Journal of Computational Physics 358 (2018) 88–102

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

A nearest-neighbour discretisation of the regularized stokeslet boundary integral equation

David J. Smith

School of Mathematics, University of Birmingham, Edgbaston, Birmingham, B15 2TT, UK

ARTICLE INFO

Article history:
Received 28 April 2017
Received in revised form 3 December 2017
Accepted 5 December 2017
Available online 28 December 2017

Keywords:
Stokes flow
Regularized stokeslet
Boundary integral
Meshfree

ABSTRACT

The method of regularized stokeslets is extensively used in biological fluid dynamics due to its conceptual simplicity and meshlessness. This simplicity carries a degree of cost in computational expense and accuracy because the number of degrees of freedom used to discretise the unknown surface traction is generally significantly higher than that required by boundary element methods. We describe a meshless method based on nearest-neighbour interpolation that significantly reduces the number of degrees of freedom required to discretise the unknown traction, increasing the range of problems that can be practically solved, without excessively complicating the task of the modeller. The nearest-neighbour technique is tested against the classical problem of rigid body motion of a sphere immersed in very viscous fluid, then applied to the more complex biophysical problem of calculating the rotational diffusion timescales of a macromolecular structure modelled by three closely-spaced non-slender rods. A heuristic for finding the required density of force and quadrature points by numerical refinement is suggested. Matlab/GNU Octave code for the key steps of the algorithm is provided, which predominantly use basic linear algebra operations, with a full implementation being provided on github. Compared with the standard Nyström discretisation, more accurate and substantially more efficient results can be obtained by de-refining the force discretisation relative to the quadrature discretisation: a cost reduction of over 10 times with improved accuracy is observed. This improvement comes at minimal additional technical complexity. Future avenues to develop the algorithm are then discussed.

1. Introduction

When attempting to formulate and solve mathematical models of microscopic biological flow systems, for example involving macromolecular structures, swimming cells and cilia, a significant challenge to overcome is that the flow domain is typically bounded by curved, moving surfaces. Often it is of interest to model line-like objects such as cilia and flagella, and point-like bodies such as suspensions of many bacteria, in addition to genuinely 2D surfaces. The Stokes flow equations are linear, and in some celebrated cases it has been found possible to make significant analytical progress, for example by exploiting small amplitude expansions in the boundary movement [1] or slenderness [2–4], for certain idealised problems (for a more detailed review of the field, see Lauga & Powers [5]). However the majority of problems of practical interest, typically involving multiple cells, non-planar domains and large amplitude motions, require computational modelling, and there has been intensive activity in this area in the last decade.

E-mail address: D.J.Smith@bham.ac.uk.

https://doi.org/10.1016/j.jcp.2017.12.008

The linearity of the flow equations enables the formulation of methods based on the boundary integral equation for Stokes flow; these methods remove the need to discretise and solve directly in the flow volume, as would be necessary for the finite element method. This reduction in dimensionality both removes the need to mesh and re-mesh the evolving flow domain, and vastly reduces the size of the linear algebra problem resulting from discretisation. In certain respects these methods were anticipated by the computational slender body theory work of Higdon [6] and [7]; relatively early examples of the 'fully-fledged' boundary element method for Stokes flow was developed by Phan-Thien and colleagues [8,9]. The achievements of the latter group with late 1980s/early 1990s computational hardware set a benchmark for work in the current era of desktop machines with multi-gigabyte RAM. It should of course be noted that there have been major algorithmic developments in numerical methods for Stokes flow in the intervening period, including the completed double-layer boundary integral equation [10,11], hybrid boundary integral-multipole methods [12], spectral discretisation combined with the fast multipole method [13,14], quadrature by expansion [11], and slender body theory combined with these techniques [15]. These approaches are generally employed by numerical experts to solve problems at the limits of computational feasibility, involving very large numbers of interacting bodies.

The classical boundary element method for Stokes flow, along with the more advanced methods described above, are both accurate and efficient. However, they present two technical challenges in their implementation, particularly when considered from the point of view of users who are not computational specialists. The first challenge is the need to generate a surface *mesh*, i.e. a geometric discretisation of all surfaces in the problem consisting of oriented smooth, and smoothly-connected, patches which interpolate several surface points.[1] While much easier than the volumetric meshing that would be required for the finite element method, meshes of even moderately complicated biomolecular or cellular structures may require significant time and ingenuity to create, and may not be suited to automated generation – as might be needed to study biological heterogeneity. Furthermore, some objects will appear to a very good approximation as lines or points – detailed surface meshing of these bodies may involve a level of computational refinement that is unwarranted. The second challenge – which has arguably been addressed through the availability of library code such as BEMLIB [16] – is the singularity of the stokeslet velocity and stress kernels, and requirement for semi-analytical quadrature methods. The latter issue does however present an additional layer of complexity for those who are not numerical specialists.

The method of regularized stokeslets, introduced by Cortez and colleagues [17–20], has proved to be an effective and accessible method for simulating and analysing microscale biological flows. This method deals effectively with both of the above difficulties by removing the need for a true mesh, requiring only a set of discrete points approximating the solid objects in the flow, and regularizing the integral kernel so that specialised quadrature is not required. The core idea is the derivation of a family of regularized versions of the singular stokeslet/Oseen tensor kernel that nevertheless satisfy exact conservation of mass. Whereas the singular stokeslet corresponds to the Stokes flow produced by a Dirac delta force-per-unit-volume distribution, a regularized stokeslet corresponds to the Stokes flow produced by a 'blob', i.e. a finite force-per-unit-volume distribution which approximates a Dirac delta function. Cortez and colleagues have derived various versions of the regularized stokeslet corresponding to both 2D [17] and 3D [18] domains, to various forms of blob distribution [17], with image systems to represent a plane boundary [19,21], and for periodic problems [20]. We will not attempt to give a comprehensive survey of applications of the method of regularized stokeslets; it suffices to note that a Google Scholar search on 28th April 2017 with the term "regularized stokeslets" produced 250 results since 2012.

The standard numerical implementation of the method of regularized stokeslets is to employ a Nyström discretisation of the Fredholm integral equation, which replaces the integral directly with a quadrature rule. This method is very simple to implement, and has been used in the great majority of published work. This simplicity does however come at a computational cost, arising from the fact that the quantity of interest in a boundary integral equation method, the surface traction distribution, varies much more slowly than the near-singular kernel. Therefore very many degrees of freedom, corresponding to the discretisation of the traction, are required in order for the quadrature to be accurate. Furthermore, there is a coupling between the discretisation length scale and the regularization parameter that must be satisfied in order for results to be considered converged. As a consequence, the RAM requirements alone for relatively simple geometries may be very high, as evident in a number of recent studies on helical flagella for example.

The issue of the computational cost of the method of regularized stokeslets was discussed in an earlier paper [22], in which we suggested employing a boundary element discretisation of the regularized stokeslet boundary integral equation. This approach is undoubtedly computationally efficient, and formed the basis for subsequent detailed modelling of the left right organising structures of mouse [23] and zebrafish [24,25], however it becomes necessary to generate a mesh in the same way as the classical boundary element method.

In this paper we will describe an alternative 'nearest-neighbour' discretisation of the method of regularized stokeslets which retains the meshless simplicity of the standard approach, but has greatly reduced computational cost. Alongside the mathematical description, an implementation in Matlab®/GNU Octave will be given, and applied to a simple test problem of the drag and moment on a sphere or prolate spheroid undergoing rigid body motion, followed by a more complex problem of calculating the rotational diffusion timescale of a biological macromolecule.

[1] In this paper the term *mesh* will be reserved for an object (P, E) where $P = \{\boldsymbol{x}[1], \ldots, \boldsymbol{x}[N]\} \in \mathbb{R}^3$ is an ordered set of *points/nodes*, and E is a table defining the *elements* of the mesh, e.g. for a mesh of flat triangles, the elements take the form $(\boldsymbol{x}[E(1,e)], \boldsymbol{x}[E(2,e)], \boldsymbol{x}[E(3,e)])$. Where we refer to a set P without the associated table defining the elements, the terms *discretisation* or *points* will be used instead. The aim of this study is to achieve improved accuracy and efficiency without needing to construct E.

2. Stokeslets and boundary integral methods

The very low Reynolds numbers associated with microscopic flows on the length scales of macromolecules and cells motivates the study of the Stokes flow equations for viscous-dominated flow. The dimensionless form of these equations is,

$$-\nabla p + \nabla^2 \boldsymbol{u} = 0, \quad \nabla \cdot \boldsymbol{u} = 0, \tag{1}$$

augmented with the no-slip, no-penetration boundary condition $\boldsymbol{u}(\boldsymbol{X}) = \dot{\boldsymbol{X}}$ for boundary points $\boldsymbol{X}$. The basis for boundary integral and singularity methods is to exploit the linearity of eq. (1) to construct solutions satisfying the required boundary conditions from sums and/or integrals of fundamental solutions.

The classical singular fundamental solution is the stokeslet or Oseen tensor, given by the second rank tensor S_{jk} and first rank tensor P_k for which $\boldsymbol{u} = (S_{1k}, S_{2k}, S_{3k})$ and $p = P_k$ are the solutions of the Stokes flow equations with a Dirac delta distribution force-per-unit-volume located at $\boldsymbol{y}$:

$$-\nabla p + \nabla^2 \boldsymbol{u} + 8\pi \boldsymbol{e}_k \delta(\boldsymbol{x} - \boldsymbol{y}) = 0, \quad \nabla \cdot \boldsymbol{u} = 0. \tag{2}$$

The form of the stokeslet in 3D is,

$$S_{jk}(\boldsymbol{x}, \boldsymbol{y}) = \frac{\delta_{jk}}{|\boldsymbol{x} - \boldsymbol{y}|} + \frac{(x_j - y_j)(x_k - y_k)}{|\boldsymbol{x} - \boldsymbol{y}|^3}, \tag{3}$$

$$P_k(\boldsymbol{x}, \boldsymbol{y}) = 2\frac{x_k - y_k}{|\boldsymbol{x} - \boldsymbol{y}|^3}. \tag{4}$$

The singularity method for Stokes flow involves seeking an approximate solution to equation (1) by locating Stokeslets, and sometimes higher order stokes-multipoles, outside of the flow domain. For example, singularities may be located inside cells, or along the centrelines of cilia and flagella as in slender body theory; the simplest example is perhaps the solution to Stokes flow driven by a translating sphere, which can be expressed as the sum of a stokeslet and source-dipole (the latter being a special case of the stokes-quadrupole) at the centre of the sphere. Review and references are given for example Smith et al. [22].

Conversely, the boundary integral method for Stokes flow involves formulating the exact integral equation,

$$u_j(\boldsymbol{y}) = -\frac{1}{8\pi} \iint_{\partial D} S_{ij}(\boldsymbol{x}, \boldsymbol{y}) f_i(\boldsymbol{x}) dS(\boldsymbol{x}) + \frac{1}{8\pi} \iint_{\partial D} u_i(\boldsymbol{x}) T_{ijk}(\boldsymbol{x}, \boldsymbol{y}) n_k(\boldsymbol{x}) dS(\boldsymbol{x}), \tag{5}$$

where T_{ijk} is the stress tensor associated with the Stokes flow $\boldsymbol{u} = (S_{1k}, S_{2k}, S_{3k})$, $p = P_k$, given by

$$T_{ijk}(\boldsymbol{x}, \boldsymbol{y}) = -\frac{6(x_i - y_i)(x_j - y_j)(x_k - y_k)}{|\boldsymbol{x} - \boldsymbol{y}|^5}. \tag{6}$$

The summation convention for repeated indices is used throughout. The boundary integral equation is solved numerically by taking the limit of equation (5) as $\boldsymbol{y}$ approaches the bounding surfaces of the domain from within the fluid, then performing discretisation of the surface geometry ∂D and traction $\boldsymbol{f}$. If the boundary of the domain is stationary and immersed objects in the domain are rigid bodies, the 'double layer' term arising from the integral of the stress is identically zero and so the flow is given exactly by a surface distribution of stokeslets only; under the weaker condition that $\iint_{\partial D} \boldsymbol{u} \cdot \boldsymbol{n} dS = 0$ it can also be shown that the double layer integral may be eliminated by taking a modified Stokeslet density, which is no longer precisely the surface traction. In either case, the flow is given exactly by boundary integrals of 'single layer' stokeslet velocity tensors only [26].

A detailed exposition of the boundary element method for Stokes flow and its numerical implementation is given by Pozrikidis [26,16]. The boundary integral and singularity methods may be hybridised to formulate approximate but accurate and efficient simulation of cell movement [27].

The integral equation problem formed from equation (5) in the limit $\boldsymbol{y} \to \boldsymbol{Y} \in \partial D$ possesses singular integrals which require specialised evaluation; moreover line and point singularity distributions, while they may not lie strictly in the flow domain, may nevertheless complicate the evaluation of flow fields for purposes such as particle tracking. An additional complication for boundary element methods is the requirement to build a true surface mesh. It should be emphasised that these issues are technical complications rather than inherent problems, however methods which do not possess these complications are appealing, particularly for biological flow, as evidenced by the rapid adoption and use of the method of regularized stokeslets, which we will briefly review in the next section.

3. The method of regularized stokeslets and its numerical implementation

Cortez [17] formulated the regularized stokeslet as the exact solution to the incompressible Stokes flow equations forced by a spatially-smoothed force per unit volume, $\phi_\epsilon(\boldsymbol{x} - \boldsymbol{y})$,

$$-\nabla p + \nabla^2 \boldsymbol{u} + 8\pi \boldsymbol{e}_k \phi_\epsilon(\boldsymbol{x} - \boldsymbol{y}) = 0, \quad \nabla \cdot \boldsymbol{u} = 0. \tag{7}$$

The 'blob' ϕ_ϵ denotes a family of functions parameterised by ϵ satisfying $\int \ldots \int_{\mathbb{R}^n} \phi_\epsilon dV = 1$, and tending to a Dirac delta distribution in the limit $\epsilon \to 0$. The derivation of specific forms of the regularized stokeslet were discussed by Cortez and colleagues [17,18]; we will suffice by noting that a frequently-used form for 3D flow is based on the blob function,

$$\phi_\epsilon(\boldsymbol{\xi}) = \frac{15\epsilon^4}{8\pi(|\boldsymbol{\xi}|^2+\epsilon^2)^{7/2}}, \tag{8}$$

which leads to the regularized Stokeslet pressure and velocity tensors,

$$P_j^\epsilon(\boldsymbol{x},\boldsymbol{y}) = (x_j - y_j)\frac{2|\boldsymbol{x}-\boldsymbol{y}|^2+5\epsilon^2}{(|\boldsymbol{x}-\boldsymbol{y}|^2+\epsilon^2)^{5/2}}, \tag{9}$$

$$S_{ij}^\epsilon(\boldsymbol{x},\boldsymbol{y}) = \delta_{ij}\frac{|\boldsymbol{x}-\boldsymbol{y}|^2+\epsilon^2}{(|\boldsymbol{x}-\boldsymbol{y}|^2+\epsilon^2)^{3/2}} + \frac{(x_i-y_i)(x_j-y_j)}{(|\boldsymbol{x}-\boldsymbol{y}|^2+\epsilon^2)^{3/2}}, \tag{10}$$

$$T_{ijk}^\epsilon(\boldsymbol{x},\boldsymbol{y}) = -\frac{6(x_i-y_i)(x_j-y_j)(x_k-y_k)}{(|\boldsymbol{x}-\boldsymbol{y}|^2+\epsilon^2)^{5/2}} - \frac{3\epsilon^2[(x_i-y_i)\delta_{jk}+(x_j-y_j)\delta_{ik}+(x_k-y_k)\delta_{ij}]}{(|\boldsymbol{x}-\boldsymbol{y}|^2+\epsilon^2)^{5/2}}. \tag{11}$$

The regularized counterpart to the classical boundary integral equation (5) in 3D is,

$$\begin{aligned} u_j(\boldsymbol{y}) &\approx \iiint_{\mathbb{R}^3} u_j(\boldsymbol{x})\phi_\epsilon(\boldsymbol{x}-\boldsymbol{y})dV(\boldsymbol{x}) \\ &= -\frac{1}{8\pi}\iint_{\partial D} S_{ij}^\epsilon(\boldsymbol{x},\boldsymbol{y})f_i(\boldsymbol{x})dS(\boldsymbol{x}) - \frac{1}{8\pi}\iint_{\partial D} u_i(\boldsymbol{x})T_{ijk}^\epsilon(\boldsymbol{x},\boldsymbol{y})n_k(\boldsymbol{x})dS(\boldsymbol{x}). \end{aligned} \tag{12}$$

Unlike the classical boundary integral equation, the regularized version (12) is approximate even before the numerical discretisation is carried out; for the blob function (8) the error is $O(\epsilon^2)$ for $\boldsymbol{y}$ greater than distance $\sqrt{5\epsilon/2}$ from the boundary, and $O(\epsilon)$ otherwise [18]. The double layer integral is typically eliminated in practical implementations of the regularized stokeslet. This elimination may be formally justified for boundaries undergoing rigid body motion, for example models of spirochetes as rotating helices [18] and cilia undergoing purely rotational motion [24], however for bodies which undergo significant flexible motion such as respiratory cilia and sperm flagella, this elimination is an approximation which must be justified by either post hoc numerical checks [22] or slender body theory analysis [28]. The resulting approximate single-layer boundary integral equation is then,

$$u_j(\boldsymbol{y}) \approx -\frac{1}{8\pi}\iint_{\partial D} S_{ij}^\epsilon(\boldsymbol{x},\boldsymbol{y})f_i(\boldsymbol{x})dS(\boldsymbol{x}). \tag{13}$$

In what follows we will treat the approximation as exact, however it should be borne in mind that there is error associated with both the continuous integral equation (13) in addition to the error associated with subsequent discretisation. In what follows we will find it convenient to use the identity $S_{ij}^\epsilon(\boldsymbol{x},\boldsymbol{y}) = S_{ji}^\epsilon(\boldsymbol{y},\boldsymbol{x})$; relabelling, and treating the approximation as exact we have,

$$u_i(\boldsymbol{x}) = -\frac{1}{8\pi}\iint_{\partial D} S_{ij}^\epsilon(\boldsymbol{x},\boldsymbol{y})f_j(\boldsymbol{y})dS(\boldsymbol{y}). \tag{14}$$

If the body motion is prescribed, the no-slip condition $\boldsymbol{u}(\boldsymbol{x}) = \dot{\boldsymbol{x}}$ can be applied on the surface ∂D to convert equation (14) to a Fredholm first kind integral equation for the unknown force distribution $\boldsymbol{f}(\boldsymbol{y})$ – a *resistance problem*.

$$\dot{x}_i = -\frac{1}{8\pi}\iint_{\partial D} S_{ij}^\epsilon(\boldsymbol{x},\boldsymbol{y})f_j(\boldsymbol{y})dS(\boldsymbol{x}) \quad \text{all} \quad \boldsymbol{x}\in\partial D. \tag{15}$$

If the body is rigid, or its surface velocity is known up to a rigid body motion, and the total force and moment $\boldsymbol{F}$, $\boldsymbol{M}$ are known, the result is the *mobility problem*,

$$\begin{aligned} \dot{x}_i + U_i + \epsilon_{ijk}\Omega_j x_k &= -\frac{1}{8\pi}\iint_{\partial D} S_{ij}^\epsilon(\boldsymbol{x},\boldsymbol{y})f_j(\boldsymbol{y})dS(\boldsymbol{y}) \quad \text{all} \quad \boldsymbol{x}\in\partial D, \\ F_i &= \iint_{\partial D} f_i(\boldsymbol{y})dS(\boldsymbol{y}), \\ M_i &= \iint_{\partial D} \epsilon_{ijk}y_j f_k(\boldsymbol{y})dS(\boldsymbol{y}), \end{aligned} \tag{16}$$

where the rigid body velocity $\boldsymbol{U}$ and angular velocity $\boldsymbol{\Omega}$, and the force distribution $\boldsymbol{f}(\boldsymbol{y})$, are unknown; ϵ_{ijk} is the Levi-Civita alterating tensor. The mobility problem arises from situations such as a sedimenting body (for which the force is given by gravity or centrifugal force and the moment is zero), or a swimming cell in the inertialess regime of Stokes flow (for which the force and moment are both zero).

To solve the problems (15) and (16), the method of numerical discretisation described by Cortez et al. [18] and used in the majority of studies to date takes advantage of the regularity of the S_{ij}^{ϵ} kernel and directly approximates the surface integrals with a quadrature rule followed by collocation on the quadrature points. The result is a system such as,

$$\dot{x}_i[m] = \frac{1}{8\pi}\sum_{n=1}^{N} S_{ij}^{\epsilon}(\boldsymbol{x}[m], \boldsymbol{x}[n]) g_j[n] A[n], \tag{17}$$

for the resistance problem, where $(\boldsymbol{x}[n], A[n])$ are quadrature nodes and weights, and $g_j[n] = -f_j(\boldsymbol{x}[n])$. For the mobility problem, we have,

$$\begin{aligned} \dot{x}_i[m] &= \frac{1}{8\pi}\sum_{n=1}^{N} S_{ij}^{\epsilon}(\boldsymbol{x}[m], \boldsymbol{x}[n]) g_j[n] A[n], \quad \text{for} \quad m = 1, \ldots, N, \\ F_i &= \sum_{n=1}^{N} g_i[n] A[n], \\ M_i &= \sum_{n=1}^{N} \epsilon_{ijk} x_j[n] g_k[n] A[n]. \end{aligned} \tag{18}$$

The above approach has the principal advantage of computational simplicity, and the principal disadvantage that the degrees of freedom of the resulting linear system are tied to the quadrature required to approximate the rapidly-varying kernel $S_{ij}^{\epsilon}(\boldsymbol{x}, \boldsymbol{X})$ for $|\boldsymbol{x} - \boldsymbol{X}| = O(\epsilon)$ – and associated high computational expense for a given level of accuracy.

Boundary element methods take an alternative approach to numerical discretisation – to discretise the unknown density $\boldsymbol{f}(\boldsymbol{y})$ with basis functions $\Phi_n(\boldsymbol{y})$, i.e. $\boldsymbol{f}(\boldsymbol{y}) = -\sum_{n=1}^{N} \boldsymbol{g}[n]\Phi_n(\boldsymbol{y})$. The integral operator can then be written as,

$$-\iint_{\partial D} S_{ij}^{\epsilon}(\boldsymbol{x}, \boldsymbol{y}) f_j(\boldsymbol{y}) dS(\boldsymbol{y}) = \sum_{n=1}^{N} g_j[n] \iint_{\partial D} S_{ij}^{\epsilon}(\boldsymbol{x}, \boldsymbol{y}) \Phi_n(\boldsymbol{y}) dS(\boldsymbol{y}). \tag{19}$$

In the simplest 'constant force' implementation, the basis functions $\{\Phi_1, \ldots, \Phi_N\}$ are indicator functions on the elements of the mesh $\{E_1, \ldots, E_N\}$. The stokeslet integrals are then decoupled from the force discretisation, and can be subjected to suitably fine spatial discretisation as appropriate, without unnecessarily increasing the number of degrees of freedom in the system – a major saving in both computational storage and time. This approach was suggested in the context of regularized stokeslet methods by Smith [22], and subsequently applied to problems in developmental biology [24,25] and sperm cell motion [29]. The practical drawback of this method is the need to generate a true surface mesh, which for complex geometries may be time-consuming.

To retain the advantages of both approaches – ease of implementation and computational efficiency – we suggest an alternative approach based on nearest-neighbour interpolation.

4. Nearest-neighbour discretisation of the regularized stokeslet boundary integral

Suppose that we have two surface discretisations of ∂D, $\{\boldsymbol{x}[1], \ldots, \boldsymbol{x}[N]\}$ and $\{\boldsymbol{X}[1], \ldots, \boldsymbol{X}[Q]\}$, which we will refer to as the *force discretisation* and *quadrature discretisation* respectively. These discretisations are not true meshes because they are not equipped with a mapping from nodes to elements, and we will not need to evaluate integrals in local coordinate systems. In general, $N \ll Q$ because the kernel $S_{ij}^{\epsilon}(\boldsymbol{x}, \boldsymbol{y})$ varies much more rapidly than the surface traction $\boldsymbol{f}(\boldsymbol{y})$.

Provided that they do not vary rapidly relative to the force points, the force $\boldsymbol{f}(\boldsymbol{y})$ and surface metric $dS(\boldsymbol{y})$ may then be discretised using nearest-neighbour interpolation. Denote by $\mathcal{N} : \{1, \ldots, Q\} \to \{1, \ldots, N\}$ the nearest-neighbour discretisation such that,

$$\mathcal{N}(q) := \underset{n=1,\ldots,N}{\operatorname{argmin}} |\boldsymbol{x}[n] - \boldsymbol{X}[q]|, \tag{20}$$

so that $f_j(\boldsymbol{X}[q])dS(\boldsymbol{X}[q]) \approx f_j(\boldsymbol{x}[\mathcal{N}(q)])dS(\boldsymbol{x}[\mathcal{N}(q)]) =: -g_j[\mathcal{N}(q)]A[\mathcal{N}(q)]$. The nearest-neighbour operator $\mathcal{N}$ can be expressed as a $Q \times N$ matrix,

$$\nu[q,\hat{n}] = \begin{cases} 1 & \text{if} \quad \hat{n} = \underset{n=1,\ldots,N}{\operatorname{argmin}} |\boldsymbol{x}[n] - \boldsymbol{X}[q]|, \\ 0 & \text{otherwise}, \end{cases} \tag{21}$$

so that $g_i[\mathcal{N}(q)]A[\mathcal{N}(q)] = \sum_{n=1}^{N} \nu[q,n] g_i[n] A[n]$.

With the above discretisation, the regularized stokeslet boundary integral may be approximated as,

$$\begin{aligned} -\iint_{\partial D} S_{ij}^{\epsilon}(\boldsymbol{x},\boldsymbol{y}) f_j(\boldsymbol{y}) dS(\boldsymbol{y}) &\approx -\sum_{q=1}^{Q} S_{ij}^{\epsilon}(\boldsymbol{x},\boldsymbol{X}[q]) f_j(\boldsymbol{x}[\mathcal{N}(q)]) A[\mathcal{N}(q)], \\ &= \sum_{q=1}^{Q} S_{ij}^{\epsilon}(\boldsymbol{x},\boldsymbol{X}[q]) \sum_{n=1}^{N} \nu[q,n] g_j[n] A[n]. \end{aligned} \tag{22}$$

Applying the discretisation (22) to the boundary integral equation (14), followed by performing collocation on the force discretisation $\boldsymbol{u}(\boldsymbol{x}[m]) = \dot{\boldsymbol{x}}[m]$, leads to the discretised resistance problem,

$$\dot{x}_i[m] = \frac{1}{8\pi} \sum_{n=1}^{N} g_j[n] A[n] \sum_{q=1}^{Q} S_{ij}^{\epsilon}(\boldsymbol{x}[m],\boldsymbol{X}[q]) \nu[q,n], \tag{23}$$

and mobility problem,

$$\begin{aligned} \dot{x}_i[m] &= \frac{1}{8\pi} \sum_{n=1}^{N} g_j[n] A[n] \sum_{q=1}^{Q} S_{ij}^{\epsilon}(\boldsymbol{x}[m],\boldsymbol{X}[q]) \nu[q,n], \quad \text{for} \quad m = 1,\ldots,N, \\ F_i &= \sum_{n=1}^{N} g_i[n] A[n] \sum_{q=1}^{Q} \nu[q,n], \\ M_i &= \sum_{n=1}^{N} g_k[n] A[n] \sum_{q=1}^{Q} \epsilon_{ijk} X_j[q] \nu[q,n]. \end{aligned} \tag{24}$$

The discrete resistance problem (22) can be written as a $3N \times 3N$ linear system $\mathsf{A}\mathsf{f} = \mathsf{b}$, where the unknown $3N$-vector f has components,

$$\mathsf{f}[N(j-1)+n] = g_j[n] A[n], \tag{25}$$

the $3N \times 3N$ left hand side matrix A has components,

$$\mathsf{A}[N(i-1)+m, N(j-1)+n] = \frac{1}{8\pi} \sum_{q=1}^{Q} S_{ij}^{\epsilon}(\boldsymbol{x}[m],\boldsymbol{X}[q]) \sum_{n=1}^{N} \nu[q,n], \tag{26}$$

and the right hand side velocity is given by,

$$\mathsf{b}[N(i-1)+m] = \dot{x}_i(\boldsymbol{x}[m]). \tag{27}$$

The discrete mobility problem can be written similarly as a $3(N+2) \times 3(N+2)$ linear system, where A and b are augmented by six rows discretising the force and moment constraints, and f has six additional scalar unknowns representing the values of $\boldsymbol{U}$ and $\boldsymbol{\Omega}$.

The discrete problems (23) and (24) may be implemented in Matlab® or GNU Octave by assembling matrices representing $S_{ij}^{\epsilon}(\boldsymbol{x}[m],\boldsymbol{X}[q])$ and $\nu[q,n]$. Details are provided in Appendix A.

5. Numerical results and analysis

The core numerical codes for implementation of the method given by equations (25)–(27) are given in Appendices A.1–A.3. The full code (approximately 1000 lines) used to produce the results in this report is available from github at https://github.com/djsmithbham/NearestStokeslets. The quadrature weights are absorbed into the $g_i[n]$ and so are never calculated explicitly.

For numerical testing we will denote the maximum discretisation spacing (i.e. maximum distance of a point to its nearest-neighbour) by h_f for the force points and h_q for the quadrature points:

$$h_f = \max_{m=1,\ldots,N} \min_{\substack{n=1,\ldots,N \\ n\neq m}} |\boldsymbol{x}[m] - \boldsymbol{x}[n]|$$
$$h_q = \max_{p=1,\ldots,Q} \min_{\substack{q=1,\ldots,Q \\ q\neq p}} |\boldsymbol{x}[p] - \boldsymbol{x}[q]|. \tag{28}$$

This parameter may be computed for a given discretisation as described in Appendix A.4.

5.1. Rigid body motion of a sphere

The simplest test case is perhaps Stokes' law for a translating or rotating sphere in an infinite fluid. Taking a sphere of radius 1 translating with velocity $\boldsymbol{U} = (1, 0, 0)$, the exact solution to the resistance problem yields total force $\boldsymbol{F} = (6\pi, 0, 0)$; rotation with velocity $\boldsymbol{\Omega} = (1, 0, 0)$ yields total moment $\boldsymbol{M} = (8\pi, 0, 0)$. Discretising the sphere by projecting onto the six faces of a cube yields the discretisations shown in Fig. 1 (a – force/collocation points, b – quadrature points).

Numerical experiments assessing the L_2 relative error in total force and moment compared with analytic solutions, for varying regularization parameter ϵ, force points h_f and quadrature points h_q, are shown in Tables 2, 6 and 7 and example computational timings are given in Appendix C, Table 9. The entries on the main diagonal ($h_f = h_q$) correspond to the Nyström discretisation; 'non-trivial' nearest-neighbour results are above the main diagonal ($h_f < h_q$). Results below the main diagonal correspond to more force points than quadrature points; in all cases the system is ill-conditioned (Table 3) and the Matlab® linear solver returns 'NaN' (not-a-number). Conditioning is generally not a problem provided that $h_f < h_q$, or if $h_f = h_q$ and the force and quadrature discretisations coincide. If $h_f = h_q$ and the discretisations are non-overlappling, singular matrices can result – data not shown).

It is immediately clear from examining the table rows that for fixed force discretisation spacing h_f, decreasing the quadrature discretisation spacing h_q typically results in improved accuracy, notwithstanding a slight reversal in this tendency which may occur for very coarse $h_f = 0.58$ and very fine $h_q < 0.02$. This behaviour can be interpreted as progressively finer h_q enabling more progressively more accurate quadrature, until the error is instead dominated by errors associated with force discretisation. An error estimate will follow in section 5.2.

Examining the columns of Tables 2 (see also Tables 6 and 7) reveals a more interesting behaviour of the algorithm. If the quadrature discretisation size h_q is fixed, more accurate results are obtained with the force spacing h_f taken *coarser* than the quadrature spacing ($h_f > h_q$) than with the Nyström method ($h_f = h_q$). Appendix C confirms that, for fixed h_q, the choice $h_f = h_q$ can be rather inaccurate, and is sensitive to the value of ϵ, whereas taking $h_f \approx 2h_q$ reliably produces results which are accurate to within a few percent, and at much lower computational cost (see Appendix C). A similar result is observed for the slightly more complex problem of calculating the resistance tensor of a prolate spheroid (Appendix D).

The effect of the regularization parameter is discussed in Appendix A.4. Reducing ϵ typically reduces the error for all finite ϵ tested, provided that $h_q < h_f/2$. The regularization error is proportional to ϵ, however it may be expected that as ϵ is reduced, h_q may have to be reduced proportionately in order to approximate the integral of the increasingly-peaked kernel more accurately. However, this behaviour was not observed in the test cases analysed (for which ϵ was taken as small as 10^{-6}). In applications in which evaluation of the velocity field is of interest, a balance between small regularization error and smooth/efficient evaluation of the velocity field may be sought, motivating an intermediate choice of ϵ.

The final quantity to consider is the force discretisation length h_f. This discretisation must be fine enough to resolve variations in the surface force density. The translating sphere case in fact is not a good way to assess this convergence, because the surface stress is constant [30, p. 233]! The rotating sphere does however possess a non-constant surface force density, which varies from zero at the poles to its maximum at the equator. From the results in Tables 6–7 it is clear that the coarsest force discretisation $h_f = 0.58$ produces acceptably accurate results (i.e. within about 1% error) provided that the quadrature discretisation is sufficiently fine.

5.2. Error estimate

Following these numerical experiments, we shall briefly outline an error estimate for the nearest-neighbour method. There are three sources of error: (i) regularization error associated with the use of the regularized version of the boundary integral equation with parameter ϵ – which was discussed above following equation (12), (ii) discretisation error associated with the approximation of the integral by its values on the quadrature points, which have spacing h_q, (iii) discretisation error associated with the approximation of the force and metric by their values on the coarser force points, which have spacing h_f.

The discretisation error associated with the approximation of the integral by its values on the quadrature points will be chiefly determined by the contribution associated with the rapid variation in the kernel. We will restrict to the case where $h_f \gg \epsilon$. The lowest order estimate of quadrature error follows from taking the mean value inequality, i.e. $|S_{jk}(\boldsymbol{x}, \boldsymbol{y}) - S_{jk}(\boldsymbol{x}, \boldsymbol{X}[q])| \leqslant M_1|(\boldsymbol{y} - \boldsymbol{X}[q])|$, where M_1 is a bound on $|\nabla_{\boldsymbol{y}} S_{jk}(\boldsymbol{x}, \boldsymbol{X}[q])|$. The integrand is sharply-peaked but in a small area – to take account of this behaviour more precisely, the integral will be split into three regions based on the value of

Table 1
Heuristic for calculating converged results.

1.	Choose ϵ much smaller than the lengthscale of the problem geometry L. Regularization error will typically be linear in ϵ, so results which are required to be highly accurate will require a proportionately small value of ϵ.
2.	Generate the force discretisation – initially this discretisation would be chosen relatively coarse.
3.	Generate the quadrature discretisation at least four times as fine as the force discretisation, i.e. h_q is no larger than $h_f/4$.
4.	Assess convergence by halving h_f, keeping h_q constant, and halving h_q, keeping h_f constant. Variations comparable to or smaller in magnitude than ϵ are considered acceptable. Larger variations are unacceptable; halve h_f and h_q and repeat until convergence.

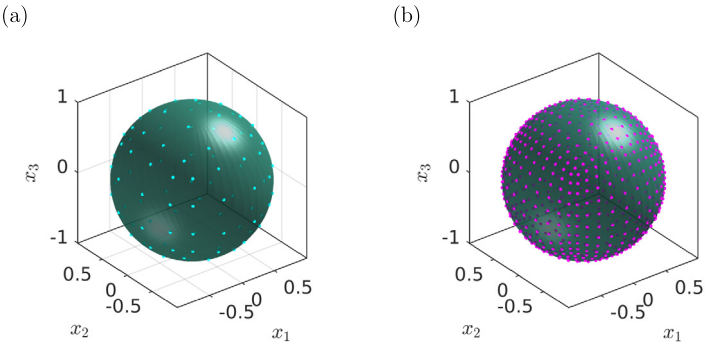

Fig. 1. Visualisation of discretisations on the surface of a sphere: (a) force/collocation points with $N = 96$ (4×4 subdivisions per face), (b) quadrature discretisation with $Q = 600$ (10×10 subdivisions per face.)

$r = |\boldsymbol{x} - \boldsymbol{y}|$, the regions (i) $0 < r < h_f$, (ii) $h_f < r < h_f^{1/2}$ and (iii) $h_f^{1/2} < r$, and the error estimated on each region in turn and summed.

(i) Considering first the 'near' part of the integral encountered around the collocation point, i.e. where $|\boldsymbol{x} - \boldsymbol{y}| \leqslant h_f$, and noting that the regularized stokeslet is dominated by the behaviour of $(r^2 + \epsilon^2)^{-1/2}$, the bound $M_1 = O(\epsilon^{-2})$ and so the error in the surface integral is $O(\epsilon^{-2} h_f^2 h_q)$, because the area of the region is $O(h_f^2)$ and the spacing between collocation points is $O(h_q)$.

(ii) In the intermediate region $h_f < |\boldsymbol{x} - \boldsymbol{y}| \leqslant h_f^{1/2}$, the bound $M_1 = O(h_f^{-2})$ over an area $O(h_f)$, yielding a quadrature error $O(h_f^{-1} h_q)$.

(iii) For the outer region $h_f^{1/2} < |\boldsymbol{x} - \boldsymbol{y}|$, the bound $M_1 = O(h_f^{-1})$ and the area is $O(1)$, giving a quadrature error $O(h_f^{-1} h_q)$ again.

The total discretisation error associated with quadrature can therefore be estimated as $O(\epsilon^{-2} h_f^2 h_q) + O(h_f^{-1} h_q)$. The first term may not be a sharp estimate; the results of Table 8 suggest that accurate results may be obtained (perhaps for certain types of discretisation) for very small ϵ compared with h_f and h_q. The second term emphasises the advantage of taking $h_f > h_q$, i.e. the force points coarser than the quadrature points.

Finally, the discretisation error associated with the approximation of the force and metric by their values on the force points can be estimated by noting that the error of nearest-neighbour interpolation is again of the form $M_2 |\boldsymbol{X}[q] - \boldsymbol{x}[\mathcal{N}(q)]|$, where M_2 is a bound on $\|\nabla_{\boldsymbol{y}}(\boldsymbol{f}(\boldsymbol{y}) dS(\boldsymbol{y}))\|$. Hence the force discretisation error is $O(h_f)$.

In summary, our estimate of the error associated with the regularization and nearest-neighbour discretisation of the boundary integral equation is $O(\epsilon) + O(\epsilon^{-2} h_f^2 h_q) + O(h_f^{-1} h_q) + O(h_f)$. The numerical results are consistent with the finding that there are independent errors due to regularization (see Appendix B) and to the force discretisation (see the rightmost column of Table 8 for which the regularization error is minimal); moreover it is advantageous to take $h_f^{-1} h_q$ to be small, i.e. $h_f > h_q$.

5.3. A refinement heuristic

For practical purposes we can therefore recommend the heuristic in Table 1: Discretisation convergence can then be assessed by (1) halving h_f while keeping h_q constant; (2) halving h_q while keeping h_f constant.

The heuristic in Table 1 can be applied to the rotating sphere problem as follows. We choose $\epsilon = 0.01$ as the regularization parameter, and consider numerical errors comparable to 1% acceptable. Taking a relatively coarse force discretisation with $h_f = 0.5796$ and a finer quadrature discretisation of $h_q = 0.0416$ – less than $h_f/4$ – we compute the total moment associated with the rigid body motion $\boldsymbol{\Omega} = (1, 0, 0)$. We then assess convergence by halving each of h_f and h_q. The results are shown in Table 4.

Table 2
Relative error for the resistance problem of a unit sphere undergoing rigid body motion in Stokes flow in an infinite fluid; regularization parameter $\epsilon = 0.01$. (a) Translation with velocity $\boldsymbol{U} = (1, 0, 0)$. (b) Rotation with angular velocity $\boldsymbol{\Omega} = (1, 0, 0)$.

(a)

			Q	864	3456	13824	55296	221184
			h_q	0.1611	0.0826	0.0416	0.0208	0.0104
N	DOF	h_f						
54	162	0.5796		0.0147	0.0052	0.0002	0.0006	0.0012
216	648	0.2942		0.0166	0.0079	0.0038	0.0022	0.0020
864	2592	0.1611		0.1262	0.0083	0.0043	0.0027	0.0025
3456	10368	0.0826		NaN	0.0277	0.0043	0.0028	0.0025

(b)

			Q	864	3456	13824	55296	221184
			h_q	0.1611	0.0826	0.0416	0.0208	0.0104
N	DOF	h_f						
54	162	0.5796		0.0300	0.0086	0.0019	0.0036	0.0047
216	648	0.2942		0.0378	0.0182	0.0095	0.0063	0.0058
864	2592	0.1611		0.2193	0.0194	0.0109	0.0078	0.0074
3456	10368	0.0826		NaN	0.0495	0.0110	0.0080	0.0075

Table 3
Condition number for the stokeslet matrix associated with the solution of the translation and rotation problems for a unit sphere in Stokes flow; regularization parameter $\epsilon = 0.01$.

		Q	864	3456	13824	55296	221184
N	DOF						
54	162		132.807	903.874	415.837	292.569	260.382
216	648		100.744	242.112	904.001	3344.192	1721.377
864	2592		8.129	218.433	638.954	4823.278	15637.232
3456	10368		Inf	38.309	617.308	3696.167	1870680.776

Table 4
Results from applying heuristic 1 to the problem of calculating the total moment on a unit sphere with unit angular velocity with $\epsilon = 0.01$. The result for $h_f = 0.5796$, $h_q = 0.0416$ is accurate to approximately 1% relative error. The result shown in parentheses would not be calculated via this heuristic, but confirms the accuracy of the method.

			Q	13824	55296
			h_q	0.0416	0.0208
N	DOF	h_f			
54	162	0.5796		25.0854	25.0430
216	648	0.2942		25.3707	(25.2904)

5.4. Rotational diffusion of a macromolecular structure

The technique will now be applied to a problem from bioinorganic chemistry: determining the rotational diffusion coefficient of a novel macromolecular structure. The scientific application of the calculations will be contained in a future collaborative publication. The structure can be modelled as three nanoscale rods with slightly different orientations, in close proximity, as shown in Fig. 2, moving together as a single rigid body. The rods are discretised by subdividing equally in angle, and equally along the length of the rods; the angle and length spacings are chosen based on a target distance in both axial and azimuthal directions.

The grand resistance tensor [16] is defined as the 6×6 matrix,

$$\mathcal{R} = \begin{pmatrix} R_{FU} & R_{F\Omega} \\ R_{MU} & R_{M\Omega} \end{pmatrix}, \tag{29}$$

where R_{FU} is the force-velocity resistance matrix, $R_{F\Omega}$ is the force-rotation coupling, R_{MU} is the moment-translation coupling and $R_{M\Omega}$ is the moment-rotation resistance. This matrix relates the force $\boldsymbol{F}$ and moment $\boldsymbol{M}$ exerted by a rigid body on a viscous fluid to the body's translational velocity $\boldsymbol{U}$ and angular velocity $\boldsymbol{\Omega}$,

$$\begin{pmatrix} \boldsymbol{F} \\ \boldsymbol{M} \end{pmatrix} = \mathcal{R} \begin{pmatrix} \boldsymbol{U} \\ \boldsymbol{\Omega} \end{pmatrix}. \tag{30}$$

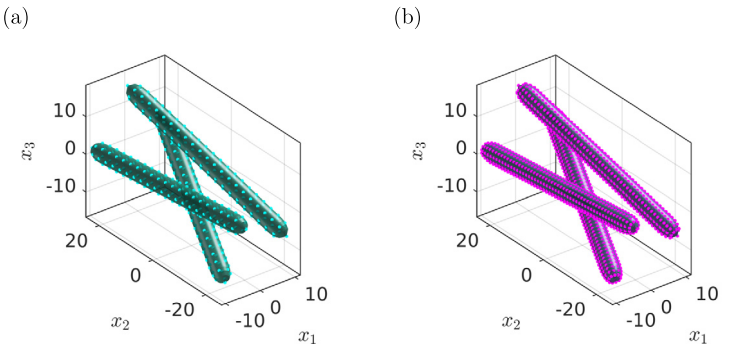

Fig. 2. Simplified representations of the macromolecular structure of interest, with (a) force discretisation ($N = 384$) and (b) quadrature discretisation ($Q = 1689$) shown.

Table 5
Calculation of the rotational diffusion timescale τ_1 for the macromolecular model shown in Fig. 2. The regularization parameter ϵ is taken as $0.01L$ where L is the approximate half-length of the peptide, 25 Å. The absolute temperature $T = 310$ K and dynamic viscosity $\mu = 10^{-3}$ Pa.s. Results with $h_f = h_q$ (second sub-diagonal) relate to the classic Nyström discretisation, results with $h_f < h_q$ relate to 'nearest-neighbour' discretisations. (a) Rotational diffusion timescale τ_1 in nanoseconds for each discretisation tested; discretisation parameters h_f and h_q are given in Å. (b) Computational timings (in seconds; notebook specification given in Appendix C).

(a)

		Q	3270	12678	49926
		h_q	1.5544	1.0543	0.7288
N	DOF	h_f			
246	738	3.7165	5.9021	6.0156	5.9831
870	2610	2.3598	5.7480	5.9255	5.9160
3270	9810	1.5544	5.6406	5.8281	5.8916

(b)

		Q	3270	12678	49926
N	DOF				
246	738		2.542	11.814	99.667
870	2610		10.526	34.312	173.740
3270	9810		125.025	316.446	742.062

The individual components of the 3×3 matrices $R_{\cdot\cdot}$ are calculated by solving the resistance problems $\boldsymbol{U} = \boldsymbol{e}_j$ and $\boldsymbol{\Omega} = \boldsymbol{e}_j$ in turn and calculating the force and moment in each case.

The diffusion tensor is given by $\mathcal{D} = k_B T \mathcal{R}^{-1}$, where k_B is the Boltzmann constant and T is absolute temperature. The rotational part of the diffusion tensor is the lower right 3×3 block of $\mathcal{D}$ [31], which we denote D_R,

$$\mathcal{D} = \begin{pmatrix} D_T & D_C' \\ D_C & D_R \end{pmatrix}. \tag{31}$$

The D_R block has no dependence on choice of origin [31] (unlike the other blocks of $\mathcal{D}$); it has been verified numerically that moving the origin does not affect the calculation of D_R.

It is convenient to report the smallest eigenvalue λ_1 of D_R, which corresponds to the smallest coefficient of rotational diffusion about each of the principal axes of rotation. The characteristic timescale of rotational diffusion is then given by $\tau_1 = 1/(6\lambda_1)$. The results are given in Table 5(a).

Starting in the top left corner, applying our heuristic, repeating the process of dividing both h_f and h_q yields the values given on the main diagonal. The point to terminate the refinement process depends on the degree of accuracy required, and indeed if a relative error of less than 1% is required, the process should be continued further. However for many biophysical applications, the level of modelling error (for example, approximating the structure by three straight rigid rods, assuming rigidity) does not warrant extremely precise numerical calculations. Fixing h_q and examining the first three columns, the nearest-neighbour method with coarser force discretisations $h_f \approx 1.5h_q$–$2.4h_q$ out-performs the Nyström method ($h_f = h_q$) for both accuracy and efficiency (Table 5(b)).

6. Conclusions

We have presented a simple-to-implement modification of the standard discretisation of the method of regularized stokeslets for modelling particle dynamics at zero Reynolds number. The modification is based on the use of two discretisations, one for the unknown surface force per unit area and one for the stokeslet quadrature, combined with nearest-neighbour discretisation of the force distribution. Practically, the method can be implemented by assembling a nearest-neighbour operator matrix, which can be achieved with a few lines of Matlab®/GNU Octave code. Numerical experiments on the resistance problem of a sphere undergoing rigid body motion, and the calculation of the rotational diffusion timescale of a macromolecular structure provide evidence that the method enables more accurate results to be obtained at lower computational cost than the standard implementation, despite not being substantially more complicated to implement. Our initial error estimate $O(\epsilon) + O(\epsilon^{-2}h_f^2 h_q) + O(h_f^{-1}h_q) + O(h_f)$ provides insight into the independent effects of h_f and h_q on the numerical error, and the potential advantage of taking $h_f > h_q$, provided that h_f is not too large. Numerical results did not however reflect the sensitivity to ϵ suggested by this estimate – further investigation of this phenomenon, and possible sharpening of the estimate, may be topics for future work.

The standard Nyström discretisation uses, in our framework, the same discretisation for the force and the quadrature. The present approach shows that this choice much less reliably produces accurate results than if the quadrature discretisation is kept the same but the force discretisation is made twice as coarse. Making the force discretisation twice as coarse means that the number of degrees of freedom is at least halved (more typically reduced by a factor of four). The matrix assembly cost is therefore reduced by a factor of at least four, and the linear solver cost is reduced by a factor of at least eight (for a direct solver). Our practical results suggest a cost reduction of over 10 times may be typical. This reduction in cost means that more complex problems can be solved with a given computational resource – a useful facet, particularly within biological and biophysical fluid dynamics. The code implementation described in this paper makes use primarily of basic linear algebra operations rather than serial for-loops, and therefore can be accelerated through built-in software and hardware parallelisation of these operations. It will be of interest to explore how the algorithm scales on multicore or GPU hardware.

The nearest-neighbour approach still has limitations, particularly if compared with boundary element methods – which may involve higher order force discretisation and adaptive quadrature, and accelerations such as the fast multipole method. However, the nearest-neighbour approach is very simple to implement, requiring only a small modification of the standard regularized stokeslet approach, and not requiring true mesh generation. It may be valuable to explore further whether adaptivity or fast multipole implementations can be introduced without excessively complicating the algorithm. Finally, we do not yet have theoretical results which definitively prove the improved efficiency and accuracy of the method. Nevertheless, for practical purposes, carrying out a sequence of discretisations with $h_f \approx 4h_q$ alongside a sequence with $h_f \approx 2h_q$ will establish convergence empirically.

The nearest-neighbour discretisation of the regularized stokeslet method is more efficient and accurate than the standard implementation, with minimal additional complexity. It may therefore enable researchers in biological and biophysical fluid dynamics to solve significantly more challenging open problems, for example involving many swimming cells, ciliated cavities, and/or suspended macromolecules. The task of explaining the properties of the method, which we are only able to explain heuristically at present, may stimulate theoretical work. Finally, the technique may also open the way for future algorithmic developments which possess the efficiency and accuracy of boundary element methods but retain its useful properties of meshlessness and simplicity.

Acknowledgements

This research was supported by Engineering and Physical Sciences Research Council grants EP/K007637/1 and EP/N021096/1. The author acknowledges Drs Anna Peacock and Sarah Newton (University of Birmingham) for suggesting the macromolecular diffusion problem in section 5.4, valuable discussions about regularized stokeslet methods and diffusion tensor calculations with Dr Rudi Schuech (University of Lincoln), and valuable comments from two anonymous referees).

Appendix A. Matlab®/GNU Octave implementation

The essentials of the Matlab®/GNU Octave implementation are given below, in particular some more subtle aspects such as the assembly of the nearest-neighbour matrix, avoidance of extensive for-loops, and use of 'blocking' to avoid memory overrun.

A.1. Regularized stokeslet matrix

Taking advantage of the vectorisation capabilities of the Matlab® language and the Kronecker product operator,

```
function S=RegStokeslet(x,X,ep)
    % x is a vector of field points:  3*M
    % X is a vector of source points: 3*Q
```

```
% ep is regularisation parameter
% outputs an array of regularized stokeslets between field
%        and source points
% blocks are [Sxx, Sxy, Sxz; Syx, Syy, Syz; Szx, Szy, Szz]
%        where Sxx is M by Q etc.
x=x(:);
X=X(:);
M=length(x)/3;
Q=length(X)/3;
r1=      x(1:M)*ones(1,Q)-ones(M,1)*      X(1:Q)';
r2=  x(M+1:2*M)*ones(1,Q)-ones(M,1)*  X(Q+1:2*Q)';
r3=x(2*M+1:3*M)*ones(1,Q)-ones(M,1)*X(2*Q+1:3*Q)';
rsq=r1.^2+r2.^2+r3.^2;
irep3=1./(sqrt((rsq+ep^2)).^3);
isotropic=kron(eye(3),(rsq+2.0*ep^2).*ireps3);
dyadic=[r1.*r1 r1.*r2 r1.*r3; r2.*r1 r2.*r2 r2.*r3; ...
        r3.*r1 r3.*r2 r3.*r3].*kron(ones(3,3),irep3);
S=(1.0/(8.0*pi))*(isotropic+dyadic);
```

A.2. Nearest-neighbour matrix

The nearest-neighbour operator ν may be discretised with the matrix `NN` produced by the following code,

```
function NClosest=NearestNeighbourMatrix(X,x,varargin)
% Vectors should be supplied with all x1 coordinates listed
% first then all x2 coordinates, then all x3 coordinates.
% if varargin is nonempty, then it should contain
% blockSize
Q=length(X)/3;N=length(x)/3;
if ~isempty(varargin)
    blockSize=varargin{1};
    blockNodes=floor(blockSize*2^27/(9*N));
else
    blockNodes=Q;
end
xQ1=X(1:Q);
xQ2=X(Q+1:2*Q);
xQ3=X(2*Q+1:3*Q);
xT1=x(1:N);
xT2=x(N+1:2*N);
xT3=x(2*N+1:3*N);
nMin=zeros(Q,1);
for iMin=1:blockNodes:Q
    iMax=min(iMin+blockNodes-1,Q);
    blockCurr=iMax-iMin+1;
    X1=xQ1(iMin:iMax)*ones(1,N)-ones(blockCurr,1)*xT1';
    X2=xQ2(iMin:iMax)*ones(1,N)-ones(blockCurr,1)*xT2';
    X3=xQ3(iMin:iMax)*ones(1,N)-ones(blockCurr,1)*xT3';
    distsq=X1.^2+X2.^2+X3.^2;
    [~,nMin(iMin:iMax)]=min(distsq,[],2);
end
NClosest=sparse(Q,N); % creates sparse all-zero matrix
NClosest([1:Q]'+Q*(nMin-1))=1;
NClosest=kron(speye(3),NClosest);
```

The above takes advantage of the speed of predominantly vector operations, whilst not exceeding the memory requirements of the system. The optional third argument, `blockSize` is a measurement in GB of the memory to be allocated to the regularized stokeslet matrix so that

```
blockNodes=floor(blockSize*2^27/(9*N));
```

gives the number of columns (corresponding to a subset of the force points) which can be dealt with simultaneously. For example, `blockSize=0.2` would be suitable for any modern hardware, and has been tested on a Raspberry Pi Model B.

The matrix `NClosest`, which corresponds to $v[q,n]$ is sparse and so will not produce a memory overflow. The final line involving the Kronecker product operation is required because the nearest-neighbour operator must be copied into three blocks, acting on the f_1, f_2 and f_3 components in turn.

A.3. Resistance problem

The 'left hand side' matrix A for the discrete resistance problem (26) can then be assembled as,

```
A = RegStokeslet(x,X,ep)*NearestNeighbourMatrix(X,x);
```

The regularized stokeslet matrix may be too large to fit in memory, particularly if Q is very large, as may be the case for problems possessing complex geometry. In this case, the problem may be assembled 'block-by-block' as follows,

```
NN=NearestNeighbourMatrix(X,x,blockSize);
A=zeros(3*M,3*N);
for iMin=1:blockNodes:Q
    iMax=min(iMin+blockNodes-1,Q);
    iRange=[iMin:iMax Q+iMin:Q+iMax 2*Q+iMin:2*Q+iMax];
    A=A+RegStokeslet(x,X(iRange),ep)*NN(iRange,:);
end
```

As in the function `NearestNeighbour`, blocking is used to prevent overrun. In all calculations in the present report, the linear system was solved with the 'backslash' operator, i.e. `f=A`.

A.4. Discretisation size calculation

The discretisation size parameters h_f and h_q are calculated using the following function,

```
function [h,hMin,nMin,distsq] = CalcDiscr_h(x)
% CalcDiscr_h This function calculates the maximum over all
%       points in a discretisation x of the distance to
%       the nearest-neighbour point
N=length(x)/3;
X1=x(1:N)*ones(1,N)-ones(N,1)*x(1:N)';
X2=x(N+1:2*N)*ones(1,N)-ones(N,1)*x(N+1:2*N)';
X3=x(2*N+1:3*N)*ones(1,N)-ones(N,1)*x(2*N+1:3*N)';
distsq=X1.^2+X2.^2+X3.^2+100*eye(N);
[hMin,nMin]=min(distsq,[],2);
h=sqrt(max(hMin));
```

Appendix B. Effect of the regularization parameter for the rigid sphere test problem

To assess the sensitivity of the method to regularization parameter, we present test results with $\epsilon = 0.02$ and $\epsilon = 0.005$ in Tables 6 and 7 respectively. When h_f and h_q are taken equal, the results are highly sensitive to the value of ϵ, however provided h_q is taken no larger than $0.25h_f$, the error is relatively insensitive. As ϵ is reduced, the finite regularization error (evident in the rightmost entries in the tables) is reduced to below 1%, however convergence to the smaller error with h_q is slower. Perhaps surprisingly, it does not appear necessary to choose h_q dependent on ϵ, at least within the range of values explored. There also does not appear to be any clear advantage to taking $\epsilon = 0.02$ as opposed to $\epsilon = 0.005$, begging the question of how small ϵ can be taken. While $\epsilon = 0$ is equivalent to non-regularized stokeslets (and hence singular matrix entries whenever the collocation and quadrature points coincide), taking a very small but finite value of $\epsilon = 10^{-6}$ yields the results of Table 8, which are typically at least as accurate as the results with larger value of ϵ, provided that $h_f \geqslant 2h_q$.

Appendix C. Timing results for the rigid sphere test problem

Typical timing results (in seconds) for the solution of the translation and rotation resistance problems (with $\epsilon = 0.01$) computed on a modest notebook computer (2011 Lenovo Thinkpad X220; Intel(R) Core(TM) i5-2520M CPU @ 2.50GHz; 8GB DDR3 RAM) are given in Table 9.

Appendix D. Testing the method on a prolate spheroid

To explore further whether the efficiency of the choice $h_f \approx 2h_q$ is problem-dependent, we may assess the performance of the nearest-neighbour method in calculating the grand resistance tensor $\mathcal{R}$ (defined in equation (29)) of a rigid prolate spheroid, which has a well-known analytical solution [32, p. 64]. Taking a prolate spheroid with long semi-axis $a = 5$ and

Table 6
Relative error for the resistance problem of a unit sphere undergoing rigid body motion in Stokes flow in an infinite fluid; regularization parameter $\epsilon = 0.02$. (a) Translation with velocity $\boldsymbol{U} = (1, 0, 0)$. (b) Rotation with angular velocity $\boldsymbol{\Omega} = (1, 0, 0)$. 'NaN' denotes 'not-a-number', and indicates a singular linear system.

(a)

			Q	864	3456	13824	55296	221184
			h_q	0.1611	0.0826	0.0416	0.0208	0.0104
N	DOF	h_f						
54	162	0.5796		0.0149	0.0056	0.0016	0.0017	0.0014
216	648	0.2942		0.0167	0.0083	0.0052	0.0046	0.0046
864	2592	0.1611		0.0522	0.0087	0.0056	0.0051	0.0051
3456	10368	0.0826		NaN	0.0046	0.0057	0.0052	0.0051

(b)

			Q	864	3456	13824	55296	221184
			h_q	0.1611	0.0826	0.0416	0.0208	0.0104
N	DOF	h_f						
54	162	0.5796		0.0313	0.0114	0.0034	0.0036	0.0030
216	648	0.2942		0.0391	0.0210	0.0147	0.0136	0.0135
864	2592	0.1611		0.0917	0.0222	0.0161	0.0151	0.0150
3456	10368	0.0826		NaN	0.0032	0.0162	0.0152	0.0152

Table 7
Relative error for the resistance problem of a unit sphere undergoing rigid body motion in Stokes flow in an infinite fluid; regularization parameter $\epsilon = 0.005$. (a) Translation with velocity $\boldsymbol{U} = (1, 0, 0)$. (b) Rotation with angular velocity $\boldsymbol{\Omega} = (1, 0, 0)$.

(a)

			Q	864	3456	13824	55296	221184
			h_q	0.1611	0.0826	0.0416	0.0208	0.0104
N	DOF	h_f						
54	162	0.5796		0.0147	0.0051	0.0000	0.0013	0.0023
216	648	0.2942		0.0165	0.0079	0.0036	0.0016	0.0008
864	2592	0.1611		0.2424	0.0082	0.0041	0.0021	0.0013
3456	10368	0.0826		NaN	0.0691	0.0041	0.0021	0.0014

(b)

			Q	864	3456	13824	55296	221184
			h_q	0.1611	0.0826	0.0416	0.0208	0.0104
N	DOF	h_f						
54	162	0.5796		0.0297	0.0079	0.0033	0.0062	0.0082
216	648	0.2942		0.0375	0.0176	0.0081	0.0037	0.0021
864	2592	0.1611		0.3876	0.0188	0.0095	0.0053	0.0038
3456	10368	0.0826		NaN	0.1262	0.0096	0.0054	0.0040

Table 8
Relative error for the resistance problem of a unit sphere undergoing rigid body motion in Stokes flow in an infinite fluid; regularization parameter $\epsilon = 10^{-6}$. (a) Translation with velocity $\boldsymbol{U} = (1, 0, 0)$. (b) Rotation with angular velocity $\boldsymbol{\Omega} = (1, 0, 0)$.

(a)

			Q	864	3456	13824	55296	221184
			h_q	0.1611	0.0826	0.0416	0.0208	0.0104
N	DOF	h_f						
54	162	0.5796		0.0147	0.0051	0.0000	0.0014	0.0027
216	648	0.2942		0.0165	0.0079	0.0036	0.0015	0.0004
864	2592	0.1611		0.9994	0.0082	0.0040	0.0020	0.0009
3456	10368	0.0826		NaN	0.9977	0.0041	0.0020	0.0010

(b)

			Q	864	3456	13824	55296	221184
			h_q	0.1611	0.0826	0.0416	0.0208	0.0104
N	DOF	h_f						
54	162	0.5796		0.0296	0.0077	0.0037	0.0071	0.0100
216	648	0.2942		0.0374	0.0174	0.0077	0.0028	0.0004
864	2592	0.1611		0.9997	0.0186	0.0091	0.0044	0.0021
3456	10368	0.0826		NaN	0.9988	0.0092	0.0046	0.0023

Table 9
Timing results for the calculation of the translation and rotation resistance problems for the unit sphere with $\epsilon = 0.01$.

		Q	864	3456	13824	55296	221184
N	DOF						
54	162		0.052	0.455	5.288	81.071	1319.765
216	648		0.287	0.882	6.471	84.248	1318.965
864	2592		5.773	8.726	15.953	105.991	1399.280
3456	10368		352.833	320.650	373.632	535.916	2160.078

Table 10
Test results for the grand resistance tensor of rigid body motion of a prolate spheroid with long semi-axis 5, short semi-axis 1; regularization parameter $\epsilon = 0.01$.

			Q	864	3456	13824	55296	221184
			h_q	0.2171	0.1117	0.0554	0.0278	0.0139
N	DOF	h_f						
54	162	1.0064		0.0596	0.0221	0.0096	0.0045	0.0052
216	648	0.4305		0.0459	0.0180	0.0079	0.0036	0.0021
864	2592	0.2171		0.3675	0.0207	0.0097	0.0049	0.0034
3456	10368	0.1117		NaN	0.1171	0.0099	0.0052	0.0036

short semi-axis $c = 1$, the relative error in $\mathcal{R}$ in the $\|\cdot\|_2$ norm is given in Table 10. The results are not likely to be optimal as the discretisation has been created by simply deforming the sphere discretisation depicted in Fig. 1 without any attempt to space the points uniformly in the directions of the long and short semi-axes.

References

[1] G.I. Taylor, Analysis of the swimming of microscopic organisms, Proc. R. Soc. Lond. A 209 (1951) 447–461.
[2] G.J. Hancock, The self-propulsion of microscopic organisms through liquids, Proc. R. Soc. Lond. B 217 (1953) 96–121.
[3] J. Gray, G.J. Hancock, The propulsion of sea urchin spermatozoa, J. Exp. Biol. 32 (1955) 802–814.
[4] A.T. Chwang, T.Y. Wu, A note on the helical movement of micro-organisms, Proc. R. Soc. Lond. B 178 (1052) (1971) 327–346.
[5] E. Lauga, T.R. Powers, The hydrodynamics of swimming microorganisms, Rep. Prog. Phys. 72 (2009) 096601.
[6] J.J.L. Higdon, A hydrodynamic analysis of flagellar propulsion, J. Fluid Mech. 90 (1979) 685–711.
[7] R.E. Johnson, C.J. Brokaw, Flagellar hydrodynamics: a comparison between resistive-force theory and slender-body theory, Biophys. J. 25 (1979) 113–127.
[8] N. Phan-Thien, T. Tran-Cong, M. Ramia, A boundary-element analysis of flagellar propulsion, J. Fluid Mech. 185 (1987) 533–549.
[9] M. Ramia, D.L. Tullock, N. Phan-Thien, The role of hydrodynamic interaction in the locomotion of microorganisms, Biophys. J. 65 (1993) 755–778.
[10] H. Power, G. Miranda, Second kind integral equation formulation of Stokes' flows past a particle of arbitrary shape, SIAM J. Appl. Math. 47 (4) (1987) 689–698.
[11] L.a. Klinteberg, A.-K. Tornberg, A fast integral equation method for solid particles in viscous flow using quadrature by expansion, J. Comput. Phys. 326 (2016) 420–445.
[12] A.Z. Zinchenko, R.H. Davis, An efficient algorithm for hydrodynamical interaction of many deformable drops, J. Comput. Phys. 157 (2) (2000) 539–587.
[13] S.K. Veerapaneni, D. Gueyffier, D. Zorin, G. Biros, A boundary integral method for simulating the dynamics of inextensible vesicles suspended in a viscous fluid in 2D, J. Comput. Phys. 228 (7) (2009) 2334–2353.
[14] S.K. Veerapaneni, A. Rahimian, G. Biros, D. Zorin, A fast algorithm for simulating vesicle flows in three dimensions, J. Comput. Phys. 230 (14) (2011) 5610–5634.
[15] E. Nazockdast, A. Rahimian, D. Zorin, M. Shelley, A fast platform for simulating semi-flexible fiber suspensions applied to cell mechanics, J. Comput. Phys. 329 (2017) 173–209.
[16] C. Pozrikidis, A Practical Guide to Boundary Element Methods with the Software Library BEMLIB, CRC, 2002.
[17] R. Cortez, The method of regularized Stokeslets, SIAM J. Sci. Comput. 23 (4) (2001) 1204–1225.
[18] R. Cortez, L. Fauci, A. Medovikov, The method of regularized Stokeslets in three dimensions: analysis, validation, and application to helical swimming, Phys. Fluids 17 (2005) 031504.
[19] J. Ainley, S. Durkin, R. Embid, P. Boindala, R. Cortez, The method of images for regularized Stokeslets, J. Comput. Phys. 227 (2008) 4600–4616.
[20] R. Cortez, F. Hoffmann, A fast numerical method for computing doubly-periodic regularized Stokes flow in 3D, J. Comput. Phys. 258 (2014) 1–14.
[21] R. Cortez, D. Varela, A general system of images for regularized Stokeslets and other elements near a plane wall, J. Comput. Phys. 285 (2015) 41–54.
[22] D.J. Smith, A boundary element regularized Stokeslet method applied to cilia- and flagella-driven flow, Proc. R. Soc. Lond. A 465 (2009) 3605–3626.
[23] D.J. Smith, A.A. Smith, J.R. Blake, Mathematical embryology: the fluid mechanics of nodal cilia, J. Eng. Math. 70 (2011) 255–279.
[24] A.A. Smith, T.D. Johnson, D.J. Smith, J.R. Blake, Symmetry breaking cilia-driven flow in the zebrafish embryo, J. Fluid Mech. 705 (2012) 26–45.
[25] P. Sampaio, R.R. Ferreira, A. Guerrero, P. Pintado, B. Tavares, J. Amaro, A.A. Smith, T. Montenegro-Johnson, D.J. Smith, S.S. Lopes, Left-right organizer flow dynamics: how much cilia activity reliably yields laterality?, Dev. Cell 29 (6) (2014) 716–728.
[26] C. Pozrikidis, Boundary Integral and Singularity Methods for Linearized Viscous Flow, Cambridge Univ. Press, 1992.
[27] D.J. Smith, E.A. Gaffney, J.R. Blake, J.C. Kirkman-Brown, Human sperm accumulation near surfaces: a simulation study, J. Fluid Mech. 621 (2009) 289–320.
[28] R. Cortez, M. Nicholas, Slender body theory for Stokes flows with regularized forces, Commun. Appl. Math. Comput. Sci. 7 (1) (2012) 33–62.
[29] T.D. Montenegro-Johnson, H. Gadêlha, D.J. Smith, Spermatozoa scattering by a microchannel feature: an elastohydrodynamic model, Open Sci. 2 (3) (2015) 140475.
[30] G.K. Batchelor, An Introduction to Fluid Dynamics, Cambridge University, 1967.
[31] W.A. Wegener, Diffusion coefficients for rigid macromolecules with irregular shapes that allow translation–rotation coupling, Biopolymers 20 (1981) 303–326.
[32] S. Kim, S.J. Karrila, Microhydrodynamics: Principles and Selected Applications, Butterworth-Heinemann, Boston and London, 2013.