boldsymbol{x})\phi_\epsilon(\boldsymbol{x}-\boldsymbol{y})dV(\boldsymbol{x})\end{aligned}$$

Using the divergence theorem on the left-hand side of the equation we get that

$$\frac{1}{8\pi\mu}\iint_{\partial S}\left[S^\epsilon_{ij}(\boldsymbol{x},\boldsymbol{y})\,\sigma_{ik}(\boldsymbol{x},\boldsymbol{y}) - \mu u_i(\boldsymbol{x})T^\epsilon_{ijk}(\boldsymbol{x},\boldsymbol{y})\right] n_k ds(\boldsymbol{x}) = \iiint_S u_j\phi_\epsilon(\boldsymbol{x}-\boldsymbol{y})dV(\boldsymbol{x})$$

where $\boldsymbol{n}$ is the outwards facing unit normal vector of ∂S. If we consider the limit as the radius of the sphere tends to infinity, we find that the only contributions to the left-hand side come from ∂D. We will introduce the traction on the surface of the sphere as $g_i = -\sigma_{ik} n_k$ (as the unit normal points into D)and obtain

$$\begin{aligned} &\frac{1}{8\pi\mu} \iint_{\partial D} S_{ij}^{\epsilon}(\boldsymbol{x}, \boldsymbol{y})\, g_i(\boldsymbol{x}) ds(\boldsymbol{x}) - \frac{1}{8\pi} \iint_{\partial D} u_i(\boldsymbol{x}) T_{ijk}^{\epsilon}(\boldsymbol{x}, \boldsymbol{y})\, n_k(\boldsymbol{x}) ds(\boldsymbol{x}) \\ &= \iiint_S u_j(\boldsymbol{x}) \phi_\epsilon(\boldsymbol{x} - \boldsymbol{y}) dV(\boldsymbol{x}) \end{aligned} \tag{2.23}$$

Considering the fluid inside of the solid body D we realise that the velocity must satisfy the zero-deformation condition

$$\frac{\partial u_i}{\partial k} + \frac{\partial u_k}{\partial i} = 0$$

This gives us that $\sigma_{ik} = -p\delta_{ik}$ inside of D and as such we obtain that for each $j = 1, 2, 3$

$$\iiint_D \frac{\partial}{\partial x_k} \left[S_{ij}^{\epsilon}(\boldsymbol{x}, \boldsymbol{y})\, \sigma_{ik}(\boldsymbol{x}, \boldsymbol{y}) \right] dV(\boldsymbol{x}) = -p \iiint_D \frac{\partial}{\partial x_k} \left[S_{kj}^{\epsilon}(\boldsymbol{x}, \boldsymbol{y}) \right] dV(\boldsymbol{x}) = 0$$

from the incompressibility condition eq. (2.11). If we now integrate eq. (2.22) over D instead of S and use the above integral we have that given $p \neq 0$

$$\begin{aligned} -\frac{1}{8\pi} \iiint_D \frac{\partial}{\partial x_k} \left(u_i(\boldsymbol{x}) T_{ijk}^{\epsilon}(\boldsymbol{x}, \boldsymbol{y}) \right) dV(\boldsymbol{x}) &= \frac{1}{8\pi} \iint_{\partial D} u_i(\boldsymbol{x}) T_{ijk}^{\epsilon}(\boldsymbol{x}, \boldsymbol{y})\, n_k(\boldsymbol{x}) ds(\boldsymbol{x}) \\ &= \iiint_D u_j(\boldsymbol{x}) \phi^{\epsilon}(\boldsymbol{x} - \boldsymbol{y}) dV(\boldsymbol{x}) \end{aligned} \tag{2.24}$$

We again have used the divergence theorem to convert the volume integral to a surface integral with n_k being defined to point into D. Observing that the sum over eqs. (2.23) and (2.24) will give us the integral over $\mathbb{R}^3$ and, using the fact that the velocity is continuous on the boundary ∂D we determine that final boundary integral equation is

$$\iiint_{\mathbb{R}^3} u_j(\boldsymbol{x}) \phi^{\epsilon}(\boldsymbol{x} - \boldsymbol{y})\, dV(\boldsymbol{x}) = -\frac{1}{8\pi\mu} \iint_{\partial D} S_{ij}^{\epsilon}(\boldsymbol{x}, \boldsymbol{y})\, g_i(\boldsymbol{x}) ds(\boldsymbol{x}) \tag{2.25}$$

As the traction $\boldsymbol{g}$ denotes the traction exerted by the fluid on the body it must have

the opposite sign to the stokeslet traction $\boldsymbol{f}$ and as such $\boldsymbol{f} = -\boldsymbol{g}$ where $\boldsymbol{f}$ denotes the force per unit area acting on the fluid. In order to use eq. (2.25) we need to have an explicit equation for $\boldsymbol{u}(\boldsymbol{x})$, as such we need to remove the integral and the cutoff function from the left hand side of eq. (2.25), we do this though the approximation that $\int_{\mathbb{R}^3} u_j(\boldsymbol{x})\phi^\epsilon(\boldsymbol{x}-\boldsymbol{y})\,dV(\boldsymbol{x}) \approx u_j(\boldsymbol{y})$. By taking the approximation as exact we write that

$$u_j(\boldsymbol{y}) = \frac{1}{8\pi\mu}\iint_{\partial D} S_{ij}^\epsilon(\boldsymbol{x},\boldsymbol{y})\, f_i(\boldsymbol{x}) ds(\boldsymbol{x}) \tag{2.26}$$

By using the fact that $S_{ij}^\epsilon(\boldsymbol{x},\boldsymbol{y}) = S_{ji}^\epsilon(\boldsymbol{y},\boldsymbol{x})$ we can write eq. (2.25) as

$$u(\boldsymbol{x})_i = \frac{1}{8\pi\mu}\iint_{\partial D} S_{ij}^\epsilon(\boldsymbol{x},\boldsymbol{y})\, f_j(\boldsymbol{y}) ds(\boldsymbol{y}) \tag{2.27}$$

The analysis of this error for our particular cutoff function can be see in [12] and is found to be of order $\mathcal{O}(\epsilon)$ close to the boundary and $\mathcal{O}(\epsilon^2)$ away from the boundary.

The analytical computation of eq. (2.27) is possible for certain cases allowing for the computation of the fluid velocity given prescribed body forces on the fluid, however, they are often hard or impossible to do by hand. In this case we consider the numerical integration of such problems. By approximating the boundary integral equation with a quadrature rule we obtain a formula for the velocity of the fluid at a point $\boldsymbol{y}$. To do this we consider the integral on the right-hand side of eq. (2.27) as the sum over N stokeslets located on the surface of D.

$$u_i(\boldsymbol{x}) = \frac{1}{8\pi\mu}\sum_{n=1}^{N}\sum_{i=1}^{3} S_{ij}^\epsilon(\boldsymbol{x},\boldsymbol{y}_n)\, f_j(\boldsymbol{y}_n) A_n \tag{2.28}$$

where $f_i(\boldsymbol{y}_n)$ is the ith component of the force on the fluid at the point $\boldsymbol{y}_n$ and A_n is the corresponding quadrature weight of the nth stokeslet.

The discretisation of the integral in eq. (2.27) introduced a discretisation error of $\mathcal{O}(h)$ where h is the average spacing between quadrature points and a quadrature error which is also dependent on ϵ. Analysis done by Cortez, Fauci and Medovikov [12] estimated the quadrature error introduced by this method to be $\mathcal{O}(h^2\epsilon^{-3})$ although more recently

analysis by Gallagher, Choudhuri and Smith [17] refined this estimate to $\mathcal{O}(h^2\epsilon^{-1}) + \mathcal{O}(P\epsilon^{-1/P}h^{1-P})$ for any integer $P > 3$.

2.3 Resistance Problem

As eq. (2.28) can be used to calculate the velocity at any point in the fluid $\boldsymbol{x}$. By forming a set of collocation points $\{\boldsymbol{x}_m\}$ for $m = 1, ..., M$ we form a dense system of $3M$ equations which calculate the velocity at all points in $\{\boldsymbol{x}_m\}$. A common problem in Stokes flow is the resistance problem, where the force distribution, and hence total force and moment, on a body is calculated from a prescribed rigid body motion. By prescribing velocities at each point $\{\boldsymbol{x}_m\}$ form a system of $3M$ equations with $3N$ unknowns $F_j(\boldsymbol{y}_n) := f_j(\boldsymbol{y}_n)A_n$. If $M \neq N$ we either under prescribe or over prescribe the system of equation, so we often choose to place both the collocation and quadrature points at the same set of positions, we will refer to this as the Nyström discretisation [18]. For each of the rigid body motion modes, unit velocity translation along each axis and unit angular velocity rotation about each axis, are calculated we can form the grand resistance matrix A[7]. By the linearity of Stokes equation A relates the force F and moment M to the velocity U and angular velocity V for any rigid body motion.

$$\begin{pmatrix} \boldsymbol{F} \\ \boldsymbol{M} \end{pmatrix} = \underbrace{\begin{pmatrix} A_{FU} & A_{F\Omega} \\ A_{MU} & A_{M\Omega} \end{pmatrix}}_{A} \begin{pmatrix} \boldsymbol{U} \\ \boldsymbol{\Omega} \end{pmatrix}.$$

For example, Stokes Law gives us for a sphere of radius r centred at the origin

$$A = \begin{pmatrix} 6\pi\mu r I & 0 \\ 0 & 8\pi\mu r^3 I \end{pmatrix}$$

where I is the 3×3 identity matrix and 0 is a 3×3 matrix of zeros. The formulation and computation of this systems promotes the use of formulating the problem in terms of a Vector matrix system where we can make use of highly optimised BLAS (Basic Linear

Algebra Subprograms) and LAPACK (Linear Algebra PACKage) packages over direct computations in the code. This allows for the software to use the latest in hardware and software optimisations such as improved multicore algorithms and across multiple types of hardware with no changes to the higher-level code.

2.4 Matrix formulation

The computation of eq. (2.28) for a set of collocation points $\{\boldsymbol{x}_m\}$ for $m = 1, ..., M$ can be seen as a can be viewed as as the product of a dense $3M \times 3N$ matrix with a $3N \times 1$ column vector where M is the number of collocation points and N the number of stokeslet points. By taking the Nyström discretisation where we choose $\{\boldsymbol{x}_m\} = \{\boldsymbol{y}_n\}$ the stokeslet matrix therefore becomes a $3N \times 3N$ matrix where the velocity can be calculated from a set of prescribed traction $\{\boldsymbol{F}(\boldsymbol{x}_n)\}$.

$$u_i\left(\boldsymbol{x}_m\right) = \frac{1}{8\pi\mu} \sum_{n=1}^{N} \sum_{i=1}^{3} S_{ij}^{\epsilon}\left(\boldsymbol{x}_m, \boldsymbol{x}_n\right) F_j(\boldsymbol{x}_n) \tag{2.29}$$

where $n = 1, \ldots, N$, $m = 1, \ldots, N$ and $j = 1, 2, 3$. The computation of this matrix vector product can be written as

$$\underline{U} = A\underline{F} \tag{2.30}$$

where $\underline{U}$ is a $3N \times 1$ column vector, A is a $3N \times 3N$ matrix and $\underline{F}$ is a $3N \times 1$. In order to form an equation of this form we choose to write

$$\underline{U} = [u_1(\boldsymbol{x}_1),\ u_2(\boldsymbol{x}_1),\ u_3(\boldsymbol{x}_1),\ u_1(\boldsymbol{x}_2),\ u_2(\boldsymbol{x}_2),\ u_3(\boldsymbol{x}_2),\ \ldots\ u_1(\boldsymbol{x}_M),\ u_2(\boldsymbol{x}_M),\ u_3(\boldsymbol{x}_N)]^T u$$

and

$$\underline{F} = [f_1(\boldsymbol{x}_1)A_1,\ f_2(\boldsymbol{x}_1)A_1,\ f_3(\boldsymbol{x}_1)A_1,\ \ldots\ f_1(\boldsymbol{x}_N)A_N,\ f_2(\boldsymbol{x}_N)A_N,\ f_3(\boldsymbol{x}_N)A_N]^T$$

This gives the form of A as

$$A = \begin{bmatrix} S^\epsilon_{11}(\boldsymbol{x}_1,\boldsymbol{x}_1) & S^\epsilon_{12}(\boldsymbol{x}_1,\boldsymbol{x}_1) & S^\epsilon_{13}(\boldsymbol{x}_1,\boldsymbol{x}_1) & \dots & S^\epsilon_{11}(\boldsymbol{x}_N,\boldsymbol{x}_1) & S^\epsilon_{13}(\boldsymbol{x}_N,\boldsymbol{x}_1) & S^\epsilon_{13}(\boldsymbol{x}_N,\boldsymbol{x}_1) \\ S^\epsilon_{21}(\boldsymbol{x}_1,\boldsymbol{x}_1) & S^\epsilon_{22}(\boldsymbol{x}_1,\boldsymbol{x}_1) & S^\epsilon_{23}(\boldsymbol{x}_1,\boldsymbol{x}_1) & \dots & S^\epsilon_{21}(\boldsymbol{x}_N,\boldsymbol{x}_1) & S^\epsilon_{22}(\boldsymbol{x}_N,\boldsymbol{x}_1) & S^\epsilon_{23}(\boldsymbol{x}_N,\boldsymbol{x}_1) \\ S^\epsilon_{31}(\boldsymbol{x}_1,\boldsymbol{x}_1) & S^\epsilon_{32}(\boldsymbol{x}_1,\boldsymbol{x}_1) & S^\epsilon_{33}(\boldsymbol{x}_1,\boldsymbol{x}_1) & \dots & S^\epsilon_{31}(\boldsymbol{x}_N,\boldsymbol{x}_1) & S^\epsilon_{32}(\boldsymbol{x}_N,\boldsymbol{x}_1) & S^\epsilon_{33}(\boldsymbol{x}_N,\boldsymbol{x}_1) \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ S^\epsilon_{11}(\boldsymbol{x}_1,\boldsymbol{x}_N) & S^\epsilon_{12}(\boldsymbol{x}_1,\boldsymbol{x}_N) & S^\epsilon_{13}(\boldsymbol{x}_1,\boldsymbol{x}_N) & \dots & S^\epsilon_{11}(\boldsymbol{x}_N,\boldsymbol{x}_N) & S^\epsilon_{12}(\boldsymbol{x}_N,\boldsymbol{x}_N) & S^\epsilon_{13}(\boldsymbol{x}_N,\boldsymbol{x}_N) \\ S^\epsilon_{21}(\boldsymbol{x}_1,\boldsymbol{x}_N) & S^\epsilon_{22}(\boldsymbol{x}_1,\boldsymbol{x}_N) & S^\epsilon_{23}(\boldsymbol{x}_1,\boldsymbol{x}_N) & \dots & S^\epsilon_{21}(\boldsymbol{x}_N,\boldsymbol{x}_N) & S^\epsilon_{22}(\boldsymbol{x}_N,\boldsymbol{x}_N) & S^\epsilon_{23}(\boldsymbol{x}_N,\boldsymbol{x}_N) \\ S^\epsilon_{31}(\boldsymbol{x}_1,\boldsymbol{x}_N) & S^\epsilon_{32}(\boldsymbol{x}_1,\boldsymbol{x}_N) & S^\epsilon_{33}(\boldsymbol{x}_1,\boldsymbol{x}_N) & \dots & S^\epsilon_{31}(\boldsymbol{x}_N,\boldsymbol{x}_N) & S^\epsilon_{32}(\boldsymbol{x}_N,\boldsymbol{x}_N) & S^\epsilon_{33}(\boldsymbol{x}_N,\boldsymbol{x}_N) \end{bmatrix}$$

We do note that this particular choice in the structure of U and F is not unique and other literature may use different arrangements however they will all produce the same results. The computation of the velocities at the set of points $\{\boldsymbol{x}_n\}$ through the direct computation of the full matrix-vector product as the direct product. Assembly of the stokeslet Matrix requires $9N^2$ function evaluations followed by $\mathcal{O}(N^2)$ operations to compute the vector product. The formulation of the direct product also allows for easy application to resistance problem where $\underline{F} = A^{-1}\underline{U}$, This problem can either be solved through the inversion of the matrix A or in our implementation the use of Matlab's mldivide (\) function which both take $\mathcal{O}(N^3)$ operations to compute the solution. For larger systems where the use of mldivide becomes too computationally costly, either from time or memory, and the solution can be solved using the Generalized Minimal RESidual (GMRES) method (See appendix A) [19, 20]. GMRES can perform the inverse in at worst $\mathcal{O}(N^3)$ however will more often converge to the solutions at high level of accuracy in significantly less iterations, particularly when an appropriate preconditiner is used.

3 NEAREST

When we look at the over all error estimation for the method of regularised stokeslets we find the total error to be $\mathcal{O}(\epsilon) + \mathcal{O}(h) + \mathcal{O}(h^2\epsilon^{-1}) + \mathcal{O}(P\epsilon^{-1/P}h^{1-P})$. In order to reduce the overall error we must reduce ϵ such that the regularisation error decreases, however doing this necessitates the reduction of h the average quadrature spacing to keep the discretisation error at a similar size. In order to still fully cover our boundary we must therefore increase the number of quadrature points used in the calculation, which will increase the computation needed to both perform the direct vector matrix product or the matrix inversion required for resistance problems.

Methods such as the Boundary Element Method (BEM) [21] attempt to fix this problem by decoupling the quadrature and traction discretisation, allowing for the stokeslet kernel to be evaluate in the most appropriate means without affecting the overall number of degrees of freedom. This work has allowed for larger systems to be explored, such as model cilia-driven flows [22, 23]. Computational experiments done show that BEM methods can take up to 3 orders of magnitude less processing time and memory [21] when compared to the Nyström method.

The NEAREST method proposed by Smith [24] aims to decouple the quadrature and traction discretisation's while still preserving the simplicity of the low order quadrature rule used in eq. (2.29). When analysing the values for both the kernel $\{S_{ij}^{\epsilon}(\boldsymbol{y}, \boldsymbol{x})\}$ and force values $\{\boldsymbol{f}(\boldsymbol{x})\}$ obtained in the method of regularised Stokeslets we find that in most cases the kernel changes much more rapidly than over the surface of the sphere. In order to capture these changes with the least amount of degrees of freedom, we propose to use a finer quadrature rule with more points in order to capture the rapid changes in the kernel and a coarser discretisation to capture the slower changes in $\boldsymbol{f}(\boldsymbol{x})$. In order to achieve this goal we will consider two surface discretisations of ∂D, one coarse and one fine denoted by $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ and $\{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_Q\}$ respectively and it is assumed that $N \leq Q$. The coarse discretisation will be our set of points in which we will discretise the force and our fine discretisation will serve as our set of quadrature points for the kernel. In order to map our fine quadrature discretisation onto our set of coarse force point will

then use nearest-neighbour interpolation defined by $\mathcal{N} : \{1, \ldots, Q\} \to \{1, \ldots, N\}$ where

$$\mathcal{N}(q) := \underset{n=1,\ldots,N}{\operatorname{argmin}} |\boldsymbol{x}_n - \boldsymbol{X}_q| \tag{3.1}$$

This allows us to define eq. (2.27) as

$$\begin{aligned} u_i(\boldsymbol{y}) &= \frac{1}{8\pi\mu} \int_{\partial D} S_{ij}^{\epsilon}(\boldsymbol{y}, \boldsymbol{X}_q) \, f_j(\boldsymbol{X}_q) ds(\boldsymbol{X}_q) \\ &= \frac{1}{8\pi\mu} \int_{\partial D} S_{ij}^{\epsilon}(\boldsymbol{y}, \boldsymbol{X}_q) \, f_j(\mathcal{N}(q)) ds(\mathcal{N}(q)) \\ &= \frac{1}{8\pi\mu} \sum_{q=1}^{Q} S_{ij}^{\epsilon}(\boldsymbol{y}, \boldsymbol{X}_q) \, f_j[\mathcal{N}(q)] A[\mathcal{N}(q)] \end{aligned} \tag{3.2}$$

where the final line is the discretisation of the integral with the fine quadrature points. In order to be able to use this interpolation we need to express the operator $\mathcal{N}(q)$ in its matrix form, a $Q \times N$ sparse matrix defined by

$$\nu[q, \hat{n}] = \begin{cases} 1 & \text{if } \hat{n} = \underset{n=1,\ldots,N}{\operatorname{argmin}} |x_n - X_q|, \\ 0 & \text{otherwise.} \end{cases} \tag{3.3}$$

This provides a direct mapping for each point from our fine quadrature set to the coarse force set. When considering large quadrature sets we can find cases where a fine quadrature points is equidistant to two or more force points. Under the current form of $\nu[q, \hat{n}]$ the quadrature point is mapped to the first force point in the set. This means that the same set of force and quadrature points can have different results depending on how the both set are arranged. If we instead weight the quadrature point evenly between all equidistant force points we can eliminate this problem, This can be implemented by replacing the first condition when $\hat{n} = \underset{n=1,\ldots,N}{\operatorname{argmin}} |x_n - X_q|$ with $1/k$ where k is the number of points in $\{x_n\}$ which are equidistant to $\{X_q\}$. This allows a fine point to be mapped to multiple coarse points if they are equidistant, with its contribution evenly distributed

between them. The sparse matrix form of $\mathcal{N}(q)$ allows us to express eq. (3.2) as

$$u_i(\boldsymbol{y}) = \frac{1}{8\pi\mu} \sum_{n=1}^{q} f_j(\boldsymbol{x}_n) A_n \sum_{q=1}^{Q} S_{ij}^{\epsilon}\left(\boldsymbol{y}, \boldsymbol{X}_q\right) \nu[q, n] \tag{3.4}$$

As we considered in the direct product eq. (2.30) we can calculate the velocity at a set of collocation points $\{x_n\}$ with $n = 1, \ldots, N$ as a matrix-vector product. If we write $\underline{U}$ and $\underline{F}$ as before, the stokeslet matrix A however now takes the form of a $3N \times 3Q$ matrix. The nearest neighbour matrix $\nu[q, \hat{n}]$ can be used by taking the Kronecker product of ν with the 3×3 identity matrix ($\Pi = \nu \otimes \mathbb{1}_3$) to map each axis at every point. This allows use to form the system

$$\underline{U} = A \cdot \Pi \cdot \underline{F}. \tag{3.5}$$

The NEAREST method allows for both the forward matrix product to compute the velocity of the fluid as well as solving the resistance problem through $\underline{F} = (A \cdot \Pi)^{-1}\underline{U}$. The NEAREST method is not directly aimed at compeating directly with BEM methods in performance it does offer a easy mesh-free adaptation to the standard Nyström method which allows for the computation of larger scale systems than that of the standard Direct product.

By introducing the second discretisation we have now changed the error estimations for the method, we still keep the regularisation error of $\mathcal{O}(\epsilon)$ and traction discretisation error $\mathcal{O}(h_f)$. The quadrature error now depends on if the traction points are contained (within distance $O(\epsilon)$ or closer) in the quadrature set. We introduce the distances h_q and h_f to be the length scales associated with the quadrature and force discretisation respectively. In the contained case we obtain a quadrature error of $\mathcal{O}(\epsilon^{-1}h_q^2) + \mathcal{O}(P\epsilon^{-1/P}h_q^{1-1/P})$ for all integer $P > 3$ and for the disjoint case the error decouples from ϵ to be $\mathcal{O}(h_q^3\delta^2) + \mathcal{O}(Ph_q^{1-2/P})$ where δ is the closest distance between quadrature and traction points [17].

3.1 Comparison of NEAREST to Nyström method

4 Kernel-Independent Fast Multipole Method

If we consider a many body problem formed of N potential/quadrature points $\{\boldsymbol{y}_n\}$ each with an associated potential/force $\{\boldsymbol{f}_n\}$ that we will evaluate on M collocation points $\{\boldsymbol{x}_m\}$ then we can compute the velocity at each $\boldsymbol{x}_m$ where $m = 1, 2, \ldots, M$ through

$$\boldsymbol{u}(\boldsymbol{x}) = \sum_{i=1}^{N} \Phi(\boldsymbol{x}, \boldsymbol{y}_i)\boldsymbol{f}_i$$

where Φ is the green's function associated with the underlying partial differential equation governing the dynamics of the system. In particular, $\Phi = S^\epsilon$ is our kernel for the case of the method of regularised Stokeslets where we compute the velocity of the fluid at a set of collocation points based on a set of body forces defined on solid boundaries within the fluid. This can be written as a matrix-vector product as described in eq. (2.30). This direct solution has a computational complexity of $\mathcal{O}(NM)$ which for either a large number of collocation points or quadrature points quickly becomes prohibitive due. In order to compute large systems we need to look at faster methods to approximate this vector-matrix product. One such set of methods is the familiy of fast multipole method's which can approximate eq. (2.30) with complexity $\mathcal{O}(N + M)$. The standard fast multipole method (FMM) algorithm uses multipole expansion's of the kernel about the quadrature points in order to approximate long-range interaction between distant quadrature and target points [25, 26, 27, 28]. This method proves time-consuming to find for particular kernels such as regularised Stokeslets where the corresponding expansions first need to be computed by hand, it also means that the entire program is kernel dependent and any changes to the kernel need a complete recalculation of the expansions employed. Instead we will look at an adaption to the FMM algorithm to create a kernel independent fast multipole algorithm (KIFMM) [29, 30, 31, 32]. Using equivalent potentials instead of multipole expansions we can approximate long-range interactions using only kernel evaluations without computing every particle to particle interaction. This dramatically

reduces the number of kernel evaluations compared to computation by the direct solution.

4.1 Hierarchical decomposition of the computational domain

Both FMM and KIFMM are based on the Hierarchical decomposition of the computational domain. We let $\mathcal{D}$ be the computational domain, which we define to be a cube in $\mathcal{R}^3$ such that it encompasses all points in $\{\boldsymbol{y}_n\}$ and $\{\boldsymbol{x}_m\}$. We typically define this to be the smallest possible cube which includes all the points. The FMM method builds an Octree structure of cubes in 3D with the cube $\mathcal{D}$ as the root node. We will label this node the Level 0 partition where no subdivision had occurred. Each subsequent division is made of 8 equal-sized cubes each with a side length of half of the cube above. For each level, we subdivide a node into its 8 children if the number of source points or target points within the cube is greater than some threshold $s \geq 1$. For efficiency only points of the type which exceed the threshold are moved from to the child nodes. This subdivision creates a new level $L+1$ containing the children of all subdivided nodes in level L. We refer to the node on level L as the parent of nodes on level $L+1$. This process is repeated until all cubes have less than s source or target points contained within them which we will refer to as leaf nodes.

An example of a simple decomposition can be seen in fig. 3. In the non-adaptive case if any node has greater than s source points then all nodes on that level are subdivided creating 8^L new nodes. We might choose to implement a tree of this form when the potential distribution is uniform as it reduces the types and complexity of the interactions described later but we will only consider the adaptive case in this discussion as it is more efficient in the non-uniform cases we will be considering.

4.1.1 Interaction Lists

For any node B in the Octree, we define its relation to neighbouring node through 4 interaction lists I_U^B, I_V^B, I_X^B and I_W^B.

- The list I_U^B contains B itself and all leaf nodes which are adjacent to B. The U list is empty for any non-leaf nodes.

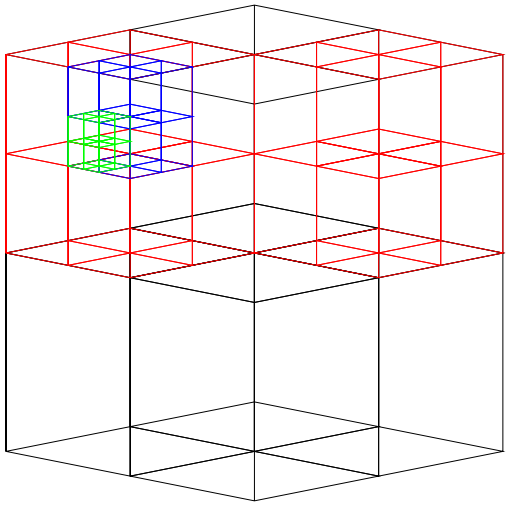


Figure 3: A 3D example of a 4 level Octree generated on a set of random points (Points not shown). Black, red, blue and green cubes represents nodes on Level 1, 2, 3 and 4 of the Octree respectively.

- The list I_V^B contains all nodes which are children of the nodes which are neighbours of B's parent. This means all nodes in I_U^B are of the same level as B.

- If B is a leaf node then the interaction list I_W^B contains all descendants of B's neighbours whose parents are adjacent to B but are not adjacent to B themselves. In order to not compute the effect of the node twice we only include nodes A, where A and all its decedents, obey the condition above. If B is not a leaf node then I_W^B is empty.

- The list I_X^B contain all nodes A in which B appears in the interaction list I_W^A.

In the figure 4 we show the 2D interactions list for a node B for a random arrangement of nodes, note that these might not all be leaf nodes. These lists contain all the nodes from which we need to consider the contributions as well as the node's parent. In order to see how these nodes interact with each other, we will define the near and far-field of each node. We will denote the near field of a node B to be $\mathcal{N}(B)$ which is the set of all nodes in I_U^B and I_X^B, and the far-field $\mathcal{F}(B)$ to be all remaining nodes. While geometrically

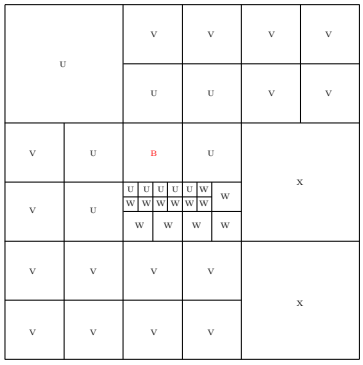


Figure 4: A 2D example of the interaction lists I_U^B, I_V^B, I_X^B and I_W^B for a node B

these definitions may be convoluted with nodes in the far-field being geometrically closer than those in the near field they provide definitions for which nodes need to be computed directly or through the use of equivalent surfaces.

4.2 Equivalent surfaces

The KIFMM makes use of two equivalent surfaces for each node in the Octree to approximate the effect between source and source points inside and outside the node. The upwards equivalent surface is taken to be the boundary of the node B (denoted Ω^U) and the downwards equivalent surface to be the surface of a cube of width three times larger and centred on the node B (denoted Ω^D) as illustrated in fig. 5 [29]. Upon both of these surfaces we denote equivalent potential densities $\boldsymbol{f}^{BU}$ and $\boldsymbol{f}^{BD}$ for the upwards and downwards equivalent surface's of B respectively. The upward equivalent surface is used to approximate the effect of all source points contained within the node and its decedents [31]. As such we can say that for every point $\boldsymbol{x} \in \mathcal{F}(B)$ the velocity induced by that of the equivalent surface is the same as that induced by all source points in B, this gives us

$$\forall\, \boldsymbol{x} \in \mathcal{F}(B): \quad \int_{\Omega^U} S^{\epsilon}(\boldsymbol{x}, \boldsymbol{y}) \boldsymbol{f}^{BU}(\boldsymbol{y}) d\boldsymbol{y} = \sum_{\boldsymbol{y}_n \in B} S^{\epsilon}\left(\boldsymbol{x}, \boldsymbol{y}_n\right) \boldsymbol{f}_n \tag{4.1}$$

The same can be applied to the downwards equivalent surface where instead the contributions are obtained from source points in $\mathcal{F}(B)$, we can say that for any point in B the velocity induced by the downwards surface equals that induced by the source

points in $\mathcal{F}(B)$, which again gives us that

$$\forall\, \boldsymbol{x} \in B: \quad \int_{\Omega^D} S^{\epsilon}(\boldsymbol{x},\boldsymbol{y})\boldsymbol{f}^{BD}(\boldsymbol{y})d\boldsymbol{y} = \sum_{\boldsymbol{y}_n \in \mathcal{F}(B)} S^{\epsilon}\left(\boldsymbol{x},\boldsymbol{y}_n\right)\boldsymbol{f}_n \tag{4.2}$$

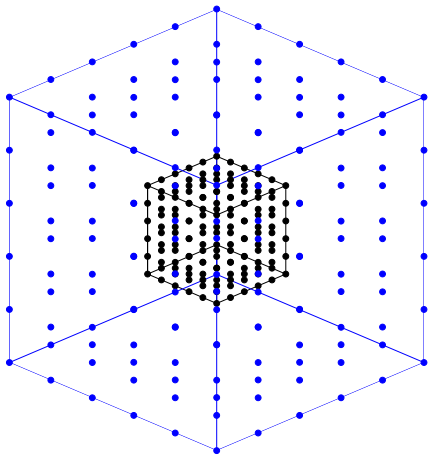


Figure 5: A diagram representing the upwards and downwards equivalent surface for an arbitrary node with bounds given in black. Both surfaces are discredited by a $6 \times 6 \times 6$ set of quadrature points. The upwards equivalent surface (black) lies on the boundary of the node while the downward equivalent surface (blue) is a cube with sides 3 times larger and centred on the node.

In order to be able to use these equivalent potentials densities in our method, we form a quadrature rule over an equally spaced uniform Cartesian grid with N_q quadrature point. While the number of quadrature points in the upwards and downwards equivalent surfaces does not need to be the same, for convenience we will take this to be the case. We will indicate the quadrature points the upwards equivalent potential for a node B with $\{\boldsymbol{q}_m^{BU}\}$ for $m = 1, \ldots, N_q$ and $\{\boldsymbol{q}_m^{BD}\}$ for $m = 1, \ldots, N_q$ indicating the positions of the quadrature points for the downwards equivalent potential. Now each of these quadrature points has a corresponding equivalent potential denoted by $\{\boldsymbol{f}^{BU}(\boldsymbol{q}_m^{BU})\}$ and $\{\boldsymbol{f}^{BD}(\boldsymbol{q}_m^{BD})\}$ for the upward and downward equivalent potentials respectively. Writing the left-hand side of eqs. (4.1) and (4.2) using this quadrature rule we obtain that

$$\forall\, \boldsymbol{x} \in \mathcal{F}(B): \quad \boldsymbol{u}(\boldsymbol{x}) = \frac{1}{8\pi\mu}\sum_{m=1}^{N_q} A_m^{BU} S^{\epsilon}\left(\boldsymbol{x},\boldsymbol{q}_m^{BU}\right)\boldsymbol{f}^{BU}\left(\boldsymbol{q}_m^{BU}\right) \tag{4.3}$$

and the contribution to the velocity $\boldsymbol{u}(\boldsymbol{x})$ for $\boldsymbol{x} \in B$ from source points in $\mathcal{F}(B)$ as

$$\forall\, \boldsymbol{x} \in B: \quad \boldsymbol{u}(\boldsymbol{x}) = \frac{1}{8\pi\mu} \sum_{m=1}^{N_q} A_m^{BD} S^{\epsilon}\left(\boldsymbol{x}, \boldsymbol{q}_m^{BD}\right) \boldsymbol{f}^{BD}\left(\boldsymbol{q}_m^{BD}\right) \tag{4.4}$$

Where $\{A_m^{BU}\}$ and $\{A_m^{BD}\}$ are the corresponding quadrature weights of the upwards and downward equivalent points respectively. We will not explicitly calculate these weights and will take them to be part of their corresponding equivalent potential. The advantage of using eqs. (4.3) and (4.4) is to avoid the direct computation of source points and target points and instead use the quadrature points as an approximation of the particles within the box.

We note that the downward equivalent surface lies in the $\mathcal{F}(B)$ and the downward equivalent surface lies in B and hence we obtain the following systems of equations

$$\sum_{m=1}^{N_q} A_m^{BU} S^{\epsilon}\left(\boldsymbol{q}_k^{BD}, \boldsymbol{q}_m^{BU}\right) \boldsymbol{f}^{BU}\left(\boldsymbol{q}_m^{BU}\right) = \sum_{\boldsymbol{y}_n \in B} S^{\epsilon}\left(\boldsymbol{q}_k^{BD}, \boldsymbol{y}_n\right) \boldsymbol{f}_n A(\boldsymbol{y}_n), \quad k = 1, \ldots, N_q \tag{4.5}$$

and

$$\sum_{m=1}^{N_q} A_m^{BD} S^{\epsilon}\left(\boldsymbol{q}_k^{BU}, \boldsymbol{q}_m^{BD}\right) \boldsymbol{f}^{BD}\left(\boldsymbol{q}_m^{BD}\right) = \sum_{\boldsymbol{y}_n \in \mathcal{F}(B)} S^{\epsilon}\left(\boldsymbol{q}_k^{BU}, \boldsymbol{y}_n\right) \boldsymbol{f}_n A(\boldsymbol{y}_n), \quad k = 1, \ldots, N_q \tag{4.6}$$

We will use eqs. (4.5) and (4.6) to construct the upwards and downwards potential of all nodes through the approximation the left-hand sides of the equation.

4.3 Evaluation

The Fast Multipole method is based on two passes, one up the tree in order to construct the upwards equivalent surfaces from source potentials and a further downward pass where we construct the downward equivalent potentials before finally evaluating the velocities at the target points.

4.3.1 Upward Pass

The upwards pass involves the postorder traversal (see appendix B) of the tree from the smallest node up to the root node and solving eq. (4.5) for each node. For each leaf node, we can evaluate the right-hand side of eq. (4.5) directly. Now for each non-leaf node B in the tree instead of summing over all source points in the decedents of B we can approximate its upwards equivalent surface from its children. We denote the children of B as C_l^B where $l = 1, ..., 8$. As we have traversed the tree in post-order we know $\{\boldsymbol{f}^{C_lU}(\boldsymbol{q}_m^{C_lU})\}$ for $l = 1, \ldots, 8$ and $m = 1, \ldots, N_q$. Since $\{\boldsymbol{q}_m^{C_lU}\}$ lies within $\{\boldsymbol{q}_m^{BU}\}$ we have that the right-hand side of eq. (4.5) can be approximated as

$$\sum_{l=1}^{8} \sum_{m=1}^{N_q} A_m^{C_lU} S^{\epsilon} \left(\boldsymbol{q}_k^{BD}, \boldsymbol{q}_m^{C_lU}\right) \boldsymbol{f}^{C_lU} \left(\boldsymbol{q}_m^{C_lU}\right), \quad k = 1, \ldots, N_q \tag{4.7}$$

We refer to this as a multipole to multipole translation (M2M). This method of approximating the parents equivalent potential from its children is significantly more efficient, as, provided that the number of quadrature points is smaller than the capacity of the node s, the number of kernel evaluations is significantly reduced compared to if the equivalent surface was computed directly using the right hand side of eq. (4.5). We note that in both summations we have computed the velocities at the downward equivalent surface, this ensures the existence of the $\{\boldsymbol{f}^{BU}(\boldsymbol{q}_m^{BU})\}$. In order to obtain these values, we simply solve the linear system created by eq. (4.5). This gives us a set of equivalent potentials for each node in the tree which approximates the effect of all source points contained within that nodes region.

4.3.2 Downwards Pass

The downward pass follows a similar structure to that of the upwards pass however we now traverse down the octree using a pre-order traversal (see appendix B). We do not need to consider the root node on level 0 so we start our traversal on level 1. We will now be considering equation eq. (4.6) and approximating the right-hand side before solving the linear system to obtain the downward equivalent potentials for all considered nodes.

For each non-root node B we need to consider the effect of source points in the far-field. In order to consider these effects, we define the Multipole to Local translation (M2L) and the Local to Local translation (L2L) translation. The M2L translation computes the contribution to the downward equivalent potential to B for a node A in $\mathcal{F}(B)$. As points, $\{\boldsymbol{q}_m^{BU}\}$ are in the $\mathcal{F}(A)$ we can consider the contribution to the right-hand side as approximately

$$\sum_{m=1}^{N_q} A_m^{AU} S^{\epsilon}\left(\boldsymbol{q}_k^{BU}, \boldsymbol{q}_m^{AU}\right) \boldsymbol{f}^{AU}\left(\boldsymbol{q}_m^{AU}\right), \quad k = 1, \ldots, N_q \tag{4.8}$$

We use eq. (4.8) to compute the effect of all nodes in the interaction list I_V^B.The effect of other nodes in $\mathcal{F}(B)$ are computed through the M2L transition where we pass information of distant source points from parent to child. As we are in preorder traversal, we have already computed the downward equivalent potential of the parent of B. We will call the parent P and it's downward equivalent potential $\{\boldsymbol{f}^{PD}(\boldsymbol{q}_m^{PD})\}$. As B is by definition inside of P we know that $\{\boldsymbol{q}_m^{BU}\}$ is in $\mathcal{N}(B)$ so we can use eq. (4.4) to calculate the contribution to the right-hand side of eq. (4.6) as

$$\sum_{m=1}^{N_q} A_m^{PD} S^{\epsilon}\left(\boldsymbol{q}_k^{BU}, \boldsymbol{q}_m^{PD}\right) \boldsymbol{f}^{PD}\left(\boldsymbol{q}_m^{PD}\right), \quad k = 1, \ldots, N_q \tag{4.9}$$

These two summations approximate the contributions of source points in $\mathcal{F}(B)$ however, we need to consider the near field contributions on $\{\boldsymbol{f}^{BD}(\boldsymbol{q}_m^{BD})\}$. For any non leaf node we have that $I_U^B = I_W^B = \emptyset$ where $\emptyset$ is the empty set. This means we only need to consider I_X^B in our calculation of the downward equivalent potential $\{\boldsymbol{f}^{\boldsymbol{BD}}(\boldsymbol{q}_m^{BD})\}$. We will consider the effect of nodes in I_U^B and I_W^B when we compute the velocity at the target points in the next step. We could use the M2L transition eq. (4.8) to approximate the forces however as any node $A \in I_X^B$ is by definition a leaf node and in $\mathcal{N}(B)$ we will consider the effects of its source points directly through the summation

$$\sum_{A_i \in I_X^B} \sum_{\boldsymbol{x}_{\mathbf{0}n} \in A_i} S^{\epsilon}\left(\boldsymbol{q}_k^{BU}, \boldsymbol{x}_{\mathbf{0}n}\right) \boldsymbol{f}_{\mathbf{0}n}, \quad k = 1, \ldots, N_q \tag{4.10}$$

Having obtained approximations for the right-hand side of eq. (4.6) we solve the linear

system to find the values of $\{\boldsymbol{f}^{BD}(\boldsymbol{q}_m^{BD})\}$ at the quadrature points.

Now we consider the leaf nodes of the system where we solve to find the velocity of the fluid at the target points. We can consider the leaf nodes in any order as the calculations for each leaf node are independent of each other even though they may lie on different levels of the Octree. In order to compute the velocity at the target points, we need to consider all points in $\mathcal{N}(B)$ and $\mathcal{F}(B)$. In the computation of $\{\boldsymbol{f}^{BD}(\boldsymbol{q}_m^{BD})\}$ we approximated the effect of points in I_V^B and I_X^B and source points outside of the interaction lists of node B. We can compute this effect though eq. (4.4), we then need to consider the remaining source points in I_U^B and I_W^B. We compute the effect of nodes in I_W^B by writing the right-hand side of eq. (4.3) as

$$\frac{1}{8\pi\mu}\sum_{m=1}^{N_q} A_m^{AU} S^{\epsilon}\left(\boldsymbol{x}, \boldsymbol{q}_m^{AU}\right) \boldsymbol{f}^{AU}\left(\boldsymbol{q}_m^{AU}\right)$$

for each node A in I_W^B. We do this as for any node $A \in I_W^B$ as we know that $B \in I_X^A \in \mathcal{F}(A)$. We compute the effects of sources in I_U^B which we do through

$$\boldsymbol{u}(\boldsymbol{x}) = \sum_{\boldsymbol{y}_n \in A} S^{\epsilon}(\boldsymbol{x}, \boldsymbol{y}_n)\boldsymbol{f}_n \tag{4.11}$$

where $A \in I_U^B$. After summing these contributions together we have constructed an approximation for the velocities at all target points considering the effects of all source points.

4.4 Optimisation

While Kernel independent fast multipole proves increadably fast at the computation of matrix vector products for radially dependent kernals we can optimise it computation in order to exploite its efficiency even more and speed up the computation of the system. In particular we will look at ways in which we can optimise the multipole to local translation (eq. (4.8)) which compute the interaction between nodes and dense particle to particle interactions which form the majority of the computation in the KIFMM method. When

looking at using the fast multipole method with iterative solvers such as GMRES to solve resistance and mobility problems we note that our points derived from the discretisation of the boundary intergral equation remain fixed. This leads to the kernel evauations needed within the fast multipole method to

4.4.1 parallelisation

The nature of Octree traversal lends itself to easy parallelisation, in both tree traversal we need either to have computed the upwards equivilent surface for the upwards pass or the computation of the parents downward equivilent surface. This means we do not need to follow post and pre order traversal exsactly as long as the required nodes are already computed. Due to the way in which the Octree is structured we know that all children are in the level below their parent so in both traversal we can compute all nodes on the same level in parallel.

4.5 Comparison of KIFMM with the Direct Method

5 NEAREST KIFMM Hybrid

The advantage of the KIFMM method comes from its fast approximation of distant source points, while Nearest leverages the differences in the rate of change of the single layer stokeslet kernel to approximate the kernel across the whole domain with less source points. If these methods were to be combined it would allow for even more efficient approximation of source points in the far field and an improvement to near field calculations. Given that the reduction in source points is enough to create a smaller Octree, particularly when a non adaptive tree is used, the KIFMM method should be able to compute the matrix problem in a shorter period of time. The adaption of the KIFMM method to use the nearest neighbour interpolation is a simple but fundamental change to the algorithm. When creating the Octree decomposition of the domain we form the tree based on target points and coarse source discretization as we would normally do, We then also generate the Nearest-Neighbour interpolation matrix ν defined in eq. (3.3) between the coarse

source points and the fine quadrature rule.

In the upwards pass when calculating the upwards equivalent potential we replace the right hand side of eq. (4.5) with a nearest neighbour interpolation

$$\sum_{\boldsymbol{y}_n \in B} \boldsymbol{f}_n(\boldsymbol{y}_n) \sum_{q=1}^{Q^*} S^{\epsilon}\left(\boldsymbol{q}_k^{BD}, \boldsymbol{X}_q\right) \nu^*[q,n]$$

We define $\nu^*[q,n]$ as the subset of the full Nearest-neighbour matrix $\nu[q,n]$ corresponding to the map between points $\boldsymbol{X}_q$ and $\boldsymbol{y}_n \in B$. Q^* is the number of fine quadrature points which map to $\boldsymbol{y}_n \in B$. This method provided a accurate approximation of the source points onto the upward equivalent surfaces. The use of Nearest-Neighbour interpolation in both the M2M translation (eq. (4.7)) and the M2L translation (eq. (4.8)) is not considered due to the need to compute the Nearest-Neighbour matrix between all boxes. While the interactions between boxes are computed during the Octree generation and the matrices could be precomputed, the number and size of these matrices make it inefficient to compute before hand, particularly when nodes with small number's of source points are considered. In the downwards pass we can also implement Nearest-Neighbour interpolation when looking at the interaction lists of a node B, both I_X^B and I_U^B consider point to point interactions directly due to them being in $\mathcal{N}(B)$. These translations are done through eqs. (4.10) and (4.11) respectively. We now replace The right hand side with

$$\sum_{A_i \in I_X^B} \sum_{\boldsymbol{y}_n \in A_i} \boldsymbol{f}_n \sum_{q=1}^{Q^*} S^{\epsilon}\left(\boldsymbol{q}_k^{BU}, \boldsymbol{X}_q\right) \nu^*[q,n], \quad k = 1, \ldots, N_q$$

and

$$\sum_{\boldsymbol{y}_n \in A} \boldsymbol{f}_n \sum_{q=1}^{Q^*} S^{\epsilon}(\boldsymbol{x}, \boldsymbol{X}_q) \nu^*[q,n]$$

respectively. Where $\nu^*[q,n]$ again corresponds to a slice of $\nu[q,n]$ which contain respective map between source points and fine quadrature points in each summation.

5.1 Nearest Comparison

6 Mobility Problems

The resistance problem is another common use of regularised stokes let where we obtain the rigid body motion prescribed force and moment. We can describe the motion of a single rigid body in a laboratory frame by defining $\boldsymbol{\xi}$ to be the body frame of rigid body, then we can describe its position through the rotation matrix $B = [\boldsymbol{B}_1, \boldsymbol{B}_2, \boldsymbol{B}_3]$ where B_i are basis vectors for the body frame and its origin point $\boldsymbol{x}_0$ relative to the laboratory frame. A position and velocity in the laboratory frame can be described by

$$\boldsymbol{x} = \boldsymbol{x}_0 + B \cdot \boldsymbol{\xi}$$
$$\dot{\boldsymbol{x}} = \dot{\boldsymbol{x}}_0 + \dot{B} \cdot \boldsymbol{\xi} + B \cdot \dot{\boldsymbol{\xi}}$$

We can redefine the velocity in terms of the rigid-body velocity $\boldsymbol{U}$ and angular velocity $\boldsymbol{\Omega}$ of the frame as

$$\dot{\boldsymbol{x}} = \boldsymbol{U} + \boldsymbol{\Omega} \times (\boldsymbol{x} - \boldsymbol{x}_0) + B \cdot \dot{\boldsymbol{\xi}}$$

The fluid flow can be described through the stokes flow equation defined in eq. (2.27),

$$u_i(\boldsymbol{y}) = \frac{1}{8\pi\mu} \int_{\partial D} S_{ij}^{\epsilon}(\boldsymbol{x}, \boldsymbol{y})\, f_j(\boldsymbol{y}) ds(\boldsymbol{y}) \quad \forall \boldsymbol{x} \in \partial D$$

The nonslip boundary condition means that the velocity of the fluid on the boundary of the rigid body ∂D is the same as the velocity of the boundary $\dot{\boldsymbol{x}}$

$$\boldsymbol{x}_i = \frac{1}{8\pi\mu} \int_{\partial D} S_{ij}^{\epsilon}(\boldsymbol{x}, \boldsymbol{y})\, f_j(\boldsymbol{y}) ds(\boldsymbol{y}) \quad \forall \boldsymbol{x} \in \partial D$$

By substituting the full equation for the velocity we can define the full Mobility problem as

$$\begin{aligned} -U_i - \epsilon_{ijk}\Omega_j\left(x_k - x_{0k}\right) + \frac{1}{8\pi\mu}\iint_{\partial D} S^{\epsilon}_{ij}(\boldsymbol{x},\boldsymbol{y}) f_j(\boldsymbol{y}) dS(\boldsymbol{y}) = B_{ij}\dot{\xi}_j \text{ for all } \boldsymbol{x} \in \partial D, \\ \iint_{\partial D} f_i(\boldsymbol{y}) dS(\boldsymbol{y}) = 0 \\ \iint_{\partial D} \epsilon_{ikj} y_k f_j(\boldsymbol{y}) dS(\boldsymbol{y}) = 0, \end{aligned} \tag{6.1}$$

where ϵ_{ijk} is the Levi-Civita symbol. We have taken the total force and Moment on the body to be 0 for the moment as we assume all external forces are negligible. Numerical discretisation of the problem leads to $3N$ scalar degrees of freedom in the traction $\boldsymbol{f}$ and a further 6 scalar degrees of freedom to describe the total velocity $\boldsymbol{U}$ and angular momentum $\boldsymbol{\Omega}$ totalling $3N + 6$ scalar degrees of freedom.

6.1 Numerical Implementation

The discretisation of the mobility problem is based on the same discretisation of that derived in section 2 or though the use of the NEAREST (section 3) method. First condering the Nyström discretisation of Cortez et al [12] we replace the integrals in eq. (6.1) with a numerical quadrature rule to obtain the problem.

$$\begin{aligned} -U_i - \epsilon_{ijk}\Omega_j\left(x_k[m] - x_{0k}\right) + \frac{1}{8\pi\mu}\sum_{n=1}^{N} S^{\epsilon}_{ij}(\boldsymbol{x}_m,\boldsymbol{x}_n) f_j(\boldsymbol{x}_n) A_n = B_{ij}\dot{\xi}_j[m] \\ \text{for all } m = 1,\ldots,N \\ \sum_{n=1}^{N} f_i(\boldsymbol{x}_n) A_n = 0 \\ \sum_{n=1}^{N} \epsilon_{ikj} x_k[n] f_j(\boldsymbol{x}_n) A_n = 0, \end{aligned}$$

where $x_i[n]$ denotes the the position of the nth quadrature point in the ith axis and A_n the qudarture weight at the point $\boldsymbol{x}_n$. We have again considered the Nyström discretisation for this problem to aid such that we have a well defined problem for which we can compute the solution to the linear system. Through the use of the a nearest neighbour matrix

(eq. (3.3))

$$\nu[q,\hat{n}] = \begin{cases} 1/k & \text{if } \hat{n} = \underset{n=1,\dots,N}{\operatorname{argmin}} |x_n - X_q|, \\ 0 & \text{otherwise.} \end{cases}$$

where k is the number of points in $\{x_n\}$ which are equidistant to $\{X_q\}$ we can perform the same discretisation seen in section 3 to obtain.

$$-U_i - \epsilon_{ijk}\Omega_j \left(x_k[m] - x_{0k}\right) + \frac{1}{8\pi\mu} \sum_{n=1}^{N} f_j(\boldsymbol{x}_n) A_n \sum_{q=1}^{Q} S_{ij}^{\epsilon}(\boldsymbol{x}_m, \boldsymbol{X}_q)\nu[q,n] = B_{ij}\dot{\xi}_j[m]$$

$$\text{for all } m = 1, \dots, N$$

$$\sum_{n=1}^{N} f_i(\boldsymbol{x}_n) A_n \sum_{q=1}^{Q} \delta_{ij}\nu[q,n] = 0$$

$$\sum_{n=1}^{N} f_j(\boldsymbol{x}_n) A_n \sum_{q=1}^{Q} \epsilon_{ikj} X_k[q]\nu[q,n] = 0,$$

Both discretisation correspond to $3N + 6$ equations in $3N + 6$ unknowns $F_j(\boldsymbol{x}_n)$, U_j and Ω_j for $n = 1, \dots, N$ and $j = 1, 2, 3$. We can consider this equation in terms of a block matrix form as we did for the resistance problem (section 2.3) where we augment the force vector of unknowns with the velocity $\boldsymbol{U}$ and angular velocity $\boldsymbol{\Omega}$ which are both 3×1 column vectors. The right hand side of the equation is augmented with two 3×1 column vectors of zeros which denote the prescribed zero total force and moment on the

rigid body. This gives us a final form of our block matrix system as

$$\begin{pmatrix} A_{11}\,A_{12}\,A_{13}\;B_1^U\,B_1^\Omega \\ A_{21}\,A_{22}\,A_{23}\;B_2^U\,B_2^\Omega \\ A_{31}\,A_{32}\,A_{33}\;B_3^U\,B_3^\Omega \\ B_1^F\;B_2^F\;B_3^F \qquad \mathbf{0}_{6\times 6} \\ B_1^M\,B_2^M\,B_3^M \end{pmatrix} \begin{pmatrix} F_1(\boldsymbol{x}_1) \\ \vdots \\ F_1(\boldsymbol{x}_N) \\ F_2(\boldsymbol{x}_1) \\ \vdots \\ F_2(\boldsymbol{x}_N) \\ F_3(\boldsymbol{x}_1) \\ \vdots \\ F_3(\boldsymbol{x}_N) \\ \boldsymbol{U} \\ \boldsymbol{\Omega} \end{pmatrix} = \begin{pmatrix} B_{1j}\dot{\xi}_j(\boldsymbol{x}_1) \\ \vdots \\ B_{1j}\dot{\xi}_j(\boldsymbol{x}_N) \\ B_{2j}\dot{\xi}_j(\boldsymbol{x}_1) \\ \vdots \\ B_{2j}\dot{\xi}_j(\boldsymbol{x}_N) \\ B_{3j}\dot{\xi}_j(\boldsymbol{x}_1) \\ \vdots \\ B_{3j}\dot{\xi}_j(\boldsymbol{x}_N) \\ \mathbf{0}_{3\times 1} \\ \mathbf{0}_{3\times 1} \end{pmatrix}, \tag{6.2}$$

where the blocks for the Nyström discretisation are given as

$$\begin{aligned} A_{ij}(m,n) &= \frac{1}{8\pi\mu} S_{ij}^{\epsilon}(\boldsymbol{x}_m, \boldsymbol{x}_n) \text{ for } m,n = 1,\ldots,N \\ B_i^U(m,j) &= -\delta_{ij} \text{ for } m = 1,\ldots,N \\ B_i^\Omega(m,j) &= -\epsilon_{ijk}(x_k[m] - x_{0k}) \text{ for } m = 1,\ldots,N \\ B_j^F(i,n) &= \delta_{ij} \text{ for } n = 1,\ldots,N \\ B_j^M(i,n) &= \epsilon_{ikj} x_k[n] \text{ for } n = 1,\ldots,N \end{aligned}$$

or for the NEAREST discretisation

$$A_{ij}(m,n) = \frac{1}{8\pi\mu}\sum_{q=1}^{Q} S_{ij}^{\epsilon}(\boldsymbol{x}_m, \boldsymbol{X}_q)\nu[q,n] \text{ for } m,n = 1,\ldots,N$$

$$B_i^U(m,j) = -\delta_{ij} \text{ for } m = 1,\ldots,N$$

$$B_i^{\Omega}(m,j) = -\epsilon_{ijk}(x_k[m] - x_{0k}) \text{ for } m = 1,\ldots,N$$

$$B_j^F(i,n) = \delta_{ij}\sum_{q=1}^{Q}\nu[q,n] \text{ for } n = 1,\ldots,N$$

$$B_j^M(i,n) = \epsilon_{ikj}\sum_{q=1}^{Q} X_k[q]\nu[q,n] \text{ for } n = 1,\ldots,N$$

6.2 Boundaries

Often when working with stokes flow we wish to include boundaries in our numerical simulation. These might be simulating microswimmers sandwiched between a microscope slides and a coverslip [33] or simulating blood cells in cylindrical arteries, any such boundary will have a noticeable effect on the stokeslets and therefore the rigid body motion [34]. While solutions around large infinite planes can be simulated through the use of the Blakelet solution [35, 36], more complex geometry requires the use of a quadrature rule of over the boundary B. Taking the boundary of B to be ∂B

$$\begin{gathered}
-U_i - \epsilon_{ijk}\Omega_j\left(x_k - x_{0k}\right) + \frac{1}{8\pi\mu}\iint_{\partial D\cup\partial B} S_{ij}^{\epsilon}(\boldsymbol{x},\boldsymbol{y})f_j(\boldsymbol{y})dS(\boldsymbol{y}) = B_{ij}\dot{\xi}_j \text{ for all } \boldsymbol{x}\in\partial D,\\
\iint_{\partial D\cup\partial B} S_{ij}^{\epsilon}(\boldsymbol{x},\boldsymbol{y})f_j(\boldsymbol{y})dS(\boldsymbol{y}) = \dot{x}_i \text{ for all } \boldsymbol{x}\in\partial B\\
\iint_{\partial D} f_i(\boldsymbol{y})dS(\boldsymbol{y}) = 0\\
\iint_{\partial D} \epsilon_{ikj}y_k f_j(\boldsymbol{y})dS(\boldsymbol{y}) = 0,
\end{gathered} \tag{6.3}$$

This addition to the problem is numerically solved in the same way as for the single swimmer, by introducing a second discretisation (or third and fourth discretisation in the case of NEAREST) of the boundary ∂B. Now taking the original number of quadrature points to be N_D and the set of new quadrature points $\{\boldsymbol{x}_n\}$ where $n = N_D+1,\ldots,N_B$. Discretisation of the problem leads to the same block structure described in eq. (6.2) how-

ever with different block components in order to compute the new boundary conditions. For the single quadrature rule discetisation we obtain the block structure

$$A_{ij}(m,n) = \frac{1}{8\pi\mu} S^{\epsilon}_{ij}(\boldsymbol{x}_m, \boldsymbol{x}_n) \text{ for } m,n = 1,\ldots,N_s, N_D+1,\ldots,N_D+N_B$$

$$B^U_i(m,j) = \begin{cases} -\delta_{ij} \text{ for } m = 1,\ldots,N_D \\ 0 \text{ for } m = N_D+1,\ldots,N_D+N_B \end{cases}$$

$$B^{\Omega}_i(m,j) = \begin{cases} -\epsilon_{ijk}(x_k[m] - x_{0k}) \text{ for } m = 1,\ldots,N \\ 0 \text{ for } m = N_D+1,\ldots,N_D+N_B \end{cases}$$

$$B^F_j(i,n) = \begin{cases} \delta_{ij} \text{ for } n = 1,\ldots,N \\ 0 \text{ for } n = N_D+1,\ldots,N_D+N_B \end{cases}$$

$$B^M_j(i,n) = \begin{cases} \epsilon_{ikj} x_k[n] \text{ for } n = 1,\ldots,N \\ 0 \text{ for } n = N_D+1,\ldots,N_D+N_B \end{cases}$$

and for the NEAREST discretisation where the coarse force discretisation is extended in the same way as the Nyström discretisation $\{\boldsymbol{x}_n\}$ where $n = 1,\ldots,N_D+N_B$ and a fine

quadrature rule $\{\boldsymbol{X}_q\}$ where $q = 1, \ldots, Q = Q_D + Q_B$

$$A_{ij}(m,n) = \frac{1}{8\pi\mu}\sum_{q=1}^{Q} S_{ij}^{\epsilon}(\boldsymbol{x}_m, \boldsymbol{X}_q)\nu[q,n] \text{ for } m,n = 1,\ldots,N_s, N_D+1,\ldots,N_D+N_B$$

$$B_i^U(m,j) = \begin{cases} -\delta_{ij} \text{ for } m = 1,\ldots,N \\ \\ \text{for } m = N_D+1,\ldots,N_D+N_B \end{cases}$$

$$B_i^{\Omega}(m,j) = \begin{cases} -\epsilon_{ijk}(x_k[m] - x_{0k}) \text{ for } m = 1,\ldots,N \\ \\ \text{for } m = N_D+1,\ldots,N_D+N_B \end{cases}$$

$$B_j^F(i,n) = \begin{cases} \delta_{ij}\sum_{q=1}^{Q}\nu[q,n] \text{ for } n = 1,\ldots,N \\ \\ 0 \text{ for } n = N_D+1,\ldots,N_D+N_B \end{cases}$$

$$B_j^M(i,n) = \begin{cases} \epsilon_{ikj}\sum_{q=1}^{Q} X_k[q]\nu[q,n] \text{ for } n = 1,\ldots,N \\ \\ 0 \text{ for } n = N_D+1,\ldots,N_D+N_B \end{cases} .$$

This is now as system of $3N = 3N_D + 3N_B + 6$ unknowns in $3N$ equations. The Vector of unknowns is again arranged as in eq. (6.2) where $N = N_D + N_B$. Writing the right hand side as $(V_1, V_2, V_3)^T$ we can write that

$$V_i[n] = \begin{cases} B_{ij}\dot{\xi}_j(\boldsymbol{x}_n) \text{ for } n = 1,\ldots,N_D \\ \\ \dot{x}_i[n] \text{ for } n = N_D+1,\ldots,N_D+N_B \end{cases}$$

7 Multiple Swimmers

It is often the cases that systems will contain multiple rigid bodies of either the same type or different types. If we consider N_{sw} swimmers then we can write for the case of

the mobility problem with boundaries that

$$
\begin{gathered}
-U_i[n] - \epsilon_{ijk}\Omega_j[n]\left(x_k[n] - x_{0k}[n]\right) + \frac{1}{8\pi\mu}\iint_{\partial D\cup\partial B} S^{\epsilon}_{ij}(\boldsymbol{x},\boldsymbol{y}) f_j(\boldsymbol{y}) dS(\boldsymbol{y}) = B_{ij}[n]\dot{\xi}_j[n] \\
\text{for all } \boldsymbol{x} \in \partial D[n], \\
\iint_{\partial D\cup\partial B} S^{\epsilon}_{ij}(\boldsymbol{x},\boldsymbol{y}) f_j(\boldsymbol{y}) dS(\boldsymbol{y}) = \dot{x}_i \text{ for all } \boldsymbol{x} \in \partial B \\
\iint_{\partial D[n]} f_i(\boldsymbol{y}) dS(\boldsymbol{y}) = 0 \\
\iint_{\partial D[n]} \epsilon_{ikj} y_k f_j(\boldsymbol{y}) dS(\boldsymbol{y}) = 0,
\end{gathered}
\tag{7.1}
$$

for each $n = 1, \ldots, N_{sw}$ where $\partial D[n]$ is the boundary of the nth swimmer and $\partial D = \bigcup_{n=1}^{N_{sw}} \partial D[n]$. The numerical discretisation of eq. (7.1) is similar to that of eq. (6.1) or eq. (6.3) and retains the same block structure as eq. (6.2) , however is significantly more notionally complex, partially when the number of points discretising each swimmer is different. For this reason we will only consider the more complex NEAREST discretisation where the Nyström can be obtained in the limiting case where the coarse force discritisation and fine quadrature rule are identical and $\nu = I_{N\times N}$.

If we consider the N_{sw} swimmers to have collocation points $x_i^{(1)}[\cdot], \ldots, x_i^{(N_{sw})}[\cdot]$ where $x_i^{(1)}[\cdot]$ denotes all the ith components of the first swimmers collocation points and $x_i^{(B)}[\cdot]$ is the ith components of the boundaries collocation points. Using the same ordering convention used in the previous section we have that

$$
\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3)^T \text{ with } \boldsymbol{x}_1 = (x_i^{(1)}[\cdot], \ldots, x_i^{(N_{sw})}[\cdot], x_i^{(B)}[\cdot])
$$

The translational and angualar velocities are given by $U_i^{(1)}, \ldots, U_i^{(N_{sw})}$ and $\Omega_i^{(1)}, \ldots, \Omega_i^{(N_{sw})}$ respectively and the the boundary collocation points by $x_i^{(b)}[\cdot]$. The ordering of discritisation remains the same as eq. (6.2) where if we denote the vector of unknowns as $(\boldsymbol{F}_1, \boldsymbol{F}_2, \boldsymbol{F}_3, \boldsymbol{U}_1, \boldsymbol{U}_2, \boldsymbol{U}_3, \boldsymbol{\Omega}_1, \boldsymbol{\Omega}_2, \boldsymbol{\Omega}_3)^T$ where $\boldsymbol{F}_i = (F_i^{(1)}[\cdot], \ldots, F_i^{(N_{sw})}[\cdot], F_i^{(b)}[\cdot])$, $\boldsymbol{U}_i = (U_i^{(1)}, \ldots, U_i^{(N_{sw})})^T$ and $\boldsymbol{\Omega}_i = (\Omega_i^{(1)}, \ldots, \Omega_i^{(N_{sw})})^T$. The same ordering convention is used for the right hand side velocities, total forces and moments. If the number of force points associated with a swimmer s is $N_D(s)$ and the number of force points on the boundary

is N_B then the total number of points is $N_F = \sum_{s=1}^{N_{sw}} N_D(s) + N_B$. We will also define an index $\iota(s) = \sum_{\alpha=1}^{s-1} N_D(\alpha)$ for $1 < s \leq N_{sw}$ to be the location of the sth swimmer in $\boldsymbol{x}_i$, as $\iota(s)$ is defined above for all of $1 < s \leq N_{sw}$ we also define $\iota(1) = 1$. The ordering of the fine quadrature points is arbitrary due to its mapping through ν, we will denote this set as $\{\boldsymbol{X}_q\}$ for $q = 1, \dots Q$ as we have before. The Stokeslet matrix remains the same and is constructed by

$$A_{ij}(\alpha, \beta) = \frac{1}{8\pi\mu} \sum_{q=1}^{Q} S_{ij}^{\epsilon}(\boldsymbol{x}_\alpha, \boldsymbol{X}_q)\nu[q, \beta] \text{ for } \alpha, \beta = 1, \dots, N_F$$

Defining $\mathbf{1}^{(n)}$ to be a column vector of ones with length n and $\mathbf{0}^{(m\times n)}$ to be a $(m \times n)$ matrix of zeros. We can therefore define the $N_F \times N_{sw}$ matrices

$$\tilde{x}_i(\cdot, \cdot) = \begin{pmatrix} x_i^{(1)}[\cdot] - x_{0i}^{(1)} & & \\ & \ddots & \\ & & x_i^{(N_{sw})}[\cdot] - x_{0i}^{(N_{sw})} \\ & \mathbf{0}^{(N_B \times N_{sw})} & \end{pmatrix}.$$

Then

$$B^U = \mathbb{1}_3 \otimes \begin{pmatrix} -\mathbf{1}^{(N_D(1))} & & \\ & \ddots & \\ & & -\mathbf{1}^{(N_D(N_{sw}))} \\ & \mathbf{0}^{(N_B \times N_{sw})} & \end{pmatrix}, \; B^\Omega = \begin{pmatrix} & -\tilde{x}_3(\cdot, \cdot) & \tilde{x}_2(\cdot, \cdot) \\ \tilde{x}_3(\cdot, \cdot) & & -\tilde{x}_1(\cdot, \cdot) \\ -\tilde{x}_2(\cdot, \cdot) & \tilde{x}_1(\cdot, \cdot) & \end{pmatrix}$$

with $\mathbb{1}_3$ denoting the 3×3 identity matrix and $\otimes$ the Kronecker product. In order to construct the $N_{sw} \times N_F$ matrices we define two $1 \times N_D(s)$ row vectors

$$\lambda^{(s)}[\cdot] = \sum_{q=1}^{Q} \nu[q, \gamma] \text{ for } \gamma = \iota(s), \cdots, \iota(s+1) - 1 \text{ and}$$

$$\chi_j^{(s)}[\cdot] = \sum_{q=1}^{Q} X_j(q)\nu[q, \gamma] \text{ for } \gamma = \iota(s), \cdots, \iota(s+1) - 1$$

these allow us to define

$$\tilde{\chi}_j(\cdot,\cdot) = \begin{pmatrix} \chi_j^{(1)}[\cdot] & & \\ & \ddots & \mathbf{0}^{(N_{sw}\times N_B)} \\ & & \chi_j^{(N_{sw})}[\cdot] \end{pmatrix}.$$

Then

$$B^F = \mathbb{1}_3 \otimes \begin{pmatrix} \lambda^{(1)}[\cdot] & & \\ & \ddots & \mathbf{0}^{(N_{sw}\times N_B)} \\ & & \lambda^{(N_{sw})}[\cdot] \end{pmatrix},\ B^M = \begin{pmatrix} & -\tilde{\chi}_3(\cdot,\cdot) & \tilde{\chi}_2(\cdot,\cdot) \\ \tilde{\chi}_3(\cdot,\cdot) & & -\tilde{\chi}_1(\cdot,\cdot) \\ -\tilde{\chi}_2(\cdot,\cdot) & \tilde{\chi}_1(\cdot,\cdot) & \end{pmatrix}$$

Finally defining the right hand side vector as $(\boldsymbol{V}_1, \boldsymbol{V}_1, \boldsymbol{V}_1, \mathbf{0}_{(6N_{sw}\times 1)})^T$ where

$$V_i^{(s)} = (B_{ij}^{(1)}\dot{\xi}_j^{(1)}[\cdot], \ldots, B_{ij}^{(N_{sw})}\dot{\xi}_j^{(N_{sw})}[\cdot], \dot{x}_i[1], \ldots, \dot{x}_i[N_B])$$

This allows us to form the full system of $3(N_F + 2N_{sw}) \times 3(N_F + 2N_{sw})$ system of linear equations e given in the same structure as eq. (6.2).

We will be solving this system through the use of GMRES (see **??**) as we are unable to form the full matrix with the KIFMM method. We can instead compute the matrix in block, we will compute the stokeslet matrix A with the KIFMM method first before constructing the B_i^U, B_i^Ω, B_i^F and B_i^M matrix blocks. For notational ease we will group these into two matrices

$$B = \begin{pmatrix} B_1^F & B_2^F & B_3^F \\ \\ B_1^M & B_2^M & B_3^M \end{pmatrix} \text{ and } B^T = \begin{pmatrix} B_1^U & B_1^\Omega \\ \\ B_2^U & B_2^\Omega \\ \\ B_3^U & B_3^\Omega \end{pmatrix}.$$

In the case of the Nyström discretisation we have that $B = -B^T$, however this is not the case for the NEAREST discretisation. We construct both B and B^T out side

of the GMRES method as they remain the same, we could also concider doing this for the KIFMM method as our discretisation points remain fixed for all GMRES iterations and as such so do the stokeslet kernel matrices uses. This would speed up the method as the majority of the computation in the KIFMM method is in the generation of the stokeslet kernel, particularly for the multipole to multipole translation (eq. (4.8)) and dence particle to particle interactions. This however would take a significant amount of memory as the stokeslet matrices needed for each M2L translation would need to be stored. A proposed optimisation is to use fast fourier transforms to instead compute the M2L translation. Concidering a M2L translation of the potential $\{\boldsymbol{f}^{AU}\}$ of a node A onto the downwards surface $\{\boldsymbol{q}^{BD}\}$ of a node B. Both the upwards and downwards equivilent potentials are defined to be a cartesian grid with the same number of quadrature points, by padding the centre of each surface with gridpoints of 0 dencity we can view the M2L translation as a 3D convolution [29] which can be carried out efficently by FFT. We would only need to compute a single FFT and IFFT (inverse fast fourier transform) per node and element wide multiplication for each node in the V or W interaction lists. We have yet to implent this method, but look to explore this approch if further research requires it.

In the hope to speed up the GMRES method without changing the underlying KIFMM method we will look at reducing the required number of GMRES iterations requried to converge to the required tolerance.

7.1 Initial Guess

An easierly implemented attempt at speeding at reduceing the number of GMRES iterations is to provide an intial guess to the GMRES solver such that initial relative residual error is already minimised and the number of iteration required to converge will be lower. As seen in appendix C the condition number of the system decreases as we decrease ϵ, this means that in most cases the required number of GMRES iteractions required for systems with smaller condition number to converge is often smaller that

7.2 Rescaling Mobility Matrix

7.3 Preconditioning

A more optimal way in which we can solve kyolov subspace methods is to use a precondtioner such that the matrix system we are try into use has a smaller condition number with more clustered eigenvalues. If we call the mobility matrix defined in eq. (6.2) M, the vector of unknowns $\boldsymbol{x}$ and the right hand side $\boldsymbol{b}$ then we can refer to the system as $Mx = b$. The Matrix M is much more badly conditioned (see **??**) that the stokeslet matrix A as as such requires a significant number of iteration to solve.

number of swimmers	number of DOF	No precondtioner	Inital Guess
		Rescaling	Least square commutator

8 Numerical Simulations

8.1 Rods in Shear flow

9 Conclusion

A GMRES

The Generalised Minimum Residual Method (GMRES) [19] is a iterative method based on Krylov methods for solving the System $A\boldsymbol{x} = \boldsymbol{b}$ where A is an arbitary non-symmetric matrix of order n and $\boldsymbol{b}$ is a known vector of order n. Krylov methods generate a Krylov subspace through repeativly performing vector matrix multiplication involving A, this allows for the computation of large systems where operation of order $\mathcal{O}(n^3)$ are too large to compute the matrix inverse or where we are unable to form the matrix by can compute the matrix vector product as in the Kernel Independent Fast Multiple Method (KIFMM)

(section 4) [37]. At each iteration $k \geq 1$ find the solution $\boldsymbol{y}_k$ from the subspace

$$\mathcal{K}_k(A, v^{(0)}) = \text{span}(\boldsymbol{v}^{(0)}, A\boldsymbol{v}^{(0)}, A^2\boldsymbol{v}^{(0)}, \ldots, A^{k-1}\boldsymbol{v}^{(0)})$$

where $\boldsymbol{v}^0$ is the normalised vector of the residual $\boldsymbol{r}^{(0)} = \boldsymbol{b} - A\boldsymbol{x}^{(0)}$ which reduces the residual to the least square problem $\min_{x \in \mathcal{K}(A,b)} \|b - Ax\|$. When no initial guess $\boldsymbol{x}^{(0)}$ is given we have that $\boldsymbol{r}^{(0)} = b$, experimental results for when an initial guess is provided can be seen in section 7.1. GMRES solves this least square problem through the use of a orthogonal basis $\{\boldsymbol{v}^{(1)}, \boldsymbol{v}^{(2)}, \ldots, \boldsymbol{v}^{(k)}\}$ for $\|(A, b)$, this is constructed using Arnoldi's method [38, 20]. The $k+1$ subspace is constructed from the k subspace by orthogonalising the vector $A\boldsymbol{v}_K$ vector against $\mathcal{K}_k(A, v^{(0)}$ subspace. This is done by

$$\hat{\boldsymbol{v}}_{k+1} = A\boldsymbol{v}_k - (h_{1,j}\boldsymbol{v}_1 + \cdots + h_{j,j}\boldsymbol{v}_j)$$

where $h_{ij} = v_j^* A v_j$ where is the conjugate transpose of v_j. We then normalise $\hat{\boldsymbol{v}}_{k+1}$ to obtain $\boldsymbol{v}_{k+1} = \hat{\boldsymbol{v}}_{k+1}/\|\hat{\boldsymbol{v}}_{k+1}\|$ By collecting the basis vectors in a matrix $V_k = [\boldsymbol{v}^{(1)}, \boldsymbol{v}^{(2)}, \ldots, \boldsymbol{v}^{(k)}]$ we get that the decomposition from Arnoldi's method as

$$AV_j = V_{j+1}H_j$$

where H_j is an upper Hessenberg matrix of size $j+1 \times j$ formed by $\hat{H}_k = [h_{ij}]_{1 \leq i \leq k+1, 1 \leq j \leq k}$. As our solution to the least square problem is in $\mathcal{K}_k(A, r^{(0)})$ then we can write it as $V_k\boldsymbol{y}^{(k)}$ for some vector $\boldsymbol{y}^{(k)}$. This means we can rewrite the least problem in terms of H_k as

$$\begin{aligned}
\beta_k &= \min_{\boldsymbol{y}^{(k)}} \|\beta_0 \boldsymbol{v}_0 - AV_k\boldsymbol{y}^{(k)}\| \\
&= \min_{\boldsymbol{y}^{(k)}} \|\beta_0 V_{k+1}\boldsymbol{e}_1 - V_{k+1}H_k\boldsymbol{y}^{(k)}\| \\
&= \min_{\boldsymbol{y}^{(k)}} \|\beta_0 \boldsymbol{e}_1 - H_k\boldsymbol{y}^{(k)}\|
\end{aligned}$$

where $\boldsymbol{e}_1 = [1, 0, \ldots, 0]^T$. This process is repeated until β_k is under some relative tolerance τ specified. The final estimation for $\boldsymbol{x}$ is therefore constructed from $\boldsymbol{x}^{(k)} = \boldsymbol{x}^{(k)} + V_k\boldsymbol{y}^{(k)}$.

Algorithm 1 The GMRES Algorithm

Choose $\boldsymbol{x}^{(0)}$, compute $\boldsymbol{r}^{(0)} = \boldsymbol{b} - A\boldsymbol{x}^{(0)}$, $\beta_0 = \|\boldsymbol{r}^{(0)}\|$, $\boldsymbol{v}^{(0)} = \boldsymbol{r}^{(0)}/\beta_0$
for $k = 1, 2, \ldots$ **Until** $\beta_k < \tau\beta_0$ **do**
    $\boldsymbol{w}_0^{(k+1)} = Av^{(k)}$
    for $l = 1$ to k **do**
        $h_{lk} = \langle \boldsymbol{w}_l^{(k+1)}, \boldsymbol{v}^{(l)} \rangle$
        $\boldsymbol{w}_{l+1}^{(k+1)} = \boldsymbol{w}_l^{(k+1)} - h_{lk}\boldsymbol{v}^{(l)}$
    end for
    $h_{k+1,k} = \|\boldsymbol{w}_{k+1}^{(k+1)}\|$
    $\boldsymbol{v}^{(k+1)} = \boldsymbol{w}_{k+1}^{(k+1)}/h_{k+1,k}$
    Compute $\boldsymbol{y}^{(k)}$ such that $\beta_k = \|\beta_0\boldsymbol{e}_1 - H_k\boldsymbol{y}^{(k)}\|$ is minimised, where
    $H_k = [h_{ij}]_{1\leq i\leq k+1, 1\leq j\leq k}$
end for
$\boldsymbol{x}^{(k)} = \boldsymbol{x}^{(k)} + V_k\boldsymbol{y}^{(k)}$, where $V_k = [\boldsymbol{v}^{(1)}, \boldsymbol{v}^{(2)}, \ldots, \boldsymbol{v}^{(k)}]$

GMRES will converge on the least square solution in at most n iterations, although will often converge to small values of β_k in less steps. The exact number of iteration is correlated closely with the condition of the matrix A where better conditioned matrices will converge quicker. In slow converging systems, where the number of iterations k required to converge to the required tolerance τ is large can result in large computation and storage requirements as the new vector $\boldsymbol{v}_{k+1}$ needs to be orthogonalised against all previous vectors in the basis. This means that the computation and storage requirements for the GMRES algorithm grows like $\mathcal{O}(kn)$. An adaption to the algorithm denoted GMRES(m) is used where after m iterations the best guess $\boldsymbol{u}^{(m)}$ is constructed and used in place of $\boldsymbol{u}^{(0)}$ as the initial guess. This provides a upper bound on the computation and storage requirement of the algorithm although may slow down convergence as each approximation is built using a smaller Krylov subspace.

B Tree traversal

In this section we will give a brief overview of the two tree traversals needed to preform the fast multiple algorithm. We will concider a simple binary tree where each node had two children nodes. If we concider the binary tree in fig. 6, then we can consider both pre and post order traversal. The post order traversal is used in cases where we need to

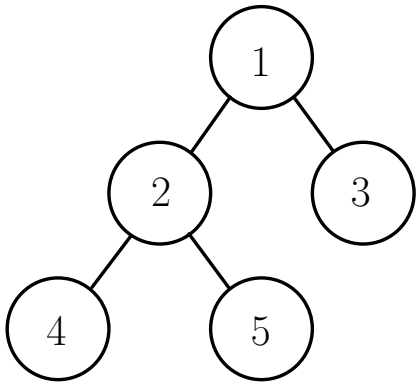


Figure 6: A simple tree structure

look at all the children of a node before looking at the node its self. This is useful for example in the upwards pass of fast multipole method where we need to compute the upwards equivilent surface of all node before computing the upwards equivilent surface of the node its-self with the multipole to multipole translation (eq. (4.7)). For the tree defined in the case of the binary tree in fig. 6 we traverse the nodes in the order $4, 5, 2, 3, 1$ with the path taken displayed in fig. 7a. Recursivly we can define this for a binary tree as

Algorithm 2 Binary post-order Traversal

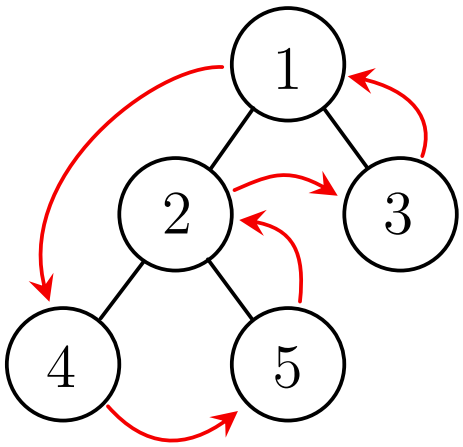
1: Recursively traverse the current node's left subtree
2: Recursively traverse the current node's right subtree
3: Visit the current node.

Another way in which we can traverse tree structures in the pre-order traversal where we work our way down from the root of the tree. We use this in fast multipole methods so we can use local to local translation (eq. (4.9)). The pre-order traversal differs from post-order traversal as we visit the current node first before recursivly traversing each subtree. For example the tree in **??** can be traversed using pre-order traversal in the order $1, 2, 4, 5, 3$ as illistrated in fig. 7b. The algorithm discribing pre-order traversal is

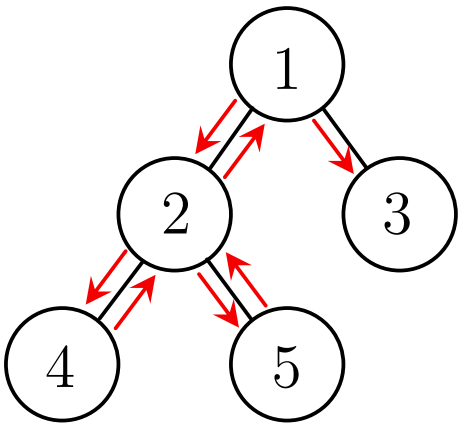
Algorithm 3 Binary pre-order Traversal

1: Visit the current node
2: Recursively traverse the current node's left subtree
3: Recursively traverse the current node's right subtree.

Application of both traversal to an octree traversal where we traverse each subtree



(a) Diagram showing post-order tree traversal



(b) Diagram showing pre-order tree traversal

Figure 7: Diagrams showing Pre and Post order traversal of the tree defined in fig. 6

individually, the order in which each subtree is traversed does not matter only only comes down to convention, however we do need to make sure that every subtree is traversed for each node.

C Condition number

Condition number of various matrices seen in the paper with $\epsilon = 1e-1$

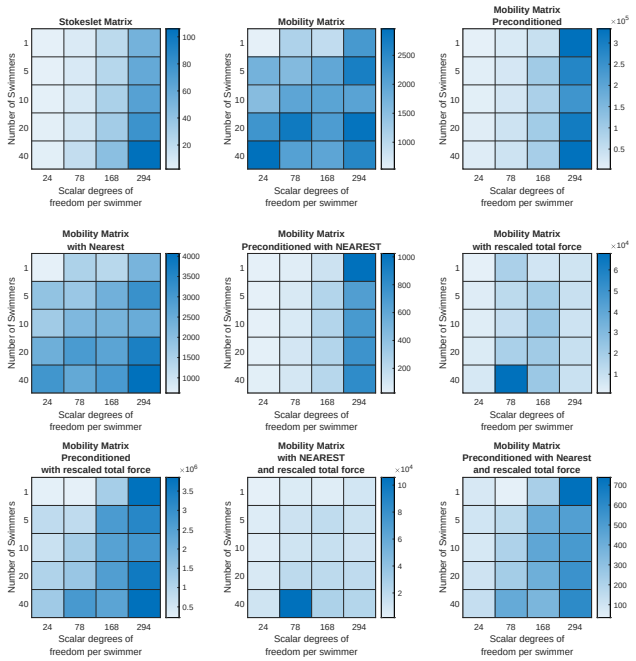


Figure 8: Caption

Condition number of various matrices seen in the paper with $\epsilon = 1e-2$

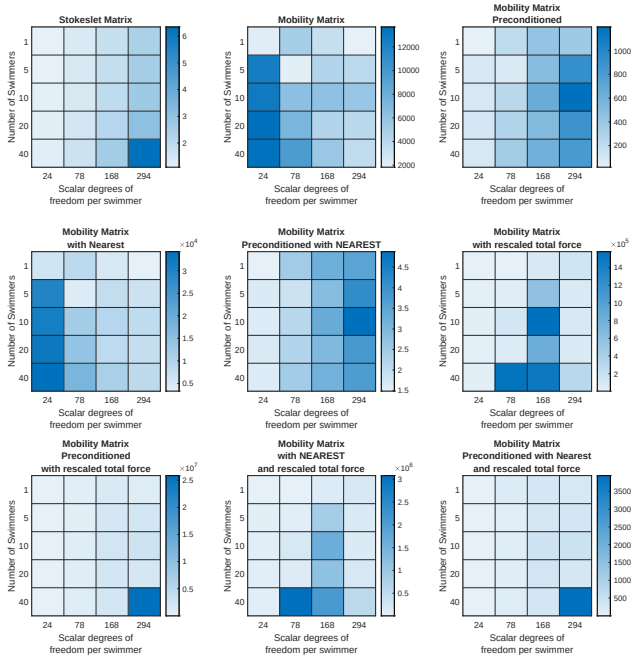


Figure 9: Caption

Condition number of various matrices seen in the paper with $\epsilon = 1e - 5$

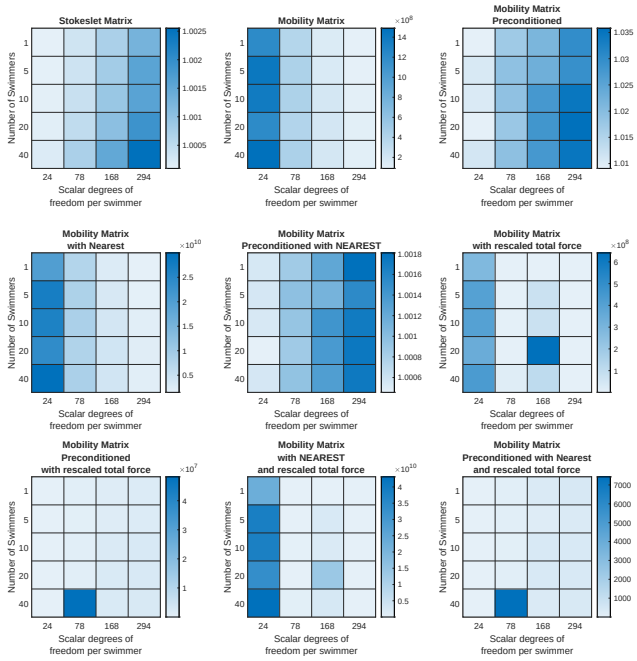


Figure 10: Caption

References

[1] C. I. Trombley and M. L. Ekiel-Jeżewska. "Basic Concepts of Stokes Flows". In: Springer, Cham, 2019, pp. 35–50. ISBN: 3030233693. DOI: 10.1007/978-3-030-23370-9_2.

[2] J. Blake. "A model for the micro-structure in ciliated organisms". In: *Journal of Fluid Mechanics* 55.1 (1972), pp. 1–23. ISSN: 1469-7645. DOI: 10.1017/S0022112072001612.

[3] J. J. Higdon. "A hydrodynamic analysis of flagellar propulsion". In: *Journal of Fluid Mechanics* 90.4 (1979), pp. 685–711. ISSN: 1469-7645. DOI: 10.1017/S0022112079002482.

[4] D. J. Smith, E. A. Gaffney, and J. R. Blake. "Mathematical modelling of cilia-driven transport of biological fluids". In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 465.2108 (Aug. 2009), pp. 2417–2439. ISSN: 14712946. DOI: 10.1098/rspa.2009.0018.

[5] N. Liron and R. Shahar. "Stokes flow due to a Stokeslet in a pipe". In: *Journal of Fluid Mechanics* 86.4 (1978), pp. 727–744. ISSN: 1469-7645. DOI: 10.1017/S0022112078001366.

[6] N. Liron and S. Mochon. "Stokes flow for a stokeslet between two parallel flat plates". In: *Journal of Engineering Mathematics 1976 10:4* 10.4 (Oct. 1976), pp. 287–303. ISSN: 1573-2703. DOI: 10.1007/BF01535565.

[7] C. Pozrikidis. *Boundary Integral and Singularity Methods for Linearized Viscous Flow*. Cambridge University Press, Feb. 1992. ISBN: 9780521406932. DOI: 10.1017/cbo9780511624124.

[8] G. J. Hancock. "The self-propulsion of microscopic organisms through liquids". In: *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences* 217.1128 (Mar. 1953), pp. 96–121. ISSN: 0080-4630. DOI: 10.1098/rspa.1953.0048.

[9] C. Oseen. *Neuere Methoden und Ergebnisse in der Hydrodynamik*. Leipzig: Akademische Verlagsgesellschaft, 1927.

[10] G. K. Batchelor. "An Introduction to Fluid Dynamics". In: *An Introduction to Fluid Dynamics* (Feb. 2000). DOI: 10.1017/CBO9780511800955.

[11] R. Cortez and S. J. Sci Comput. "The Method of Regularized Stokeslets". In: *Article in SIAM Journal on Scientific Computing* 23.4 (2001), pp. 1204–1225. DOI: 10.1137/S106482750038146X.

[12] R. Cortez, L. Fauci, and A. Medovikov. "The method of regularized Stokeslets in three dimensions: Analysis, validation, and application to helical swimming". In: *Physics of Fluids* 17.3 (Feb. 2005), p. 031504. ISSN: 10706631. DOI: 10.1063/1.1830486.

[13] S. D. Olson, S. Lim, and R. Cortez. "Modeling the dynamics of an elastic rod with intrinsic curvature and twist using a regularized Stokes formulation". In: *Journal of Computational Physics* 238 (Apr. 2013), pp. 169–187. ISSN: 0021-9991. DOI: 10.1016/J.JCP.2012.12.026.

[14] H. N. Nguyen and R. Cortez. "Reduction of the regularization error of the method of regularized stokeslets for a rigid object immersed in a three-dimensional stokes flow". In: *Communications in Computational Physics* 15.1 (2014), pp. 126–152. ISSN: 19917120. DOI: 10.4208/CICP.021112.290413A.

[15] B. Zhao, E. Lauga, and L. Koens. "Method of regularized stokeslets: Flow analysis and improvement of convergence". In: *Physical Review Fluids* 4.8 (Aug. 2019). ISSN: 2469990X. DOI: 10.1103/PhysRevFluids.4.084104.

[16] I. Stakgold. "Boundary Value Problems of Mathematical Physics: Volume 2". In: *Boundary Value Problems of Mathematical Physics: Volume 2* 2 (Jan. 1968). DOI: 10.1137/1.9780898719475.

[17] M. T. Gallagher, D. Choudhuri, and D. J. Smith. "Sharp Quadrature Error Bounds for the Nearest-Neighbor Discretization of the Regularized Stokeslet Boundary Integral Equation". In: *https://doi.org/10.1137/18M1191816* 41.1 (Feb. 2019), B139–B152. ISSN: 10957197. DOI: 10.1137/18M1191816.

[18] E. J. Nyström. "Über Die Praktische Auflösung von Integralgleichungen mit Anwendungen auf Randwertaufgaben". In: *https://doi.org/10.1007/BF02547521* 54.none (Jan. 1930), pp. 185–204. ISSN: 0001-5962. DOI: 10.1007/BF02547521.

[19] Y. Saad and M. H. Schultz. "GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems". In: *SIAM Journal on Scientific and Statistical Computing* 7.3 (1986), pp. 856–869. ISSN: 0196-5204. DOI: 10.1137/0907058.

[20] H. Elman, D. Silvester, and A. Wathen. *Finite Elements and Fast Iterative Solvers: with Applications in Incompressible Fluid Dynamics*. Oxford University Press, Sept. 2005. ISBN: 9780199678792. DOI: 10.1093/ACPROF:OSO/9780199678792.001.0001.

[21] D. J. Smith. "A boundary element regularized Stokeslet method applied to cilia- and flagella-driven flow". In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 465.2112 (Dec. 2009), pp. 3605–3626. ISSN: 14712946. DOI: 10.1098/RSPA.2009.0295.

[22] P. Sampaio et al. "Left-right organizer flow dynamics: How much cilia activity reliably yields laterality?" In: *Developmental Cell* 29.6 (June 2014), pp. 716–728. ISSN: 18781551. DOI: 10.1016/J.DEVCEL.2014.04.030/ATTACHMENT/FBEE4414-D38A-4686-BEE2-1A81267DEACF/MMC3.MP4.

[23] A. A. Smith et al. "Symmetry breaking cilia-driven flow in the zebrafish embryo". In: *Journal of Fluid Mechanics* 705 (Aug. 2012), pp. 26–45. ISSN: 1469-7645. DOI: 10.1017/JFM.2012.117.

[24] D. J. Smith. "A nearest-neighbour discretisation of the regularized stokeslet boundary integral equation". In: *Journal of Computational Physics* 358 (Apr. 2018), pp. 88–102. ISSN: 0021-9991. DOI: 10.1016/J.JCP.2017.12.008.

[25] R. Beatson and L. Greengard. "A short course on fast multipole methods". In: *Wavelets, Multilevel Methods and Elliptic PDEs*. Oxford University Press, 1997. Chap. 1, pp. 1–37. ISBN: 9780198501909.

[26] A. K. Tornberg and L. Greengard. "A fast multipole method for the three-dimensional Stokes equations". In: *Journal of Computational Physics* 227.3 (Jan. 2008), pp. 1613–1619. ISSN: 0021-9991. DOI: 10.1016/J.JCP.2007.06.029.

[27] H. Wang et al. "A parallel fast multipole accelerated integral equation scheme for 3D Stokes equations". In: *International Journal for Numerical Methods in Engineering* 70.7 (May 2007), pp. 812–839. ISSN: 1097-0207. DOI: 10.1002/NME.1910.

[28] R. Yokota and L. A. Barba. "Treecode and fast multipole method for N-body simulation with CUDA". In: *GPU Computing Gems Emerald Edition*. 2011, pp. 113–132. ISBN: 9780123849885. DOI: 10.1016/B978-0-12-384988-5.00009-7.

[29] L. Ying, G. Biros, and D. Zorin. "A kernel-independent adaptive fast multipole algorithm in two and three dimensions". In: *Journal of Computational Physics* 196.2 (May 2004), pp. 591–626. ISSN: 0021-9991. DOI: 10.1016/J.JCP.2003.11.021.

[30] L. Ying. "A kernel independent fast multipole algorithm for radial basis functions". In: *Journal of Computational Physics* 213.2 (2006), pp. 451–457. ISSN: 10902716. DOI: 10.1016/j.jcp.2005.09.010.

[31] M. W. Rostami and S. D. Olson. "Kernel-independent fast multipole method within the framework of regularized Stokeslets". In: *Journal of Fluids and Structures* 67 (Nov. 2016), pp. 60–84. ISSN: 10958622. DOI: 10.1016/j.jfluidstructs.2016.07.006.

[32] W. Yan and R. Blackwell. "Kernel aggregated fast multipole method: Efficient summation of Laplace and Stokes kernel functions". In: *Advances in Computational Mathematics* 47.5 (2021). ISSN: 15729044. DOI: 10.1007/s10444-021-09896-1.

[33] M. T. Gallagher et al. "Rapid sperm capture: High-throughput flagellar waveform analysis". In: *Human Reproduction* 34.7 (2019), pp. 1173–1185. ISSN: 0268-1161. DOI: 10.1093/HUMREP/DEZ056.

[34] N. Liron and J. R. Blake. "Existence of viscous eddies near boundaries". In: *Journal of Fluid Mechanics* 107 (1981), pp. 109–129. ISSN: 1469-7645. DOI: 10.1017/S0022112081001699.

[35] J. Ainley et al. "The method of images for regularized Stokeslets". In: *Journal of Computational Physics* 227.9 (Apr. 2008), pp. 4600–4616. ISSN: 0021-9991. DOI: 10.1016/J.JCP.2008.01.032.

[36] R. Cortez and D. Varela. "A general system of images for regularized Stokeslets and other elements near a plane wall". In: *Journal of Computational Physics* 285 (Mar. 2015), pp. 41–54. ISSN: 0021-9991. DOI: 10.1016/J.JCP.2015.01.019.

[37] I. C. Ipsen and C. D. Meyer. "The idea behind Krylov methods". In: *American Mathematical Monthly* 105.10 (1998), pp. 889–899. ISSN: 00029890. DOI: 10.2307/2589281.

[38] W. E. Arnoldi. "The principle of minimized iterations in the solution of the matrix eigenvalue problem". In: *Quarterly of Applied Mathematics* 9.1 (1951), pp. 17–29. ISSN: 0033-569X. DOI: 10.1090/QAM/42792.