Dual Reciprocity Method for studying thermal flows related to Magma Oceans

Tyler Drombosky
Ph.D. Student, Applied Mathematics Scientific Computation
Department of Mathematics
University of Maryland, College Park, MD
drombosk@math.umd.edu

Dr. Sawata Hier-Majumder
Assistant Professor, Department of Geology
University of Maryland, College Park, MD
sawata@umd.edu

Abstract

The tools to computationally model crystals settling in a magma ocean are currently not readily available. Being able to model such behavior may provide clues into the history of early Earth. New numerical simulations could lead to a better understanding of the settling. The equations that describe this settling are well understood, however, no suitable software package is currently available to solve such problems. This project will use the Dual Reciprocity Method, an extension of Boundary Element Method, to generate numerical solutions to a coupled system of Stokes flow and heat equations. The Dual Reciprocity Method allows for free boundary conditions as well as a reduction in problem dimensionality which decreases computation. The final product will combine algorithms from several papers to produce a robust, modular, and highly optimized software package.

1 Introduction

Earth's early history is marked by a giant impact with a Mars-sized object which led to the formation of the moon [1]. This impact event led to a substantial amount of melting of the Earth's interior. Subsequent cooling of the Earth involved extensive crystallization in this 'magma ocean' over a relatively short period of time. While the chemical evidence from ancient sources provide some clues on the rate of cooling, computational models of such phenomenon are sparse.

Constraints on the mode of crystal settling come from laboratory experiments [2], parameterized heat flux calculations [3][4][5][6], and chemical constraints [7]. The first two lines of evidence suggest extremely small crystals, contrary to common belief, also settle likely enhancing the efficiency of cooling. Direct estimation of heat flux and crystal size, however, are still not available. Chemical arguments suggest that the pattern of density-driven settling is also controlled by the vast pressures associated with a plant-scale magma ocean.

Modeling this physical behavior requires solving a coupled system of partial differential equations (PDEs), specifically the Stokes flow and heat equation. As closed form solutions rarely exist for problems of interest, numerical solutions are an attractive option. Methods for solving various types of PDEs, while being robust, require PDEs to conform to certain forms and restrictions.

When solving PDEs involving coupled Stokes flow and heat transfer, it is standard practice to use a finite element method (FEM) solver. FEM algorithms are very robust and can generally converge with reasonable amounts of computation. Unfortunately there are two issues with FEM that make it unpractical for the above problem. First, as the crystals settle, the region of flow will be changing. This free boundary problem means the region would have to be rediscretized at every time step. This becomes extremely costly, especially in high dimensions. In addition to discretization issues, FEM is $\mathcal{O}(n^3)$ in $\mathbb{R}^3$. While this project will only focus on domains in $\mathbb{R}^2$, using a FEM algorithm would not scale well if adapted to work in $\mathbb{R}^3$.

Boundary Element Method (BEM) is an alternative technique for solving the same problem. The goal of BEM is to write the weak form of the PDEs employing reciprocal relations, integrating by parts, and making specific choices on test functions such that the integrals move entirely to the boundary. This has the immediate advantage of reducing the dimensionality by one. Also, BEM naturally handles free boundary problems. Unlike the domain meshes need for FEM, BEM discretizes the boundary. For any evaluation, only three easily computable pieces about the boundary are need. First is the position of the node. Second, is the boundary value at the node, this can be Dirichlet or Neumann boundary condition or a prescribed condition such as the no-slip boundary condition for Stokes flow. Third is the vector normal to the boundary at the node. This can be quickly be approximated by using a local interpolation of nodes.

2 Approach

Dual Reciprocity Method (DRM) is an algorithm based on BEM [8][9][10][11]. DRM, like BEM, provides an approximation on the boundary rather than in the region. This provides a reduction in dimensionality. This makes the problem of discretization of the region significantly easier. In the case of $\mathbb{R}^2$ regions, this means only having to discretize a curve. This reduction of dimensionality also allows for adding more mesh points with less computational penalty. DRM advantage over BEM is that it provides a way to move more complicated PDE to the boundary.

DRM works on PDEs in the form

$$\mathcal{D}u = b, \tag{1}$$

where $\mathcal{D}$ is a generic non-linear operator with variable coefficients. A linear differential operator $\mathcal{L}$ is chosen such that the fundamental solution to the adjoint operator $\mathcal{L}^*$ is known. Then $\mathcal{D}$ can be written as a sum of the linear operator $\mathcal{L}$ and a residual operator $\mathcal{D}'$. Then (1) can be written

$$\mathcal{L}u = b - \mathcal{D}'u =: b'. \tag{2}$$

The combination of the original source term and residual operator are now treated as a new source term for the linear PDE. Considering the example of the non-homogeneous heat equation

$$\frac{\partial}{\partial t}u + v_i u_{,i} - u_{,ii} = b \tag{3}$$

where v is the velocity in Ω computed by solving the Stokes equation. The operator can then be broken up into

$$\begin{aligned}
\mathcal{D} &= \frac{\partial}{\partial t} + v_i(\cdot)_{,i} - (\cdot)_{,ii} && (4)\\
\mathcal{L} &= -(\cdot)_{,ii} && (5)\\
\mathcal{D}' &= \frac{\partial}{\partial t} + v_i(\cdot)_{,i} && (6)\\
b' &= b - \frac{\partial}{\partial t}u - v_i u_{,i} && (7)
\end{aligned}$$

The boundary integral equation is then formulated for (3). This is done in the same manner as in BEM. Like traditional FEM, BEM multiplies a PDE by a test function and converts it into an integral equation. The formulation for the (3) is given

$$\begin{aligned}
-u_{,ii} &= b' \qquad x \in \Omega && (8)\\
-u_{,ii}w &= b'w && (9)\\
-\int_\Omega u_{,ii} \cdot w \ \mathrm{d}\Omega &= \int_\Omega b'w \ \mathrm{d}\Omega && (10)
\end{aligned}$$

where Ω is an open set. Then integration by parts is applied twice on the left hand side

$$\begin{aligned}
\int_\Omega u_{,ii} \cdot w \ \mathrm{d}\Omega &= \int_\Gamma w u_{,i} n_i \ \mathrm{d}\Gamma - \int_\Omega u_{,i} w_{,i} \ \mathrm{d}\Omega && (11)\\
&= \int_\Gamma w u_{,i} n_i \ \mathrm{d}\Gamma - \int_\Gamma u w_{,i} n_{,i} \ \mathrm{d}\Gamma + \int_\Omega u w_{,ii} \ \mathrm{d}\Omega && (12)
\end{aligned}$$

where $\Gamma = \bar{\Omega} \backslash \Omega$ is the boundary. BEM makes the specific choice of $w = \mathcal{G}^*(x - x_0)$ where $\mathcal{G}^*$ is the fundamental solution of the dual of the Laplacian. That is $\mathcal{G}^*$ satisfies

$$\mathcal{G}^*_{,ii}(x - x_0) = -\delta(x - x_0) \tag{13}$$

for a choice of $\xi \in \Omega$. Then plugging (12) and (13) into (10)

$$u(\xi) = \int_\Gamma (\mathcal{G}^* u_{,i} - u\mathcal{G}^*_{,i}) n_i \, \mathrm{d}\Gamma + \int_\Omega b' \mathcal{G}^* \, \mathrm{d}\Omega. \tag{14}$$

To this point, the method has not differed from BEM and in fact (14) could be solved numerically if the region was discretized for integration. However, the goal of boundary methods is to avoid discretizing the region. DRM provides a way to approximate the integral over the region as an integral over the boundary. In DRM the source term $b' = b - \mathcal{D}'u$ is approximated by

$$b' = \sum_{q=1}^{\mathcal{N}} f^q \alpha_q \tag{15}$$

where f^q are chosen basis functions and α_q are coefficients. The basis functions f^q are chosen appropriately so u^q the solution to

$$u^q_{,ii} = f^q, \tag{16}$$

can easily be found. This eliminates the need for finding a particular solution for b'. Instead it is approximated by a linear combination of particular solutions for f^q which can easily be found. Using this approximation (8) becomes

$$-u_{,ii} = \sum_{q=1}^{\mathcal{N}} f^q \alpha_q \tag{17}$$

and (14) becomes

$$\begin{aligned} u(\xi) &= \int_\Gamma (\mathcal{G}^* u_{,i} - u\mathcal{G}^*_{,i}) n_i \, \mathrm{d}\Gamma + \sum_{q=1}^{\mathcal{N}} \alpha_q \int_\Omega f^q \mathcal{G}^* \mathrm{d}\Omega & (18) \\ &= \int_\Gamma (\mathcal{G}^* u_{,i} - u\mathcal{G}^*_{,i}) n_i \, \mathrm{d}\Gamma - \sum_{q=1}^{\mathcal{N}} \alpha_q \int_\Omega u^q_{,ii} \mathcal{G}^* \mathrm{d}\Omega & (19) \end{aligned}$$

employing Green's second identity yields

$$\int_\Omega (u^q_{,ii}\mathcal{G}^* - \mathcal{G}^*_{,ii} u^q) \, \mathrm{d}\Omega = \int_\Gamma (u^q_{,i}\mathcal{G}^* - \mathcal{G}^*_{,i} u^q) n_i \, \mathrm{d}\Gamma \tag{20}$$

because of (13), (20) becomes

$$\int_\Omega u^q_{,ii}\mathcal{G}^* \, \mathrm{d}\Omega = -u^q(\xi) + \int_\Gamma (u^q_{,i}\mathcal{G}^* - \mathcal{G}^*_{,i} u^q) n_i \, \mathrm{d}\Gamma \tag{21}$$

Substituting (21) into (14) yields the Dual Reciprocity equation

$$u(\xi) = \int_\Gamma (\mathcal{G}^* u_{,i} - u\mathcal{G}^*_{,i}) n_i \, \mathrm{d}\Gamma + \sum_{q=1}^{\mathcal{N}} \left(u^q(\xi) - \int_\Gamma (u^q_{,i}\mathcal{G}^* - \mathcal{G}^*_{,i} u^q) n_i \, \mathrm{d}\Gamma \right) \alpha^q \tag{22}$$

It is given this name because two reciprocity equations are used, once to convert the integral with the linear term to the boundary and then again to convert the approximation of the source term to the boundary. The advantage of DRM is that all integrals now are on the boundary.

This same procedure can be applied to the Stokes flow equation. The PDE in differential form is given by

$$-P_{j,i} + \frac{1-\lambda}{1+\lambda} v_{j,ii} + \rho b_j = 0 \tag{23}$$

Where P is the pressure, λ is the ratio of viscosity between the particle and fluid, ρ is density of the fluid, and b is the body force. After converting the integral form and moving the integration to the boundary, the boundary element formulation of Stokes flow becomes

$$v_j(x_0) = \int_\Gamma f_{,i} \mathcal{J}_{ij} \, \mathrm{d}\Gamma + \frac{1-\lambda}{1+\lambda} \int_\Gamma n_k \mathcal{K}_{ijk} v_i \, \mathrm{d}\Gamma \tag{24}$$

where f represents the surface tension and buoyancy, $\mathcal{J}$ a second order tensor, and $\mathcal{K}$ is a third order tensor. The second order tensor $\mathcal{J}$ is a Green's function for the singularity forced Stokes equation and the third order tensor $\mathcal{K}$ is stress tensor associated with the Green's function.

Since u depends on v but v does not depend on u, a solution for v is approximated first using (24) then this approximation is used in computing u using (22).

Once the boundary integral equations are derived, variables non-dimensionalized and the integrals are discretized. The discretization will be done using the method of collocation. In this method, the integral is broken up into N disjoint, connected segments $\Gamma_1, \ldots, \Gamma_n$ such that $\Gamma_i \cap \Gamma_j = \emptyset$ for $i \neq j$ and $\cup_{i=1}^N \Gamma_i = \Gamma$. Next, $x_i \in \Gamma_i$ are selected and used as the ξ values. This yields a system of N equations and N unknowns that can be solved.

Moving ξ to the boundary presents an issue. As stated earlier $\xi \in \Omega$, however to use the method of collocation, ξ must be moved to the boundary. This can be done in a limiting process. This results in both weakly and strongly singular integrals. When integrating on Γ_i with respect to the collocation point $x_j \notin \Gamma_i$, the integral is non-singular and is approximated using an eight point Gauss quadrature method. If $x_i \in \Gamma_j$, the integral is singular and it can be solved using radial approximation methods [12][13].

3 Implementation

The software package will be developed in Fortran 90 with compiler level optimizations focused on single threaded x86 processors. Fortran 90 was chosen as the programming

language for several reasons. First, Fortran 90 is an object orientated language. Types and modules allow for use of custom data types and methods. This feature is crucial for constructing modular software. Second, there is a large library of Fortran 90 code developed. These packages, both widely distributed and local, are highly modular and will cut down considerably on development time. Finally, Fortran 90 has significant speed advantages, even when compared to other low level languages such as C.

The target platform for the software is a single threaded x86 system. The specific test system is a 4 node cluster. Each cluster houses two quad core processor with each processor sharing 8 gigabytes of main memory. While an instance of this software will not use more than one of the 32 cores, this system will be convenient if a need arises to run several time consuming simultaneous simultaneously. However, as developed, the software will be able to compile and run efficiently in any x86 environment.

The software will be created in several steps. First, existing code for solving the Poisson and Stokes equations in steady-state using BEM will be merged. The softwares were written at different times for different purposes. As is expected in this case, there are several software engineering challenges that must be solved involving unifying data structures. In addition, both solvers use similar libraries that are not always identical. While it would be possible to include both copies of these libraries, this approach is sloppy and does not align with the goal of making the software highly modular. Thus, these libraries will be standardized and merged as necessary.

After the Poisson and Stokes code are merged, work will begin on developing DRM code. The only difference between DRM and BEM is the transformation of the source term to the boundary. The code for DRM will accordingly be highly modular. Once written it can be added to the existing merged code with little effort.

As mentioned earlier, DRM requires computing integrals with strong and weak singularities. While these singularities can be computationally difficult, there are many documented methods for obtaining accurate approximations to such problematic integrals. However, in practice testing is needed to determine which ones are actually successful for approximating the specific integral.

Unlike FEM, the linear systems produced by DRM and BEM are not sparse. In fact, the matrix generated by DRM is, in general, dense and asymmetric. Because of this, solving the linear system will be done using a straightforward LU decomposition with pivoting for stability through the forward and backward substitution.

A solution to overcome the dense matrix is to employ the fast multipole method (FMM) [14]. One of the reasons that DRM has a dense matrix is because each node requires unique information from every other node. This is an $\mathcal{O}(N^2)$ process which creates the dense and asymmetric matrix. FMM creates poles throughout the region and then "connects" the nodes to the poles and then poles to poles in a tree structure. Data is then shared between nodes via the poles. Doing this requires $\mathcal{O}(N)$ to set up the tree. This creates a sparse matrix with $\mathcal{O}(N \log N)$ entries. With this sparsity the linear system can then be solved with more efficient and robust methods such as GMRES. If time permits a FMM module will be added to the software package.

The software will run the Stokes solver first followed by either a Poisson or heat equation solver depending on the development stage. Each solver will run in the same order of time. As presented the algorithm will run in $\mathcal{O}(N^3)$ time where N is the number of discrete nodes on the boundary. This is dominated by the LU decompositions. Evaluating each of the $\mathcal{O}(N^2)$ integrals takes constant time. When the integral is non-singular, eight point Gauss quadrature is used. This quadrature has experimentally shown to provide a balance between accuracy and speed. For the approximation of the singular integrals, a radial integration method is used which takes constant time with respect to the number of node points. Since the linear system depends on the position of the nodes on the boundary, the formation and solving of the linear systems must be performed at every time step since the particles are constantly subject to advection.

Using FMM, the runtime of the entire software runtime reduces to $\mathcal{O}(N \log N)$ as setting up FMM takes one time work of $\mathcal{O}(N)$ and $\mathcal{O}(N \log N)$ is required for computing the entries in the mass matrix. Solving the linear system can then be done with sparse methods that converge in $\mathcal{O}(N)$ time.

3.1 Poisson Solver

Consider the Poisson equation

$$u_{,ii} = b \tag{25}$$

Applying BEM to the equation we obtain

$$u(x_0) = \int_\Gamma (\mathcal{G}^* u_{,i} - u\mathcal{G}^*_{,i}) n_i \, \mathrm{d}\Gamma + \int_\Omega b\mathcal{G}^* \, \mathrm{d}\Omega \tag{26}$$

Recalling that $\mathcal{G}^*$ satisfies (13), the free space Greens function for two dimensions is chosen

$$\mathcal{G}^*(x - x_0) = -\frac{\ln r}{2\pi} \tag{27}$$

$$\mathcal{G}^*_{,i}(x - x_0) = -\frac{x_i - x_{0i}}{2\pi r} \tag{28}$$

where $r = |x - x_0|$. To move the source term integral to the boundary, b is forced to be harmonic. A higher order function $\mathcal{K}^*$ is defined such that

$$-\mathcal{K}^*_{,ii} = \mathcal{G}^* \tag{29}$$

Then $\mathcal{K}^*$ is the solution to the forced biharmonic equation. It can be shown that for 2D, $\mathcal{K}^*$ is given by

$$\mathcal{K}^* = \frac{r^2}{8\pi}(\ln r - 1) \tag{30}$$

which also leads to

$$\mathcal{K}^*_{,i} = \frac{1}{8\pi}(2 \ln r - 1)(x_i - x_{0i}) \tag{31}$$

Then

$$b\mathcal{K}^*_{,ii} = -[b\mathcal{K}^*_{,i} - b_{,i}\mathcal{K}^*]_{,i} \tag{32}$$

Substituting (27), (28), (30), (31) and (32) into (26), results in

$$\begin{aligned} u(x_0) &= \int_\Gamma \left[\frac{u(x_i - x_{0i})n_i}{2\pi r} - \frac{q\ln r}{2\pi}\right] \mathrm{d}\Gamma \\ &\quad + \int_\Gamma \left[b(2\ln r - 1)(x_i - x_{0i}) - b_{,i}r^2(\ln r - 1)\right] \frac{n_i}{8\pi}\, \mathrm{d}\Gamma \end{aligned} \tag{33}$$

where $q = u_{,i}n_i$ is the flux across the boundary in the direction of the normal. With this form we can begin discretization and assembly of the final linear system. Discretizing (33) for the j-th boundary element, while the pole x_0 is located at the beginning of the i-th node.

$$\begin{aligned} \check{u}_i &= \sum_j \int_{\Gamma_j} \left[\frac{u(x_k - \check{x}_{ik})n_k}{2\pi r} - q\frac{\ln r}{2\pi}\right] d\Gamma_j \\ &\quad + \sum_j \int_{\Gamma_j} \left[b(2\ln r - 1)(x_k - \check{x}_{ik})\frac{n_k}{8\pi} - b_{,k}n_k\frac{r^2}{8\pi}(\ln r - 1)\right] d\Gamma_j \end{aligned} \tag{34}$$

Where $\check{u}_i$ indicates the nodal value at the i-th node and $r = |x - x_i|$. For integration, the boundary is approximated using cubic spline interpolates. Using cubic interpolation (34) can be written as integrals over the local coordinate $\xi \in [-1, 1]$.

$$\check{u}_i = \check{u}_j M_{ij} + \check{u}_{j+1}N_{ij} + \check{\mathcal{G}}_j P_{ij} + \check{\mathcal{G}}_{j+1}Q_{ij} + \sum_j R_{ij} \tag{35}$$

where

$$M_{ij} = \int_{-1}^{1} (1-\xi)\frac{(x_k - x_{ik})n_k}{4\pi r} J_j d\xi \tag{36}$$

$$N_{ij} = \int_{-1}^{1} (1+\xi)\frac{(x_k - x_{ik})n_k}{4\pi r} J_j d\xi \tag{37}$$

$$P_{ij} = \int_{-1}^{1} \left(\frac{-1}{2}\right)(1-\xi)\frac{\ln r}{2\pi} J_j d\xi \tag{38}$$

$$Q_{ij} = \int_{-1}^{1} \left(\frac{-1}{2}\right)(1+\xi)\frac{\ln r}{2\pi} J_j d\xi \tag{39}$$

$$R_{ij} = R^1_{ij} + R^2_{ij} \tag{40}$$

$$R^1_{ij} = \int_{-1}^{1} \left[\frac{2b(x_k - \check{x}_{ik})n_k}{8\pi} - \frac{b_{,k}n_k r^2}{8\pi}\right] \ln r J_j d\xi \tag{41}$$

$$R^2_{ij} = \int_{-1}^{1} \left[-b(x_k - \check{x}_{ik})n_k + b_{,k}n_k r^2\right] \frac{J_j}{8\pi} d\xi \tag{42}$$

where J_j is the Jacobian for the transformation between the global coordinate x and local coordinate ξ. The integrals can now be evaluated numerically. The majority of the integrals are regular and thus can be evaluated using Gaussian quadrature. However, when $i = j$ or $i + 1 = j$, $r = 0$ on Γ_j causing M_{ij}, N_{ij} to become strongly singular and P_{ij}, Q_{ij} and R_{ij} to become weakly singular. It can be shown analytically that the integrals are finite, however, the singularity prevents the use of Gaussian quadrature. For singular integrals, the radial integration method described in section 3.3 will be used.

Once the integrals are computed, the final linear system can be assembled with (35) becoming

$$G\check{U} = H\check{q} + R \tag{43}$$

where

$$G_{ij} = \delta_{ij} - M_{ij} - N_{ij-1} \tag{44}$$
$$H_{ij} = P_{ij} + Q_{ij-1} \tag{45}$$

The indices are periodic since the boundary is closed. This system can be solved numerically using LU decomposition as long as either the potential or flux is known at each node.

3.2 Stokes Solver

This project will use an existing solver to compute the velocities from the Stokes equation. Since this code will not be modified, the discretization and specifics of integration have been omitted. However, they are covered in [12]. In general discretization is performed in a similar manner to the Poisson equation. The Green's function and associated stress tensor for Stokes equation are

$$\mathcal{J}_{ij}(\hat{x}) = \frac{\delta_i j}{r} + \frac{\hat{x}_i \hat{x}_j}{r^3} \tag{46}$$
$$\mathcal{T}_{ijk}(\hat{x}) = -6\frac{\hat{x}_i \hat{x}_j \hat{x}_k}{r^5} \tag{47}$$

where $\hat{x} = x - x_0$. The no-slip boundary condition will be used. This states that the relative velocity between a point on the boundary of a particle and the fluid is zero. In the case when the particle is ridged, numerical errors may occur, producing velocities along the boundary of the particle that suggest the particle is deforming. We regulate this behavior by averaging the velocity over all boundary points on the boundary and using the average velocity to step the particle forward in time.

3.3 Radial Integration Method

To handle the singular integrals encountered in both the Poisson and Stokes solve, the Radial Integration Method [13] will be used. Consider the singular integral

$$I_j = \int_{\Gamma_j} \frac{\bar{f}_j(x, x_0)}{r^\beta(x, x_0)} d\Gamma_j \tag{48}$$

Within the element, x and y are functions of the arc length s. The local coordinate is also a linear function of s. For the j-th element,

$$s = \frac{1}{2}(1-\xi)s_j + \frac{1}{2}(1+\xi)s_{j+1} \tag{49}$$

$$s = \frac{1}{2}(s_j + s_{j+1}) + \frac{1}{2}(s_{j+1} - s_j)\xi \tag{50}$$

Also define

$$\Delta s_j = s_{j+1} - s_j \tag{51}$$

$$A = \frac{s_{j+1} - s}{\Delta s_j} = \frac{1-\xi}{2} \tag{52}$$

$$B = \frac{s - s_j}{\Delta s_j} = \frac{1+\xi}{2} \tag{53}$$

$$C = \frac{1}{6}(A^3 - A)\Delta s_j^2 \tag{54}$$

$$D = \frac{1}{6}(B^3 - B)\Delta s_j^2 \tag{55}$$

The second derivatives at the nodes need to be determined such that $\frac{d^2x}{ds^2}|_{x=x_j}$ and $\frac{d^2y}{ds^2}|_{y=y_j}$ are known.

Interpolate x and y in the following way

$$x = A\check{x}_j + B\check{x}_{j+1} + C\check{x}''_j + D\check{x}''_{j+1} \tag{56}$$

$$y = A\check{y}_j + B\check{y}_{j+1} + C\check{y}''_j + D\check{y}''_{j+1} \tag{57}$$

Combining $(51-55)$, (56) and (57) the first and second derivatives of x and y with respect to s can be calculated. The Jacobian for the j-th element is given by

$$J_j = \left(\left(\frac{dx}{d\xi} \right)^2 + \left(\frac{dy}{d\xi} \right)^2 \right)^{\frac{1}{2}} \tag{58}$$

Now, $d\Gamma_j = J_j d\xi$

Define

$$r = \sqrt{(x-x_0)^2 + (y-y_0)^2} = \sqrt{(x-x_j)^2 + (y-y_j)^2} \tag{59}$$

Notice that

$$dr = \left[\frac{\partial r}{\partial x}\frac{dx}{d\xi} + \frac{\partial r}{\partial y}\frac{dy}{d\xi} \right] d\xi \tag{60}$$

One can easily show that

$$\nabla r = \hat{r} = \frac{x - x_j}{r}\hat{i} + \frac{y - y_j}{r}\hat{j} \tag{61}$$

Also, $\frac{dx}{d\xi}$ and $\frac{dy}{d\xi}$ are the x and y components of the tangent vector (not the unit tangent vector which is based on the first derivatives of x and y with respect to s). Thus the tangent is given by

$$\hat{t} = \left[\frac{dx}{d\xi}\hat{i} + \frac{dy}{d\xi}\hat{j}\right] \propto \frac{1}{J_j} \tag{62}$$

and

$$d\Gamma_j = \frac{dr}{\hat{r}\cdot\hat{t}} \tag{63}$$

(63) is the heart of the radial integration.

Making the substitution in the $(j-1)$-st element yields

$$I_{j-1} = \int_{r_{j-1}}^{r_j} \frac{\bar{f}_{j-1}}{r^\beta}\frac{dr}{\hat{r}\cdot\hat{t}} \tag{64}$$

and similarly for the j-th element

$$I_j = \int_{r_j}^{r_{j+1}} \frac{\bar{f}_j}{r^\beta}\frac{dr}{\hat{r}\cdot\hat{t}} \tag{65}$$

When added together, (64) and (65) yield a Cauchy Principal Value Integral.

The integrals still contain the $r^{-\beta}$ singularity. To smooth the singularity write

$$\frac{\bar{f}_j}{\hat{r}\cdot\hat{t}} = \sum_{n=0}^{N} C_n r^n \tag{66}$$

To determine the coefficients C_n, first create a uniform grid between the node and the pole.

1. Define a vector with $N+1$ elements such that

$$\xi_i = 1 - 2\left(\frac{i-1}{N}\right) \qquad i = 1, 2, \ldots, N+1 \tag{67}$$

2. Define vectors x and y containing global coordinates using $(51-55)$ to evaluate A, B, C, D and then (56) and (57) to determine x and y.

3. Calculate r_i at each x_i and y_i using equation (59). Calculate $\frac{dx}{d\xi}$ and $\frac{dy}{d\xi}$ at each point using the relation

$$\frac{dx}{d\xi} = \left(\frac{\breve{x}_{j+1}-\breve{x}_j}{2}\right) - \left(\frac{3A_i^2-1}{12}\right)\Delta s^2 x_j'' + \left(\frac{3B_i^2-1}{12}\right)\Delta s^2 x_{j+1}'' \tag{68}$$

$$\frac{dy}{d\xi} = \left(\frac{\breve{y}_{j+1}-\breve{y}_j}{2}\right) - \left(\frac{3A_i^2-1}{12}\right)\Delta s^2 y_j'' + \left(\frac{3B_i^2-1}{12}\right)\Delta s^2 y_{j+1}'' \tag{69}$$

Calculate $\hat{t}_i$ from (68) and (69).

4. Evaluate $\hat{r} \cdot \hat{t}$ and $\bar{f}$ at each point x_i.

5. Solve (66) to obtain C_i for $i = 0, 1, \ldots, N$.

This yields

$$I_j = \sum_{n=0}^{N} C_n \frac{r^{n-\beta+1}}{n-\beta+1} \qquad n - \beta \neq 0 \tag{70}$$

$$= \ln(r) \qquad n - \beta + 1 = 0 \tag{71}$$

If the original integral contains a logarithmic singularity, i.e. it looks like

$$J = \int_{\Gamma_j} \frac{\bar{f} \ln r}{r^\beta} d\Gamma_j \tag{72}$$

Then the integral is given by

$$J_j = \sum_{n=0}^{N} C_n \frac{((n+1-\beta)\ln r + 1)}{(n+1-\beta)^2} r^{n-\beta+1} \qquad n - \beta + 1 \neq 0 \tag{73}$$

$$= \frac{1}{2}[\ln(r)]^2 \qquad n - \beta + 1 = 0 \tag{74}$$

4 Validation and Testing

The software will be validated and tested at two points in the development cycle. The first time will be after the Poisson and Stokes solvers have been merged to create a steady-state Stokes flow and heat solver. As the name of the solver implies, it will be able to solve steady state equations for temperature gradients in a flow. The solver will first be tested against problems where there is no heat source and then in problems where there is no flow.

For an isoviscous region, the steady state-solver degrades to a Poisson solver. There exist analytic solutions to the Poisson equation on a circle which can be used to validate this part of the code. Testing of the Stokes solver will follow a similar testing method. With an isothermal region the solver becomes a Stokes solver. For constant surface tension, analytical solutions to the Stokes equations can be found. With a heat source and flow, asymptotic approximations using matching asymptotes provide an approximation of steady state thermal flow for low Péclet numbers [15]. This approximation can be used to gauge first order accuracy of the numerical solutions.

Once the DRM code is implemented a similar round of validation and testing will occur. A heat equation problem and transient stokes flow problem will be approximated. In both cases, analytic solutions and numerical solutions are available from other sources. Once these two elements of the DRM have been tested, a transient problem with both flow and a source term will be approximated. At this point no analytic or asymptotic solution exists. For validation a function $\tilde{u}$ will be created randomly with appropriate initial and boundary

conditions. Then $\tilde{b} := \mathcal{L}\tilde{u}$ is computed. Finally, the solver is used to approximate to solution to $\mathcal{L}u = \tilde{b}$ with the chosen initial and boundary conditions. The approximation of u can then be compared to the exact solution $\tilde{u}$. In this way the convergence and compute scaling can also be tested.

4.1 Analytical Solutions to Poisson Equation

The following solution can be used to validate the numerical solution to the Poisson equation obtained by BEM. Consider the Poisson equation

$$u_{,ii} = b \tag{75}$$

where b itself is a harmonic function such that

$$b_{,ii} = 0 \tag{76}$$

First, start by constructing a variable separable b,

$$b = \rho(r)\Theta(\theta) \tag{77}$$

Then

$$\Delta b = \frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial b}{\partial r}\right) + \frac{1}{r^2}\frac{\partial^2 b}{\partial \theta} \tag{78}$$

becomes

$$\Delta b = \frac{\Theta}{r}\frac{d}{dr}\left(r\frac{d\rho}{r}\right) + \frac{\rho}{r^2}\frac{d^2\Theta}{d\theta^2} \tag{79}$$

Now demand Θ to be such that

$$\frac{d^2\Theta}{d\theta^2} = -\Theta \tag{80}$$

Then (76) and (77) yield,

$$\frac{1}{r}\frac{d}{dr}\left(r\frac{d\rho}{dr}\right) - \frac{\rho}{r^2} = 0 \tag{81}$$

Force ρ to look like

$$\rho = r^n \tag{82}$$

Then (81) yields

$$(n^2 - 1)r^{n-2} = 0 \tag{83}$$

which has two solutions

$$n = \pm 1 \tag{84}$$

First take $n = 1$. Also pick

$$\Theta = a_1 \sin\theta + a_2 \cos\theta \tag{85}$$

Thus

$$b = \frac{1}{r}(a_1 \sin\theta + a_2 \cos\theta) \tag{86}$$

Next proceed to find a solution for u, which is also required to be separable. Write this as

$$u = R(r)T(\theta) \tag{87}$$

Requiring as before

$$\frac{d^2T}{d\theta^2} = -T \tag{88}$$

resulting in

$$\Delta u = T\left[\frac{1}{r}\frac{d}{dr}\left(r\frac{dR}{dr}\right) - \frac{R}{r^2}\right] \tag{89}$$

putting back into (75) it is shown that

$$T\left[\frac{1}{r}\frac{d}{dr}\left(r\frac{dR}{dr}\right) - \frac{R}{r^2}\right] = \frac{1}{r}(a_1 \sin\theta + a_2 \cos\theta) \tag{90}$$

Assume

$$T = \lambda(a_1 \sin\theta + a_2 \cos\theta) \tag{91}$$

and

$$R = r^m \tag{92}$$

Then (90) leads to

$$\lambda(m^2 - 1)r^{m-2} = r^{-1} \tag{93}$$

since λ and m are constants, $m = 1$ and $\lambda \to \infty$. Thus, this value of b wont work. Select $n = 1$ and recreate

$$b = r(a_1 \sin\theta + a_2 \cos\theta) \tag{94}$$

Still requiring

$$T = \lambda(a_1 \sin\theta + a_2 \cos\theta) \tag{95}$$

and

$$R = r^m \tag{96}$$

it is found that

$$\lambda(m^2 - 1)r^{m-2} = r \tag{97}$$

Which yields

$$m = 3 \qquad \lambda = \frac{1}{8} \tag{98}$$

Thus the solution to the biharmonic equation is given by

$$u = \frac{r^3}{8}(a_1 \sin\theta + a_2 \cos\theta) \tag{99}$$

The flux, q is given by

$$q = \nabla u \cdot \hat{n} \tag{100}$$

If the boundary is a circle of radius a, then on the boundary

$$x = a\cos\theta \qquad y = a\sin\theta \tag{101}$$

The normals are given by

$$\hat{n} = \left[\frac{d^2x}{ds^2}\hat{i} + \frac{d^2y}{ds^2}\hat{j}\right] \Bigg/ \left[\left(\frac{d^2x}{ds^2}\right)^2 + \left(\frac{d^2y}{ds^2}\right)^2\right]^{\frac{1}{2}} \tag{102}$$

$$ds = \left(\left(\frac{d^2x}{d\theta^2}\right)^2 + \left(\frac{d^2y}{d\theta^2}\right)^2\right)^{\frac{1}{2}} d\theta \tag{103}$$

Thus

$$\frac{d\theta}{ds} = \frac{1}{a} \tag{104}$$

$$\frac{dx}{ds} = -\sin\theta \tag{105}$$

$$\frac{dy}{ds} = \cos\theta \tag{106}$$

and

$$\frac{d^2x}{ds^2} = -\frac{1}{a}\cos\theta \tag{107}$$

$$\frac{d^2y}{ds^2} = -\frac{1}{a}\sin\theta \tag{108}$$

So

$$\hat{n} = -(\cos\theta\hat{i} + \sin\theta\hat{j}) \tag{109}$$

On the boundary

$$u = \frac{r^2}{8}(a_1 r\sin\theta + a_2 r\cos\theta) \tag{110}$$

or

$$u = \frac{a^2}{8}(a_1 y + a_2 x) \tag{111}$$

Thus

$$q = \frac{\partial u}{\partial x}n_x + \frac{\partial u}{\partial y}n_y \tag{112}$$

is

$$q = -\frac{a}{8}(a_2 x + a_1 y) \tag{113}$$

Thus the BEM calculation can be benchmarked by prescribing q on the boundary by (113) and using (94) for b, and matching the solution with (111).

4.2 Validation of Poisson Solver

To validate the Poisson portion of the solver, the boundary conditions and source term are given by (113) and (94) respectively. Then the analytical solution can be computed using (111). Table 1 shows the relative error between the approximated solution and the analytical solution using the L_∞ with $N = 100$ boundary elements.

In all cases it is observed that the relative error is less than $2 \cdot 10^{-1}$ indicating that each approximation has at least one digits of accuracy. In the case where the source term is radially symmetric there are two digits of accuracy for the approximation.

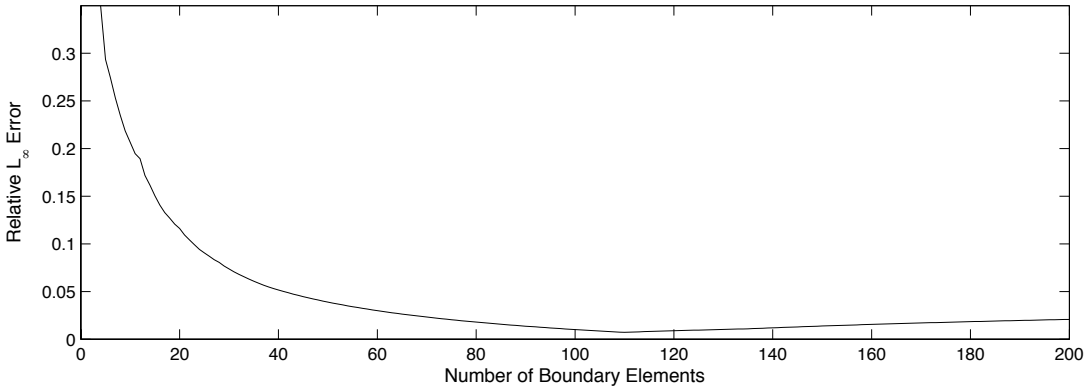
Next, convergence of the approximation is examined. For convergence, $a_1 = a_2 = 0.5$ is considered. One way to increase accuracy of the solution is to increase the number of boundary elements and nodes used in the discretization. As these increase, the error should approach zero. From a theoretical perspective, taking the number of boundary elements to infinity will produce the exact solution. However, in practice, the computational time and

Table 1: Verification of approximation for specific solutions ($N = 100$)

a_1	1.00	0.75	0.50	0.25	0.00
a_2	0.00	0.25	0.50	0.75	1.00
Relative L_∞ Error	.1905	.1211	.0101	.1207	.1907
Relative L_2 Error	.1900	.1206	.0027	.1190	.1887

roundoff errors make the limiting process impossible. Instead convergence to the analytical solution will be observed up until some finite number of boundary elements at which time either the linear system will be too large to solve, or round off errors will begin quantitatively detracting for the approximation. In this experiment, the number of boundary elements is ranged from $N = 4$ to $N = 200$ with the relative L_∞ and relative L_2 error computed for each N.

Figure 1: Relative L_∞ error with $a_1 = a_2 = 0.5$



Figures 1 and 2 show that the error quadratically decreases until $N \approx 100$ at which point there is no significant additional accuracy gained from the extra resolution. The stagnate convergence comes from the fact that many of the entries of the linear system approach machine epsilon as the number of boundary elements is increased. This roundoff error prevents the additional accuracy in the linear system to contributing to the final result.

Finally, convergence can be improved by increasing the number of quadrature points when approximating the regular, weakly singular, and strongly singular integrals. Unlike boundary elements, taking the number of quadrature points to infinity does not provide convergence to the analytical solution since the approximation will still only be over a finite number of boundary elements. However, if the number of boundary elements is fixed, using additional quadrature points may increase accuracy to the approximation.

Figure 2: Relative L_2 error with $a_1 = a_2 = 0.5$

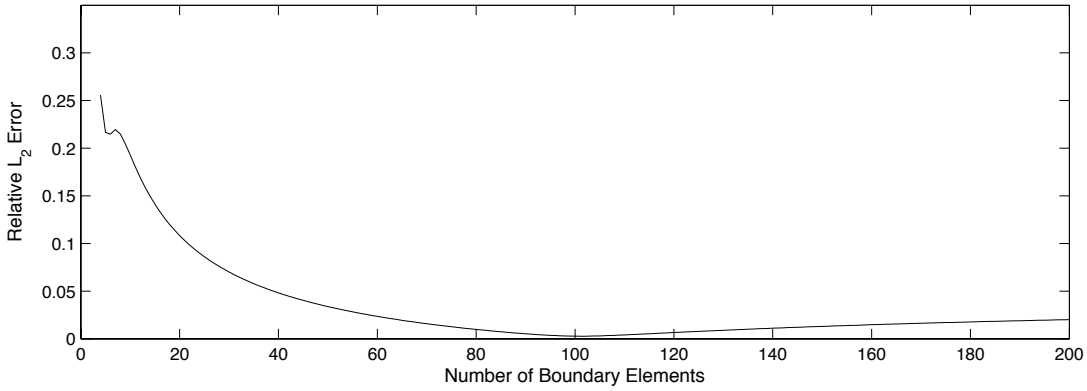


Table 2: Error against number of quadrature points with $a_1 = a_2 = 0.5$ and$N = 25$

N_{quad}	3	4	5	6	7	8
Relative L_∞ Error	.09057	.09009	.08855	.08805	.08721	.08684
Relative L_2 Error	.08617	.08572	.08420	.08372	.08288	.08253

Since an increase number of quadrature points alone cannot lead to complete convergence to the analytical solution, plotting error versus the number of quadrature points, N_{quad} for a fix N should show the error decreasing but after a finite number of quadrature points become stagnate. Table 2 shows the relative L_∞ and L_2 norms for $N_{quad} = 3$ to $N_{quad} = 8$ with $N = 100$ fixed. Adding more quadrature points does reduce the error, but not in any significant way. The effect is even less so when a higher number of boundary elements is used.

The run time of the algorithm as described is $\mathcal{O}(N^3)$. The CPU time to approximate solutions for different values of N were measured and plotted as a log scale graph. The data points approximately formed a line with slope three, indicating that the algorithm is running in the predicted amount of time.

4.3 Analytical Solutions to Stokes Equation

An analytical solution to the Stokes equation comes from the Hadamard-Rybczynski Problem [16]. This problem starts considering a spherical drop with radius a and viscosity μ_i, translating in the z-direction with speed U in a fluid with viscosity μ_0. The kinematic conditions of the equation require the velocity normal to the boundary to be continuous. Furthermore, the no-slip condition requires that the tangential component of the velocity

and traction match and be continuous across the boundary as well.

Requiring the above and forcing the drop to be spherical, the only degree of freedom that remains is the normal component of the traction across the boundary. The additional requirement that the drop not deform forces the normal of the component of traction to be matched across the boundary as well.

With no remaining degrees of freedom, Lamb's general solution can be used to find the required external velocities of the drop here written in polar coordinates

$$v_r = \left(A_1 \left(\frac{a}{r} \right) - 2B_1 \left(\frac{a}{r} \right)^3 \right) U \cos\theta \tag{114}$$

$$v_\theta = -\left(\frac{A_1}{2} \left(\frac{a}{r} \right) + B_1 \left(\frac{a}{r} \right)^3 \right) U \sin\theta \tag{115}$$

4.4 Validation of Stokes Solver

Coming by the weekend

5 Project Schedule

- Phase I (present - early November):
  - Organize Poisson and Stokes solvers
  - Analyze incompatibilities in data structures and libraries
  - Devise solution to incompatibilities, recoding as necessary
  - Merge solvers into one software
- Phase II (November - December):
  - Validate and test steady state solver
  - Optimize and fix bugs as necessary
  - Run simulations and archive results
- Phase III (December - early February):
  - Incorporate DRM code
  - Incorporate FMM (optional)
- Phase IV (February-March):
  - Validated and test DRM solver
  - Optimize and fix bugs as necessary
  - Run simulations and archive results

– Prepare final report and presentation

Phase I required a significant of code analysis and planning. The solvers for the Stokes and Poisson equations share the same standard structure and subroutines. However, because they were written at different times, there were several major incompatibilities between the two solvers. In general the Stokes solver was found to be coded in a much more robust, efficient, and modular manner. Thus, the majority of the code for the Poisson solver was rewritten while the code for solving Stokes equations suffered on cosmetic changes.

The most obvious difference is the use of data structures. Fortran 90 is an object orientated language that allows for defined types. These types can be used to combine several different data structures into one meaningful structure. The Stokes solver implemented some data structures. However, for all practical purposes, the Poisson solver was devoid of any such structures, handling all variables in isolation.

A second problem was the lack of cohesiveness between the subroutines between the two solvers. Both solvers are based on BEM, use cubic spline interpolation, Gaussian integration, radial integration methods, and compute geometric data such as normals and tangents to the boundary. Each program had an independent version of these and many other routines that differed slightly between one another in no real meaningful way.

Perhaps the largest difference between the Poisson solver and Stokes solver was the Stokes solver's ability so solve multiparticle problems while the Poisson solver was limited to only one particle. Both for compatibility and scientific need, the Poisson solver had to be updated to work on multiparticle problems.

To address the data structure issue, a new data type was created to store data that defined the problem. The structure was designed around this type of data that was absolutely essential to the operation of the solvers. Although the data structure is large, it was designed to be as minimum as possible. It was also designed in such a way that would require minimum changes to Stokes solver. The problem type data structure contains an allocatable number of particle types. Each particle type contains and allocatable number of (x, y) coordinates of its boundary nodes, heat potential u, heat flux q, and velocity v. Since the entire data structure is dynamic, the data structure and software can be used simulate a variable number of particles each with a variable resolution.

Merging subroutines was performed in one of two ways. If there were minimal differences between the subroutines, they were unified and the necessary interface changes were made. Interface changes were kept to a minimum to avoid large changes in the higher level routines. For those subroutines that performed the same task, but operated significantly differently, such as the radial integration method for Poisson and Stokes, the subroutines were kept separate but overloaded. Care was given to make sure overloaded functions still behaved similarly internally. This was an issue with the radial integration method where, although both the Poisson and Stokes solvers used the method, each method used a different quadrature and number of nodes.

Finally, the high level Poisson code was rewritten to handle multiparticle problems. Adding the multiparticle capability required additional loops. Also, the original nesting of the loops was reversed. Switching the looping order allowed for on-the-fly computing

where specific geometry could be computed exactly once and discarded immediately after use. The result is decreased memory usage with no computation penalty.

Merging of the codes was done by slowly adding components to a new code base, dead code could easily be detected and removed. This also forced the code to be modular. When a section code to be added was found to be a copy of a subroutine, the section was discarded and replaced with the appropriate function call.

6 Deliverables

This project will yield a highly modular and optimized source code and software for solving transient thermal Stokes flow problems with source terms in $\mathbb{R}^2$ using DRM. There is also the possibility of having an incorporated FMM solver in the final code base. Great care will be made to ensure the code is well organized and easily modifiable for future projects.

A report and presentation that cover the algorithm will be presented to the class. Furthermore, simulations of physical phenomena will be recorded and written in a paper to be submitted for publication. Those results will be presented in Berlin, Germany at 12^{th} International Workshop on Modeling of Mantle Convection and Lithospheric Dynamics in Summer 2011.

References

[1] R. M. Canup and E. Asphaug, "Origin of the Moon in a giant impact near the end of the Earths formation", *Nature*, Volume 412, 2001, Pages 708-712

[2] D. Martin and R. Nokes. "A Fluid-Dynamical Study of Crystal Settling in Convecting Magmas", *Journal of Petrology*, Volume 30, Issue 6, 1989, Pages 1471-1500

[3] V. Solomatov and D. Stevenson, "Suspension in Convective Layers and Style of Differentiation of a Terrestrial Magma Ocean", *Journal of Geophysical Research*, Volume 98, Issue E3, 1993, Pages 5375-5390

[4] V. Solomatov and D. Stevenson, "Nonfractional Crystallization of a Terrestrial Magma Ocean", *Journal of Geophysical Research*, Volume 98, Issue E3, 1993, Pages 5391-5406

[5] V. Solomatov and D. Stevenson, "Kinetics of Crystal Growth in Terrestrial Magma Ocean", *Journal of Geophysical Research*, Volume 98, Issue E3, 1993, Pages 5407-5418

[6] V. Solomatov, "Magma Oceans and Primordial Mantle Differentiation", *Treatise on Geophysics*, Volume 9, 2007, Pages 91-120

[7] L. T. Elkins-Tanton, E. M. Parmentier, and P. C. Hess, "Magma ocean fractional crystallization and cumulate overturn in terrestrial planets: Implications for Mars", *Meteoritics & Planetary Science*, Volume 38, Issue 12, 2003, Pages 1753-1771

[8] D. Nardini and C. A. Brebbia, "A new approach to free vibration analysis using boundary elements", *Boundary Element Methods in Engineering*, Volume 7, Issue 3, 1983, Pages157-162

[9] D. Nardini and C. A. Brebbia, "Transient dynamic analysis by the boundary element method", *Boundary Elements*, 1983, Pages 719-730

[10] D. Nardini and C. A. Brebbia, "Boundary integral formulation of mass matrices for dunamic analysis", *Topics in Boundary Element Research*, Volume 2: Time-Dependent and Vibration Problems, 1985, Pages 191-208

[11] L. Gaul, M. Kögl, M. Wagner, *Boundary Element Methods for Engineers and Scientists*, Springer, Berlin, 2003

[12] C. Pozrikidis, *Boundary integral and singularity methods for linearized viscous flow*, Cambridge University Text, New York, NY, 1992

[13] Xiao-Wei Gao, "Numerical evaluation of two-dimensional singular boundary integrals-Theory and Fortran code", *Journal of Computational and Applied Mathematics*, Volume 188, Issue 1, 2006, Pages 44-64

[14] Y. J. Liu and N. Nishimura, "The fast multipole boundary element method for potential problems: A tutorial", *Engineering Analysis with Boundary Elements*, Volume 30, Issue 5, May 2006, Pages 371-381

[15] G. Leal, *Laminar Flow and Convective Transport Processes*, Butterworth-Heinemann, 1992

[16] S. Kim and S. J. Karrila, *Microhydrodynamics: Principles and Selected Applications*, Dover, 2005