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. Saswata Hier-Majumder
Assistant Professor, Department of Geology
University of Maryland, College Park, MD
saswata@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 crystal settling behavior. 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 combines algorithms from several papers to produce a robust, modular, and highly optimized software package.

Contents

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 provides some clues on the rate of cooling, computational models of such phenomena 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 settled, likely enhancing the efficiency of cooling. Direct estimation of heat flux and crystal size, however, is 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 equation coupled with the heat equation through an advection term. As closed form solutions rarely exist for problems of interest, numerical solutions are an attractive option.

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 generally converge with reasonable amounts of computation. Unfortunately, there are two issues with FEM that make it impractical for the above problem. First, as the crystals settle, the region of flow changes. The problem's dynamic geometry means the region would have to be rediscretized at every time step. Rediscretization becomes extremely computationally costly, especially in high dimensions.

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 PDE, employing reciprocal relations, integrating by parts, and making specific choices on test functions such that the integrals move entirely to the boundary. This method has the immediate advantage of reducing the dimensionality of the problem by one. Also, BEM naturally handles the problems caused by dynamic geometry. Unlike the domain meshes needed for FEM, BEM only discretizes the boundary. For any evaluation, only three easily computable pieces of information about the boundary are needed. First is the node's position. 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. The normal can be quickly approximated using the local interpolation of nodes.

A critical limitation to BEM is the requirement of the fundamental solution to the adjoint of the differential operator. Without this solution, the integral equations cannot be transformed completely to the boundary. Unfortunately, fundamental solutions are only available for a small number of differential operators. One operator for which a fundamental solution is not known is the adjoint of the heat equation with an advection term. To approximate a solution to the heat equation using boundary integrals, the Dual Reciprocity Method (DRM) will be used.

The paper is organized as follows. Section 2 will cover the analytical procedures for developing the boundary integral equations, the key for both BEM and DRM. Section 3 outlines the discretization of the analytical formulas and implementation of the DRM software. Section 4 contains the validation and error analysis of the DRM based solvers for the Poisson and heat equations. Future research is outlined in section 5, and section 6 contains closing remarks.

2 Method

In order to solve a PDE using BEM or DRM, we must first restate the strong PDE as a boundary integral equation. The boundary integral equation is a weak formulation of the original problem with the additional restriction that the only integrals present are over the boundary of the domain. In this way, discretization of the geometry and computation of the solution need only take place on the boundary which is in a lower dimensional space than the original problem.

Section 2.1 describes the conversion of the Stokes flow PDE to the corresponding boundary integral equations detailing analytical choices made based on the physical system being modeled. Since the Stokes flow solver is not the main focus of the project, many of the analytical details are omitted. An interested reader may refer to [8].

Section 2.2 details the derivation of the Dual Reciprocity boundary integral equation. There are several choices that must be made, independent of the physical system being modeled, to fully implement DRM. The choices are outlined in the derivation. However, the final boundary integral equation is derived without considering any choices the method requires.

2.1 Stokes Boundary Integral Equation

The Stokes flow equation for viscous flows is given by

$$-\nabla p + \mu \Delta \vec{v} = -\vec{g} \qquad \vec{x} \in \mathbb{R}^2 \tag{1}$$

where p is the fluid pressure, μ the viscosity, $\vec{v}$ the velocity, and $\vec{g}$ the body force. In the case of N viscous crystals falling into a magma ocean, the viscosity term can vary between the suspending fluid and individual particles. For this specific case, we have a system of partial differential equations:

$$\begin{aligned} -\nabla p + \mu_m \Delta \vec{v} &= -\vec{g} \qquad & \vec{x} \in \Omega_m \qquad 1 \le m \le N \\ -\nabla p + \mu_0 \Delta \vec{v} &= -\vec{g} \qquad & \vec{x} \in \Omega_0 = \mathbb{R}^2 \backslash \left(\cup_{m=1}^{N} \Omega_m \right) \end{aligned} \tag{2}$$

where Ω_m, $1 \le m \le N$, is the domain for the m^{th} crystal. We assume the viscosity, μ_m, is constant throughout the crystal. So crystals do not overlap, we require $\Omega_m \cap \Omega_l = \emptyset$ for $1 \le m, l \le N$ and $m \ne l$. Physically, it is logical for the domain of the crystals to be bounded. The domain of the suspension fluid, Ω_0, is unbounded and has constant viscosity μ_0.

To solve (2), we need information about the interaction between particles and the suspension fluid along the boundaries. We apply two natural boundary conditions on Γ_m, $1 \le m \le N$. The first condition is the non-slip boundary which states that the tangential velocity is continuous across the interface. That is:

$$\vec{v}^{\Omega_0} \cdot \vec{t} = \vec{v}^{\Omega_m} \cdot \vec{t} \qquad \vec{x} \in \Gamma_m \tag{3}$$

where the superscript Ω_0 indicates the variable as viewed from the suspension fluid and the superscript Ω_m indicates the variable as viewed from the m^{th} crystal.

The second required piece of information is the normal component of the stress jump across the boundary. The jump is defined as

$$\Delta f = \left(\sigma^{\Omega_0} - \sigma^{\Omega_m} \right) \cdot \vec{\nu} \qquad \vec{x} \in \Gamma_m \tag{4}$$

where $\vec{\nu}$ is the normal vector along Γ_m pointing into Ω_0 and σ is the stress tensor defined as

$$\sigma = -PI + \mu \left(\nabla \otimes \vec{v} + (\nabla \otimes \vec{v})^T \right). \tag{5}$$

We handle the stress jump term by introducing an isotropic surface term γ. Then it can be shown that

$$\Delta f = \gamma \left(\nabla \cdot \vec{\nu} \right) \vec{\nu} - \left(I - \vec{\nu} \otimes \vec{\nu} \right) \cdot \nabla \gamma. \tag{6}$$

For simplicity, we choose γ to be constant for each particle. Therefore, the jump in stress over the interface is completely dependent by the curvature of the crystal interface.

Using boundary conditions (3) and (6), the boundary integral equation for (2) is given in [9] as

$$c(\vec{x}_i)v(\vec{x}_i) = \frac{1}{1+\lambda_q}\left[\vec{v}^{\infty} - \frac{1}{4\pi\mu_0}\sum_{m=1}^{N}\int_{\Gamma_m}[\Delta f]^T\cdot\mathcal{G}_i + \frac{1}{4\pi}\sum_{m=1}^{N}(1-\lambda_m)\int_{\Gamma_m}\vec{v}^T\cdot\mathcal{T}_i\cdot\vec{\nu}\right] \quad (7)$$

where $\vec{x}_i \in \Omega_q$, $\vec{v}^{\infty}$ is the imposed velocity at $|\vec{x}| \to \infty$, $\lambda_m = \frac{\mu_m}{\mu_0}$ is the ratio between the viscosity of the m^{th} particle and the suspension fluid, $\mathcal{G}_i$ is the Green's function for the velocity, and $\mathcal{T}_i$ is the Green's function for the stress tensor with

$$\mathcal{G}_i = -\ln(r_i)I + \frac{\hat{x}_i \otimes \hat{x}_i}{r_i^2} \quad (8)$$

$$\mathcal{T}_i = -4\frac{\hat{x}_i \otimes \hat{x}_i \otimes \hat{x}_i}{r_i^4} \quad (9)$$

where $\hat{x}_i = \vec{x} - \vec{x}_i$, $r_i = |\hat{x}_i|$, and $c(\vec{x}_i)$ is an indicator function resulting from a limiting process which, in loose terms describes what fraction of the point $\vec{x}_i$ lies in the domain Ω_q. The indicator function is defined by

$$c(\vec{x}_i) = \begin{cases} 1 & \vec{x}_i \in \Omega_q \\ 1 - \frac{\alpha}{2\pi} & \vec{x}_i \in \Gamma_q \\ 0 & \text{otherwise} \end{cases} \quad (10)$$

where α is the magnitude of the range of angles for all possible rays starting at $\vec{x}_i$ and pointing out of the domain Ω_q. In practice, $\alpha = \pi$ which corresponds to the boundary being locally continuous around $\vec{x}_i$.

2.2 Dual Reciprocity Boundary Integral Equation

DRM [10][11][12] is an extension of BEM. With BEM, the fundamental solution to the adjoint operator must be known. In addition, there are usually requirements on the source term in the PDE. For instance, in the case of the Poisson equation, the source term must be harmonic for the integral equation to be written completely in terms of boundary integrals. These restrictions limit the applicability of BEM. DRM expands the capabilities of BEM. For this project, we use DRM on a PDE of the form

$$-\Delta u = \hat{b} \qquad \vec{x} \in \Omega \subseteq \mathbb{R}^2 \quad (11)$$

with Neumann boundary conditions. We lift all restrictions on the residual term $\hat{b}$, even allowing it to be a function of the unknown u. In this way we are able to extend (11) to describe more general PDEs such as the heat equation with an advection term.

Like BEM, DRM requires knowledge of the fundamental solution to the adjoint of the differential operator. In (11), the differential operator is the Laplace operator which is self adjoint. Thus, we need to find the fundamental solution u_i^* where

$$-\Delta u_i^* = \delta(|\vec{x} - \vec{x}_i|) \qquad \vec{x} \in \mathbb{R}^2. \quad (12)$$

Following literature [13], we choose to use the whole space fundamental solution rather than the fundamental solution derived for the specific domain Ω.

It is well known that, in $\mathbb{R}^2$, the fundamental solution to the Laplace operator is given by

$$u_i^* = -\frac{1}{2\pi} \ln r_i \tag{13}$$
$$r_i = |\vec{x} - \vec{x}_i|. \tag{14}$$

For boundary methods, it is useful to define the flux density of a function on the boundary of the domain:

$$q = \vec{\nu} \cdot \nabla u \tag{15}$$

where $\vec{\nu}$ is the outward facing normal on Γ, the boundary of the domain Ω. The flux density of the solution is then given by

$$q_i^* = \frac{\hat{x}_i \cdot \vec{\nu}}{2\pi r_i^2} \tag{16}$$
$$\hat{x}_i = \vec{x} - \vec{x}_i. \tag{17}$$

In addition to the fundamental solution of the Laplace operator, DRM also requires choosing a set of basis functions, $\hat{f}_i$, to approximate the residual term:

$$\hat{b} \approx \sum_i \beta_i \hat{f}_i \tag{18}$$

where β_i are scalar coefficients. The only restriction on the basis functions is that they satisfy

$$-\Delta \hat{u}_i = \hat{f}_i \qquad x \in \mathbb{R}^2. \tag{19}$$

While not required, it is practical to choose $\hat{f}_i$ such that $\hat{u}_i$ are known analytically in a closed form. Our specific choice of basis functions is reserved for section 3.4.

To formulate the Dual Reciprocity boundary integral equation, we start by multiplying both sides of (11) by the fundamental solution (13) and integrating over the domain Ω:

$$-\int_\Omega \Delta u u_i^* = \int_\Omega \hat{b} u_i^*. \tag{20}$$

Starting with the left hand side of (20), we apply integration by parts twice:

$$\begin{aligned} -\int_\Omega \Delta u u_i^* &= -\int_\Gamma q u_i^* + \int_\Omega \nabla u \cdot \nabla u_i^* \\ &= -\int_\Gamma q u_i^* + \int_\Gamma u q_i^* - \int_\Omega u \Delta u_i^*. \end{aligned} \tag{21}$$

Recalling (12), we see that the left hand side becomes

$$-\int_\Omega \Delta u u_i^* = c(\vec{x}_i)u(\vec{x}_i) + \int_\Gamma u q_i^* - q u_i^* \tag{22}$$

where $c(x_i)$ is the indicator function as in (10).

For the right hand side, we first make use of (18) and (19) and approximate the domain integral:

$$\begin{aligned}\int_\Omega \hat{b} u_i^* &\approx \sum_j \beta_j \int_\Omega \hat{f}_j u_i^* \\ &= \sum_j \beta_j \int_\Omega -\Delta \hat{u}_j u_i^*. \end{aligned} \tag{23}$$

As before, we apply integration by parts twice to the right hand side making use of the fundamental solution to obtain

$$\int_\Omega \hat{b} u_i^* \approx \sum_j \beta_j \left(c(\vec{x}_i)\hat{u}_j(\vec{x}_i) + \int_\Gamma \hat{u}_j q_i^* - \hat{q}_j u_i^* \right). \tag{24}$$

Substituting (22) and (24) into (20), we obtain the Dual Reciprocity boundary integral equation:

$$c(\vec{x}_i)u(\vec{x}_i) + \int_\Gamma u q_i^* - q u_i^* = \sum_j \beta_j \left(c(\vec{x}_i)\hat{u}_j(\vec{x}_i) + \int_\Gamma \hat{u}_j q_i^* - \hat{q}_j u_i^* \right). \tag{25}$$

3 Implementation

The implementation details for DRM are broken up into five sections. The objective of each is to help describe the process of transforming the Dual Reciprocity boundary integral equation into a tractable computational problem. We start in section 3.1 by discretizing both the geometry and the boundary integrals to form the general Dual Reciprocity matrix equation for an unspecified residual term $\hat{b}$. Some of the integrals required for the discretization contain singularities. The singularities cause computational difficulties which require a specialized method to overcome. Section 3.2 introduces the radial integration method for computing integrals containing singularities. Section 3.3 will examine the Dual Reciprocity matrix equation for the specific residual term used in the heat equation with an advection term. The section ends with a linear system for stepping the solution forward in time. The last analytical detail, the choice of basis functions for approximating the residual, is presented in section 3.4. Finally, section 3.5 provides a detailed description of the software implementation used for the solver. Details are included on data structures, optimization, parallelization, and use of external libraries.

3.1 Discretization

Discretization of the Dual Reciprocity boundary integral equation begins with the discretization of the problem geometry. Since DRM is a boundary method, only the boundary Γ need be discretized. In the case of our problem in $\mathbb{R}^2$, thus we only need to discretizing a curve.

The discretization step allows for freedom in the choices on how to divide the boundary into a finite number of boundary elements and where to place the computational nodes on which the approximate solution will be calculated. We choose to follow the method of collocation which states that N computational nodes be placed on the boundary. The boundary elements are defined by the boundary segments that contain exactly two computational nodes located exactly at the segment's end points. DRM also requires an additional L computational nodes to be placed throughout the domain. The exact placement of the computational nodes is differed until section 4.

The locations of the N computation nodes are used to approximate the boundary elements. We use cubic spline interpolation to interpolate the boundary between the nodes. The cubic spline coefficients can be calculated in $\mathcal{O}(N)$ and use $\mathcal{O}(N)$ storage. In addition, the cubic spline interpolant is twice continuously differentiable everywhere along the boundary. Thus the normal vector exists along the entire boundary. Furthermore, cubic spline interpolation allows for a direct method of exactly calculating the normal vector at any point.

The potentials and flux densities associated with the unknown solution and the basis functions will be saved only at the $N + L$ computational nodes for the potentials and N computational nodes on the boundary for the flux densities. In order to evaluate the boundary integrals required for DRM, all potentials and flux densities must be known along the entire boundary. We will approximate the potentials and flux densities over each boundary element using a linear interpolation with respect to the local cubic spline interpolation variable.

Imposing the method of collocation, cubic spline interpolation, and linear interpolation of the potentials and flux densities, the Dual Reciprocity boundary integral equation (25) can be written in matrix form as

$$Cu + [H|0]u + Gq = (C\hat{U} + [H|0]\hat{U} + G\hat{Q})\beta \tag{26}$$

where

$$\begin{aligned}
C_{ij} &= \delta_{ij} c(\vec{x}_i) && (27)\\
H_{ij} &= \int_{-1}^{1} \frac{1-\xi}{2} q_i^* J_j \, d\xi + \int_{-1}^{1} \frac{1+\xi}{2} q_i^* J_{j-1} \, d\xi && (28)\\
G_{ij} &= \int_{-1}^{1} -\frac{1-\xi}{2} u_i^* J_j \, d\xi + \int_{-1}^{1} -\frac{1+\xi}{2} u_i^* J_{j-1} \, d\xi && (29)\\
\hat{U}_{ij} &= \hat{u}_j(\vec{x}_i) && (30)\\
\hat{Q}_{ij} &= \hat{q}_j(\vec{x}_i) && (31)
\end{aligned}$$

with $C, \hat{U} \in \mathbb{R}^{(N+L)\times(N+L)}$; $H, G \in \mathbb{R}^{(N+L)\times N}$; and $\hat{Q} \in \mathbb{R}^{N\times(N+L)}$. Additionally, $u, \beta \in \mathbb{R}^{N+L}$; and $q \in \mathbb{R}^N$ where u and q are the potential and flux density of the solution, respectively. In the integrals (28) and (29), ξ is the local cubic spline interpolation variable and J_j is the Jacobian along the boundary element Γ_j. For simplicity, we order the nodes such that node x_i with $1 \leq i \leq N$ is on the boundary and node x_i with $N+1 \leq i \leq N+L$ is in the domain.

The vector β contains the coefficients to the basis functions that approximate the residual term in (18). We can rewrite the approximation as a linear equation

$$\beta = \hat{F}^{-1}\hat{b} \tag{32}$$

where

$$\hat{F}_{ij} = \hat{f}_j(x_i) \tag{33}$$

and $\hat{b}$ is the residual discretized in space. Notice that $F \in \mathbb{R}^{(N+L)\times(N+L)}$ and $\beta, \hat{b} \in \mathbb{R}^{N+L}$.

Finally, it is convenient to define

$$\tilde{H} = C + [H|0]. \tag{34}$$

Using (34) and (32), the Dual Reciprocity matrix equation (26) becomes

$$\tilde{H}u + Gq = (\tilde{H}\hat{U} + G\hat{Q})\hat{F}^{-1}\hat{b}. \tag{35}$$

It is left to compute the integrals in (28) and (29), the residual $\hat{b}$, and pick basis functions in order to assemble $\hat{F}$, $\hat{U}$, and $\hat{Q}$.

3.2 Radial Integration Method

The assembly phase of the DRM solver is dominated by computation of integrals in (28) and (29). Each integral takes place on a boundary element Γ_j and integrates either the fundamental solution (13) or the flux density of the fundamental solution (16) centered at the computational node $\vec{x}_i$. If $\vec{x}_i \notin \Gamma_j$ the integration can be carried out using standard Gaussian quadrature. However, if $\vec{x}_i \in \Gamma_j$, the integrand becomes singular.

There are two types of singular integrals encountered in DRM. In (29), the integrals contain a $\log(r_i)$ term where

$$r_i = |\vec{x} - \vec{x}_i|. \tag{36}$$

An integral of this type is weakly singular. That is, the integrand is unbounded, however any integral with finite bounds is bounded. The integrals in (28) contain a $\frac{1}{r_i}$ term which is strongly singular. With strongly singular integrals, a definite integral containing the singularity in its interior is only finite in the Cauchy principal value sense.

In each case, we use the radial integration method [14]. This method considers integrals of the form:

$$\int_{\Gamma_e} \bar{f} \ln(r_i) \tag{37}$$

$$\int_{\Gamma_e} \frac{\bar{f}}{r^\beta} \tag{38}$$

where $\bar{f}$ is a bounded function and Γ_e is a boundary segment containing $\vec{x}_i$ in the interior. The integral is separated into two segments, Γ_e^- and Γ_e^+, where $\Gamma_e^- \cup \Gamma_e^+ = \Gamma_e$ and $\Gamma_e^- \cap \Gamma_e^+ = \vec{x}_i$. The finite part of each integral is then calculated. Note that for weakly singular integrals, the values obtained through radial integration on $\Gamma_e^{+/-}$ is the finite value of the integral on the boundary segment. However, in the case of strongly singular integrals, the finite values only make sense as a Cauchy principal value on the entire boundary segment Γ_e.

To perform the integration on the boundary, the global coordinate $\vec{x} = (x, y)$ is transformed to a local coordinate ξ where the Jacobian is given by

$$|J_\Gamma| = \sqrt{\left(\frac{\partial x}{\partial \xi}\right)^2 + \left(\frac{\partial y}{\partial \xi}\right)^2}. \tag{39}$$

We then transform from the local variable ξ to the global distance variable r where the Jacobian for this transformation is given by

$$|J_r| = \left(\frac{\partial r}{\partial x}\frac{dx}{d\xi} + \frac{\partial r}{\partial y}\frac{dy}{d\xi}\right). \tag{40}$$

The transformation from global coordinate $\vec{x}$ to global distance coordinate r is given by

$$d\Gamma = \frac{dr}{\hat{x}_i \cdot \vec{t}} \tag{41}$$

where

$$\hat{x} = \vec{x} - \vec{x}_i \tag{42}$$

and $\vec{t}$ is the unit tangent vector along the boundary segment.

The key to the radial integration method is to approximate the Jacobian and bounded function, $\bar{f}$, in terms of radial functions:

$$\frac{\bar{f}}{\hat{x} \cdot \vec{t}} \approx \sum_{i=0}^{N} C_i r_i. \tag{43}$$

The approximation (43) can be substituted into (37) and (38). With this approximation, the integrals can be solved analytically and the finite contribution is found to be

$$\int_{\Gamma_e^{+/-}} \frac{\bar{f}}{r^\beta} \approx \sum_{i=0}^{N} C_i F_i \tag{44}$$

for strongly singular integrals and

$$\int_{\Gamma_e^{+/-}} \bar{f} \ln(r_i) \approx \sum_{i=0}^{N} C_i H_i \tag{45}$$

for weakly singular integrals where

$$F_i = \begin{cases} \frac{r^{i-\beta+1}}{i-\beta+1} & i-\beta+1 \neq 0 \\ \ln(r) & i-\beta+1 = 0 \end{cases} \tag{46}$$

and

$$H_i = \frac{(i+1)\ln(r) - 1}{(i+1)^2} r^i. \tag{47}$$

In this way, we can quickly and accurately compute the finite values of weakly singular integrals and Cauchy principal values of strongly singular integrals that arise during assembly of Dual Reciprocity matrices.

3.3 The Heat Equation

As mentioned in the introduction, one of the main goals is to solve the heat equation with an advection term:

$$\frac{\partial}{\partial t} u + \vec{v} \cdot \nabla u - \Delta u = b \qquad \vec{x} \in \Omega \tag{48}$$

where $\vec{v}$ is a known velocity vector and b a known scalar body force. Until this point, we have only addressed how to solve PDEs in the form

$$-\Delta u = \hat{b} \qquad \vec{x} \in \Omega \tag{49}$$

using DRM. It was mentioned in section 2.2 that there are no restrictions on the residual term $\hat{b}$ and that it can be a function of the unknown solution. Therefore, (48) can be manipulated to fit the form of (49) by defining the residual term for the heat equation as

$$\hat{b} = b - \frac{\partial}{\partial t} u + \vec{v} \cdot \nabla u. \tag{50}$$

We plug (50) into (35), the Dual Reciprocity matrix equation, to obtain

$$\tilde{H} u + G q = (\tilde{H}\hat{U} + G\hat{Q})\hat{F}^{-1} \left[b - \frac{\partial}{\partial t} u + \vec{v} \cdot \nabla u \right]. \tag{51}$$

Notice in (51), the residual contains the gradient term ∇u. Since no knowledge of the gradient of the solution is assumed, it must be approximated.

There are several ways to approximate the gradient of a discretized function by applying a linear transformation to the discretized function. An exceptionally clever technique for approximating the gradient in the context of DRM was introduced by Partidgo and Brebbia [15]. It was shown that DRM could produce accurate solutions to the Helmholtz equation:

$$\Delta u = u \qquad \vec{x} \in \Omega \tag{52}$$

with

$$\beta = \hat{F}^{-1} u. \tag{53}$$

Furthermore, the spacial derivatives of u could be written as

$$\frac{\partial}{\partial x} u \approx \frac{\partial \hat{F}}{\partial x} = \frac{\partial \hat{F}}{\partial x} \hat{F}^{-1} u \tag{54}$$

$$\frac{\partial}{\partial y} u \approx \frac{\partial \hat{F}}{\partial y} = \frac{\partial \hat{F}}{\partial y} \hat{F}^{-1} u \tag{55}$$

and used successfully in the residual term $\hat{b}$. We use (54) and (55) to rewrite (51) as

$$\tilde{H} u + G q = (\tilde{H} \hat{U} + G \hat{Q}) \hat{F}^{-1} \left[b - \frac{\partial}{\partial t} u - \left(V_x \frac{\partial \hat{F}}{\partial x} + V_y \frac{\partial \hat{F}}{\partial y} \right) \hat{F}^{-1} u \right] \tag{56}$$

with

$$(V_x)_{ij} = \delta_{ij} v_x(\vec{x}_i) \tag{57}$$

$$(V_y)_{ij} = \delta_{ij} v_y(\vec{x}_i) \tag{58}$$

where $\vec{v} = (v_x, v_y)$. Observe that the Dual Reciprocity matrix equation for (48) is now completely discretized in space.

Notice that for the Neumann problem, the only remaining unknowns in (56) are u and $\frac{\partial}{\partial t} u$. In fact, upon examination, (56) is a first order linear ordinary differential equation with unknown u in the variable t. We use the Crank-Nicolson method to set up a time marching scheme. The exact algebraic manipulation required to do this is straightforward yet can become unwieldy and thus is omitted. The resulting linear system for time marching is

$$\left(\frac{2}{\Delta t} M + N \right) u^{T+1} = \left(\frac{2}{\Delta t} M - N \right) u^T - G q^{T+1/2} + M b^{T+1/2} \tag{59}$$

with

$$M = (\tilde{H} \hat{U} + G \hat{Q}) \hat{F}^{-1} \tag{60}$$

$$N = \tilde{H} + M \left(V_x \frac{\partial \hat{F}}{\partial x} + V_y \frac{\partial \hat{F}}{\partial y} \right) \hat{F}^{-1} \tag{61}$$

and

$$q^{T+1/2} = q^{T+1} + q^T \qquad b^{T+1/2} = b^{T+1} + b^T \tag{62}$$

where the superscript T indicates the value at the current time step, the superscript $T+1$ indicates the values at the next time step, and Δt is the size of the time step. In this way, we can march forward in time until the desired length of simulation is reached.

3.4 Basis Functions

There are many choices for basis function to use for approximating the residual term in DRM. Many of the basis functions are radial, that is they are of the form

$$\hat{f}_i = \hat{f}(r_i) \tag{63}$$

where

$$r_i = |\vec{x} - \vec{x}_i|. \tag{64}$$

Generally, it is preferable to choose basis functions that behave similarly to the residual term. In the case of radial basis functions, it is assumed the residual has some radial behavior. If this is not the case, radial basis functions can still be used, however more computational nodes should be added to assure the approximation is accurate.

Specifically, we choose to use the radial basis function

$$\hat{f}_i = 1 + r_i. \tag{65}$$

Some papers use higher order radial approximations. However, they have not been shown to be effective. With the choice of (65) we must find $\hat{u}_i$ such that

$$-\Delta \hat{u}_i = \hat{f}_i \qquad \vec{x} \in \mathbb{R}^2. \tag{66}$$

We see that

$$\hat{u}_i = -\frac{r_i^2}{4} - \frac{r_i^3}{9} \tag{67}$$

satisfies (66). Furthermore, the flux density of $\hat{u}_i$ can be found:

$$\hat{q}_i = (\hat{x}_i \cdot \vec{\nu}) \left(\frac{1}{2} + \frac{r_i}{3} \right) \tag{68}$$

where

$$\hat{x}_i = \vec{x} - \vec{x}_i \tag{69}$$

and $\vec{\nu}$ is again the outward facing normal vector along the boundary.

3.5 Software Implementation

The solver was implemented in Fortran 95. This programming language was chosen for several reasons. First, we had available two codes at the start of the project. One solved the Stokes flow equation and the other solves the Poisson equation with a harmonic source term. Both codes were written in Fortran 95. Using the same language as the existing code had the immediate benefit of being able to reuse existing code. Second, there are a great number

of highly optimized computational libraries that interface directly with Fortran. Specifically, we made use of OpenMP, BLAS, and LAPack which are all native to Fortran. The specific choice of Fortran 95 over other versions dealt with support and features. The Fortran 95 standard is well established enough to have optimized implementations in several compilers. There is also wide range of documentation, both online and through literature, that is not available with newer versions. Fortran 95 also includes several features not present in its predecessors such as custom data types and method overloading. These two features allowed for more object-orientated software design.

Data structures in particular were central to the design of the solver. Positions of computational nodes, heat potentials, heat flux densities, and velocities are constantly accessed throughout the solver. Furthermore, in the case of the multiparticle problem there may be physical parameters that may be unique for each particle. In addition to the raw size of the different data that must be stored, the data must also be organized in such a manner that it can be easily accessed in a way that makes sense with respect to the problem's geometry. The result was to create one problem data type that could robustly handle any geometry from a simple disk to a collection of tens of thousands of particles.

The main data structure contains all of the global problem parameters, as well as an allocatable array of a particle data type. Some global parameters include the viscosity of the suspension fluid and its heat conductivity. The main data structure also stores information about the discretized geometry such as how many particles there are and how many boundary and interior computational nodes there are for each.

The particle data structure contains information about each individual particle in the simulation. In addition to physical parameters such as the specific particle's viscosity or heat conductivity and book keeping information like the number of boundary and interior computational nodes in the particle, the data structure contains the actual information about the problem geometry and solutions. Each particle has an allocatable array for the position of boundary computational nodes, interior computational nodes, heat potentials, heat flux densities, and velocities on the particle's nodes. By distributing the problem geometry throughout the data structure, the data is naturally organized in a way that makes geometric sense.

The full data structure (figure 1) is fully allocatable which allows it to efficiently handle problems with a varied number of particles, each containing a various number of computational nodes. Most importantly, from a software design perspective, all of the problem data can now be passed by reference with exactly one variable.

The original Stokes and Poisson solvers used a variety of custom hand-built solvers and some provided by Numerical Recipes [16] for implementing basic linear algebra methods. These implementations were replaced by calls to the standard BLAS and LAPack computational libraries. Specifically, the Intel Math Kernel Library implementations of the libraries were used. The clear advantage of using a production caliber library is the speed, robustness, and stability that custom solvers and educational codes, such as Numerical Recipes, are just not able to match.

In addition to the linear algebra routines used to solve the Dual Reciprocity matrix

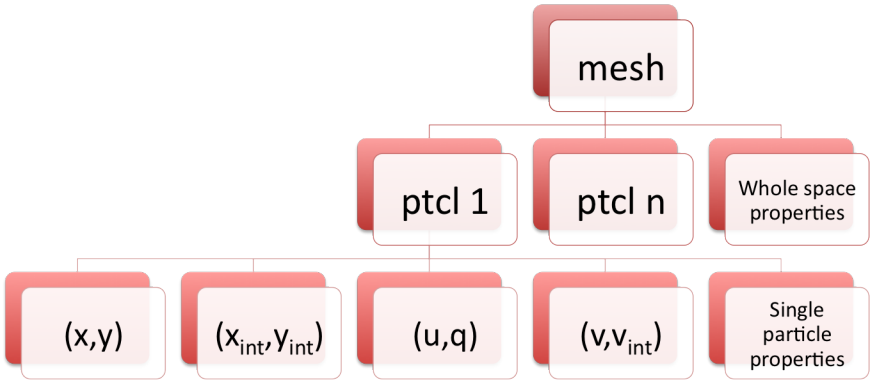

Figure 1: The full data structure. The main data type called mesh contains all of the global variables associated with the global properities of the problem. Mesh also has an allocatable number of particle data structure types, each containing geometry and solution information for a particular particle.

equation, the other computationally-intense part of the solve is the assembly of the matrices. In the current implementation, of the matrices in (56) and (59), only $\tilde{H}$, G, $\hat{F}$, $\hat{U}$ and $\hat{Q}$ are assembled entry-wise. Linear algebra methods are used to construct M and N. Of the five matrices to be assembled, $\tilde{H}$ and G are the most time-consuming as their entries are the result of integration while the others only contain explicit function evaluations.

To optimize the assembly of $\tilde{H}$ and G, the integrals over a particle are performed together. Furthermore, the integrals over a single boundary element are calculated sequentially. This ordering allows the cubic spline coefficients and boundary element interpolation to be calculated exactly once and then discarded before the next set of data is computed. Thus, we evaluate the integrals in an order that requires the minimum amount of geometry computation and storage.

Eventually, the code will run simulations containing tens of thousands of particles. Tackling this problem with a single processor is not feasible. To prepare for larger problems, the current solver uses the OpenMP parallelization library. The assembly routine was parallelized by having each thread handle all of the integrations for a specific boundary element. Parallelization of the linear algebra routines was trivial as Intel's BLAS and LAPack libraries are natively parallelized using OpenMP with a compile time flag.

4 Validation

Validation of the code was split into two phases corresponding to the development of the DRM solver. In the first phase, the solver was built to approximate solutions to the Poisson

problem. This solver made use of the general Dual Reciprocity matrices $\tilde{H}$, G, $\hat{F}$, $\hat{U}$ and $\hat{Q}$ in (35). These are the same matrices that are later used in the Dual Reciprocity heat solver. For this reason, we validate the Poisson solver to confirm the DRM matrices are being properly assembled.

After the Poisson solver was validated, the residual term for the Poisson problem was replaced by the residual term for the heat equation, and Crank-Nicolson was applied to gain a time marching scheme. Here, the second stage of validation took place. This validation ensured the matrices M and N from (59) were assembling correctly and the time marching scheme was implemented correctly.

4.1 Poisson Solver Validation

Error analysis was performed on the Poisson problem

$$\Delta u = b \qquad \vec{x} \in \Omega \tag{70}$$

$$u = g \qquad \vec{x} \in \Gamma \tag{71}$$

where Ω was chosen to be the disk of radius one centered at the origin. For validation of the algorithm, we choose a set of particular solutions to the Poisson problem. It can be shown that

$$u = \frac{r^2}{8}(a_1 x + a_2 y) \qquad (x, y) \in \Omega \cup \Gamma \tag{72}$$

$$q = -\frac{1}{8}(a_1 x + a_2 y) \qquad (x, y) \in \Gamma \tag{73}$$

$$b = a_1 x + a_2 y \qquad (x, y) \in \Omega \cup \Gamma \tag{74}$$

solves (70) and (71) where a_1 and a_2 are arbitrary scalar constants, and, as before, $q = \nabla u \cdot \vec{\nu}$ where $\vec{\nu}$ is the outward facing unit normal vector along Γ. Recall that since we are solving the problem on the boundary only, DRM will produce an approximation of q on the boundary.

First we validate the solver against various particular solutions by choosing an assorted set of values for a_1 and a_2. Table 4.1 shows the $L_2(\Gamma)$ relative error of the solution on the boundary for the choice of parameters. Notice that the solution on the disk centered at the origin is symmetric with respect to a_1 and a_2. However, the error is not symmetric due to the lack of symmetry in the discretization of the geometry. The lack of symmetry results in small differences in the solution which produce the sightly different errors.

DRM makes use of interior nodes to approximate the residual term. In the case of the Poisson problem, the residual is a known function. The distribution of the interior nodes has the ability to contribute or detract greatly to the accuracy of the approximation of the residual term and thus the solution. For the first step in error analysis, we test the convergence of the approximation for two distributions of interior nodes.

First, we examine the case where the interior nodes are randomly distributed. We place N nodes evenly spaced on the boundary and randomly distribute $L \approx N/2$ interior nodes using a uniform distribution (figure 2). Second, we distribute the nodes in a structured

(a_1, a_2)	0.00	0.25	0.50	0.75	1.00
0.00		4.087E-03	4.087E-03	4.087E-03	4.087E-03
0.25	4.045E-03	3.313E-03	3.729E-03	3.903E-03	3.975E-03
0.50	4.045E-03	3.797E-03	3.313E-03	3.517E-03	3.729E-03
0.75	4.045E-03	3.961E-03	3.590E-03	3.313E-03	3.417E-03
1.00	4.045E-03	4.023E-03	3.797E-03	3.482E-03	3.313E-03

Table 1: The relative L_∞ for exact solutions with different parameters (a_1, a_2) with $N = 50$, $L = 32$, and interior latice nodes.

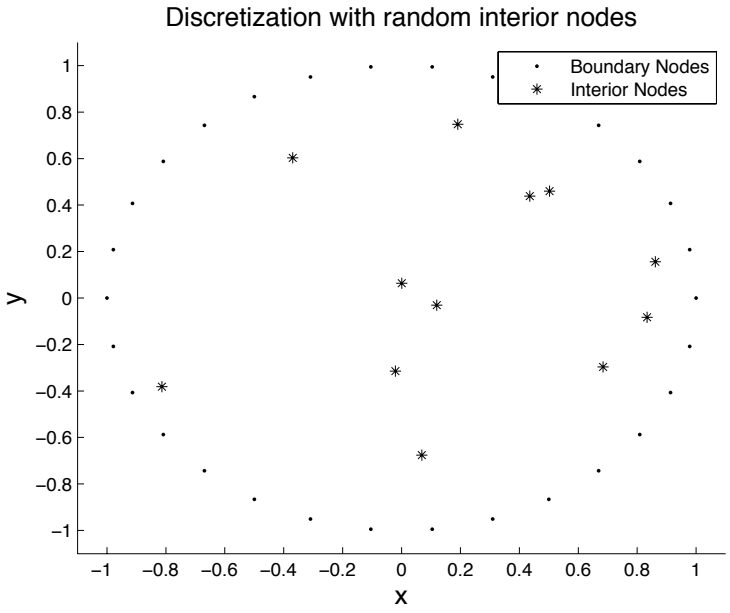

Figure 2: Interior nodes distributed randomly.

manner. The interior nodes are laid out in a rectangular lattice inside the disk (figure 3). Once again, there are N evenly distributed nodes on the boundary with $L \approx N/2$ interior nodes. Notice in each case, the number of interior nodes is approximately half of the number of nodes on the boundary. Due to the methods used to distribute the interior nodes, an equality relationship between L and N cannot be defined. Figure 4 shows the result from the randomly distribution of interior nodes, and figure 5 shows the error results from the lattice distribution.

For each distribution, the error decreases as the number of nodes is increased. Notice at N increases, the accuracy of the random distribution begins to match that of the lattice distribution, suggesting that for a large enough number of nodes, distribution may not contribute much to the over all accuracy of the solution. Also, notice that as the number of nodes increases, the accuracy gains diminish.

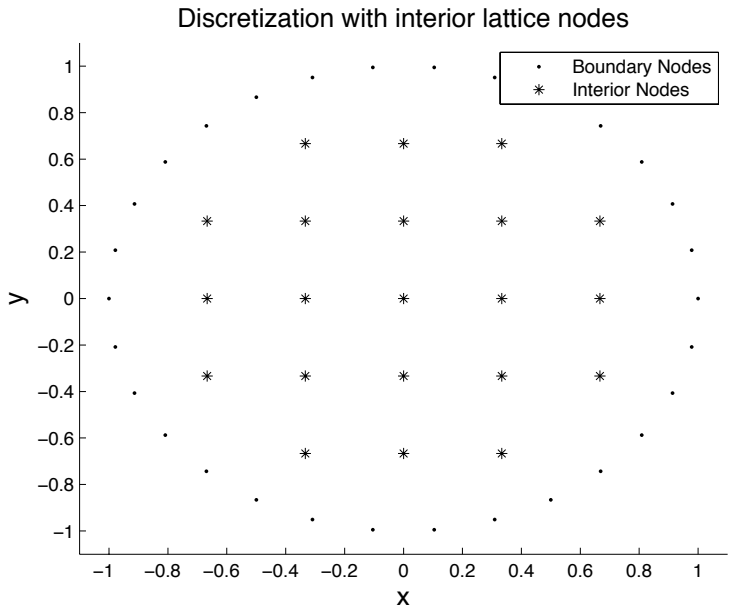

Figure 3: Interior nodes distributed on a lattice.

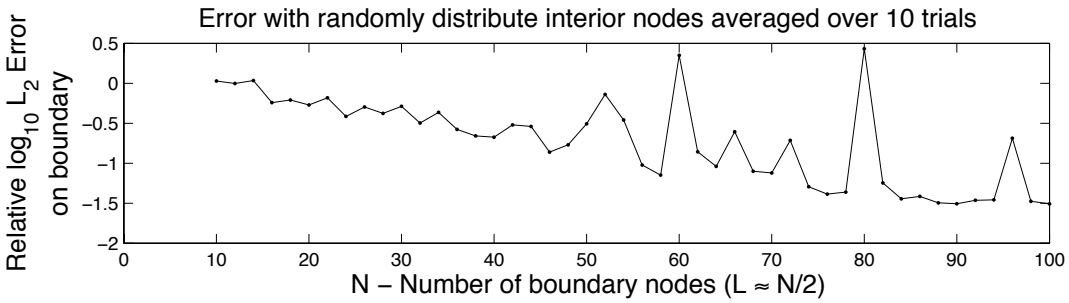

Figure 4: Error with an increasing number of nodes. The interior nodes are randomly distributed.

Finally for error analysis, we wish to test how accuracy depended on the ratio of interior nodes to boundary nodes. Figure 6 shows the error for discretizations with a fixed number of interior nodes and a varying number of boundary nodes. Figure 7 fixes the number of boundary nodes and shows the error with respect to the number of interior nodes. In each case, the interior nodes are distributed over a lattice.

Both figures show that the error decreases with the addition of either boundary nodes or interior nodes. However, adding boundary nodes has seems to have the larger impact. Intuitively this makes sense since, in the Poisson problem, the interior nodes only contribute to the approximation of the residual term in the domain. Alternatively, the addition of boundary nodes contributes to a better approximation of the residual on the boundary, a

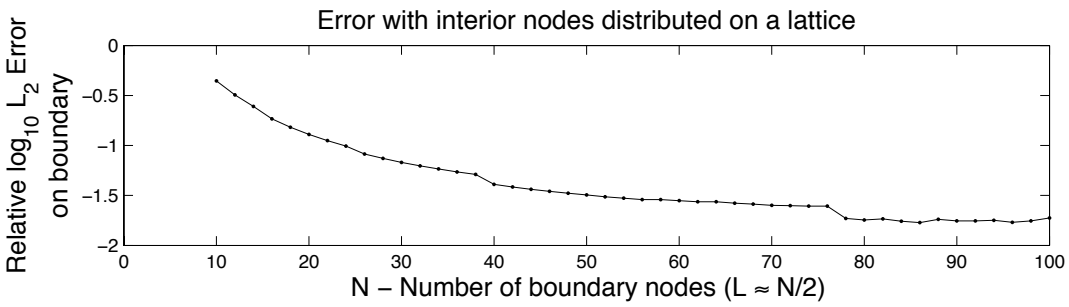

Figure 5: Error with an increasing number of nodes. The interior nodes are distributed over a lattice.

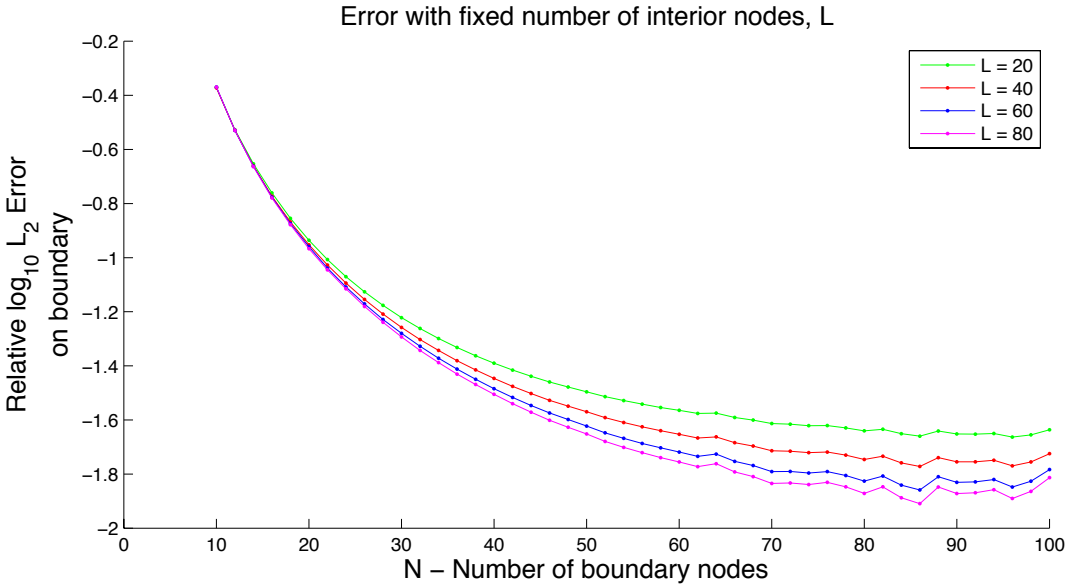

Figure 6: Error with an increasing number of boundary nodes for a fixed number of interior nodes distributed on a lattice. Note: We only show errors for even N for this plot. There was trouble generating this plot for odd N. It is assumed that there is a simple programming error which has yet to be found.

better approximation of the boundary conditions, and a more refined discretization of the unknown solution.

4.2 Heat Solver Validation

Error analysis for the heat equation was performed by examining the problem

$$\frac{\partial}{\partial t}u - \Delta u = b \qquad (\vec{x}, t) \in \Omega \times (0, T] \tag{75}$$

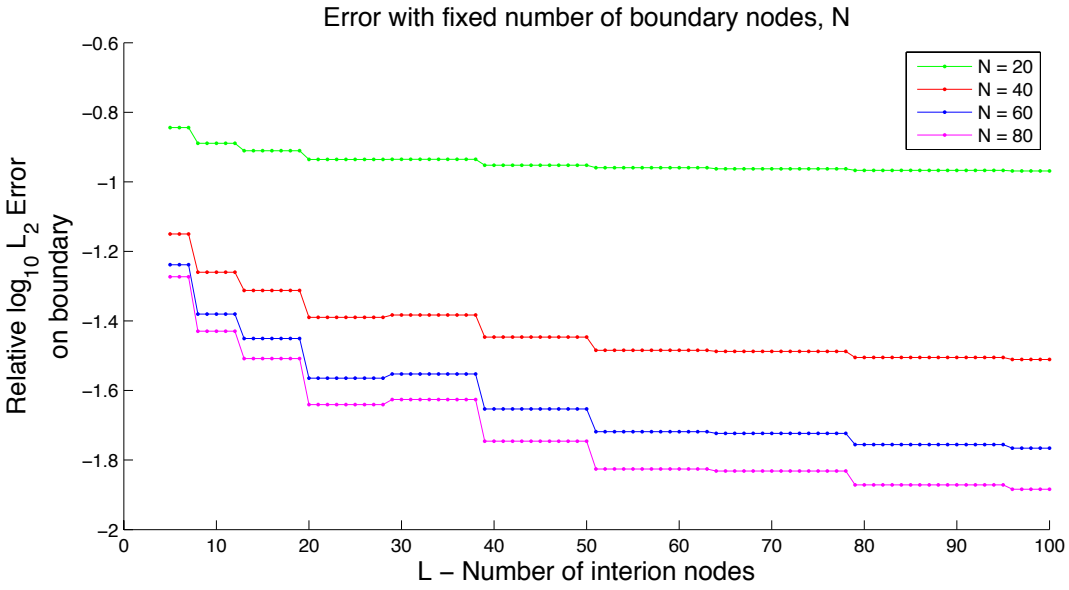

Figure 7: Error with an increasing number of interior nodes distributed on a lattice for a fixed number of boundary nodes . The error is step-like since we cannot specify the exact number of boundary nodes. The plateaus correspond to a set of desired values of L that generate the same number of interior nodes.

$$u(\vec{x},0) = u_0 \qquad \vec{x} \in \Omega \tag{76}$$

$$q = g \qquad (\vec{x},t) \in \Gamma \times (0,T] \tag{77}$$

where Ω is chosen to be the disk of radius one centered on the origin. For validation of the algorithm we approximated a particular solution. We used the function

$$u(x,y,t) = \frac{e^{-x^2-y^2}}{1+t}\left(\sin(x)+\cos(y)\right) \tag{78}$$

with appropriate source term, initial condition, and Neumann boundary condition to satisfy (75) – (77).

Unlike the Poisson equation, the heat equation posed is a Neumann problem so DRM will be used to approximate the potential. Furthermore, since the residual is now a function of the unknown potential, we must solve for the unknown not just at the boundary nodes, but also on the interior nodes.

For validation of the DRM heat equation solver, we first examine the error of the approximation on the domain $\bar{\Omega} \times (0,T]$ using a various number of time steps. As the number of time steps, N_t, increases, the size of each time step, Δt, decreases. The more time steps used, the more accurate the solution is expected to become. The penalty for the additional accuracy is the requirement of additional computational work. Since the spatial domain is discretized as well, for a fixed discrete geometry we expect a minimum error in the solution

no matter how fine the time step size. To measure the error of an approximation, we employ the relative $L_2(\bar{\Omega} \times (0,T])$ norm which is defined as

$$\|u\|_{L_2(\bar{\Omega}\times(0,T])} = \int_0^T \int_\Omega u^2 \, d\Omega \, dt. \tag{79}$$

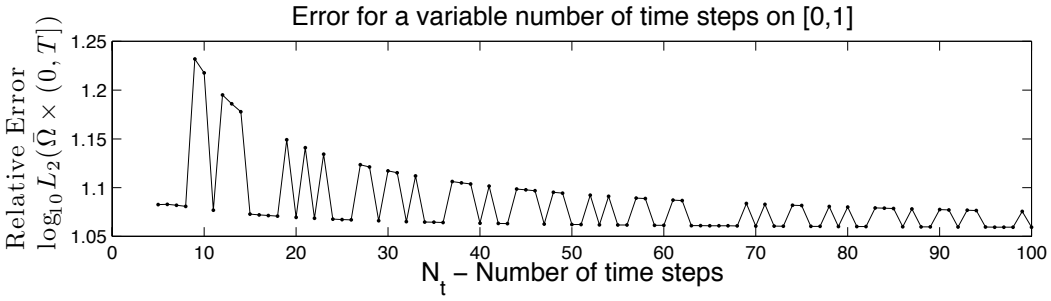

Figure 8: Total error for different numbers of time steps on the interval $[0, 1]$. Notice that the error does not tend to zero. Increasing the number of time steps only causes convergence to an approximation. To converge to the solution, the spacial discretization must be refined as well.

Figure 8 shows the relative $L_2(\bar{\Omega} \times (0,T])$ error for various numbers of time steps. The geometry is discretized using a lattice distribution for the interior nodes with $N = 50$ and $L \approx N/2$ as seen in section 4.1. The error can be seen to decrease as more time steps are used. Furthermore, it can be seen that the error does not converge to zero. This intrinsic error is an artifact from the discretization of the geometry. The important information to take away from the figure is the convergence to a single approximation as the time discretization is refined.

With any time marching algorithm, it is of interest to see how error increases at each time step. To examine the local time stepping error, we fix the number of time steps and measure the relative $L_2(\bar{\Omega})$ error at each time step. The result can be seen in figure 9. As we can see, the error between the exact solution and the approximation increases with time. This is expected as the approximation at time step $n_t + 1$ is computed based on the approximation at time step n_t. Thus the approximation at time step $n_t + 1$ inherits all of the error in time step n_t in addition to any error the algorithm contributes while taking a step forward in time. We measure and plot the percent error increase at each time in figure 10 and we see that while the over all error increase, the jump in error at each time step is bounded.

In addition to analyzing how error changes compared to the number of time steps, it is also interesting to see how the error behaves based on the discretization of the geometry. We measured the $L_2(\bar{\Omega} \times (0,T])$ error for various numbers of nodes on both the interior and the boundary. In testing we found that the error behaved sporadically for $N > 40$. The exact reason for the behavior is still being investigated. First we examine the error convergence for an increasing number of N with $L \approx N/2$. As before, we examine the interior nodes both

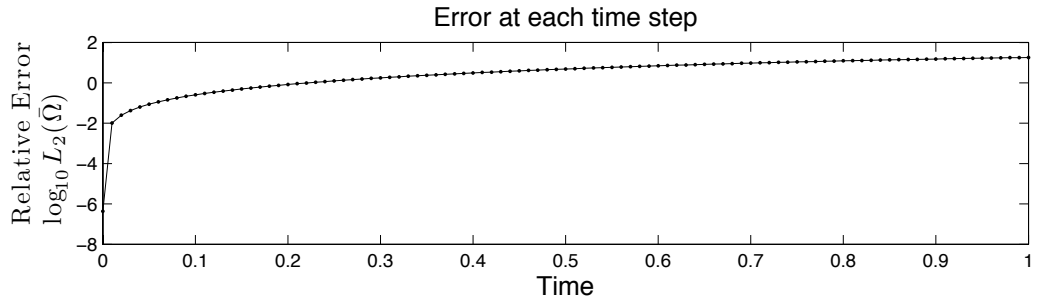

Figure 9: Error at each time step between the exact solution and approximation.

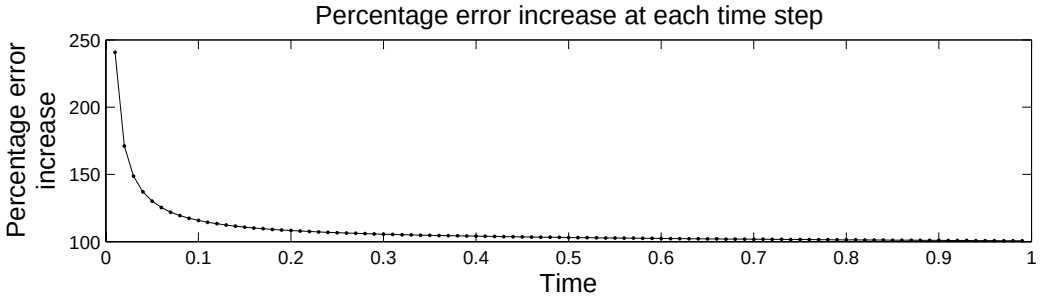

Figure 10: Percent error increase in the approximation at each time step.

distributed randomly and on a lattice. Figure 11 shows the change in error as the number of nodes is increased for each distribution. When using the lattice distribution, the error appears stagnant despite the increasing number of nodes used in the approximation. On the other hand, the error associated with the randomly distributed interior nodes decreases as the tot number of nodes increases. The drop is significant. When $30 \leq N \leq 40$, the error for the two distributions is comparable.

As in section 4.1, it is also interesting to see how error decreases with the addition of boundary or interior nodes. Figure 12 shows the $L_2(\bar{\Omega} \times (0, T])$ error for various fixed numbers of interior nodes as the number of boundary nodes is increased. We see that the error steadily decreases as the number of boundary nodes approached $N = 40$.

Figure 13 shows the error of the approximation for fixed values of boundary nodes as the number of interior nodes is increased. Once again, we see that the error decreases as the number of interior nodes are increased. Notice, unlike the with the boundary nodes, the number of interior nodes can be continually increased and the error decay remains steady.

Compared to the Poisson case, we see the opposite effect when adding interior nodes to approximate the heat equation. The addition of interior nodes produces a much more accurate approximation that the addition of boundary nodes. In addition, during testing, we were able to add a significant number of interior nodes with no computational irregularities.

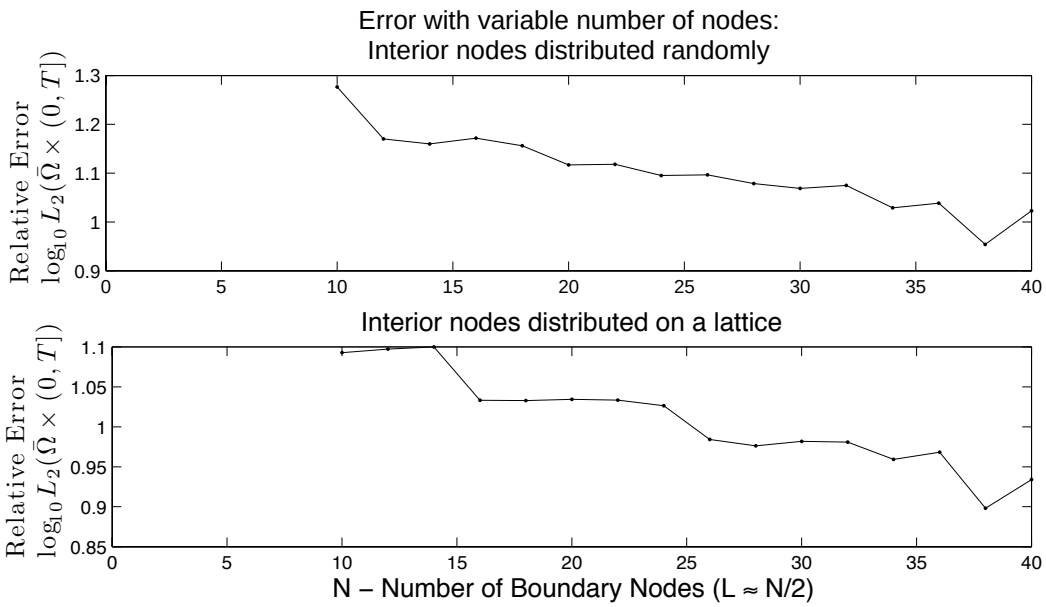

Figure 11: Total Error with an increasing number of spatial nodes. The interior nodes are distributed randomly and over a lattice grid with $N_t = 100$.

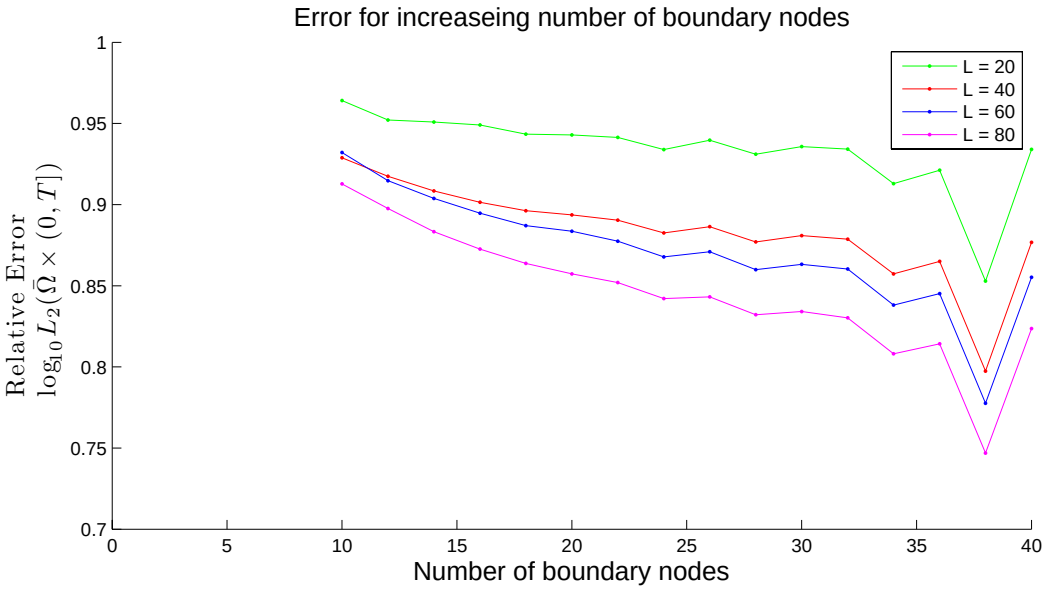

Figure 12: Total error with an increasing number of boundary nodes for a fixed number of interior nodes distributed on a lattice.

5 Future Work

The yearlong project was to create a DRM-based software package as a step toward simulating crystals falling into and cooling a magma ocean. Great progress has been made toward

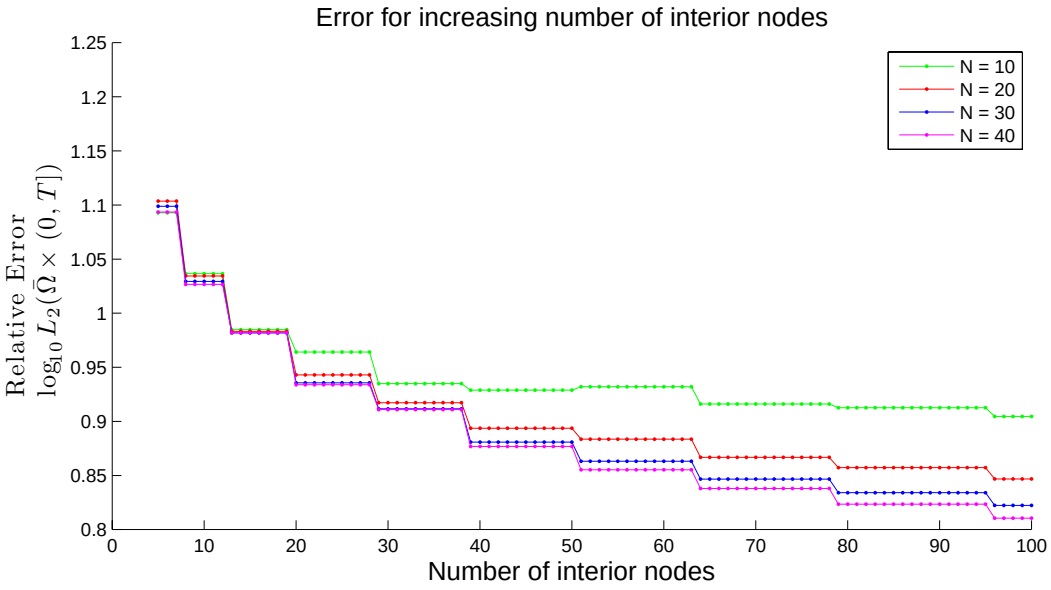

Figure 13: Total error with an increasing number of interior nodes distributed in a lattice for a fixed number of boundary nodes.

achieving the end goal. A highly-optimized software package for solving the heat equation using DRM has been developed and interfaced with an existing Stokes flow solver. However, there is still additional work and research to be done outside the scope of the project. The next few sections detail some of the paths for investigation that will soon be followed.

5.1 Integration of Stokes Solver

To model the crystal settling behavior in magma oceans, it is necessary to couple a Stokes solver with a heat equation solver. One of the initial goals of the project was to integrate an existing Stokes flow solver with an existing Poisson solver. The two solvers were integrated using the data structure described in section 3.5. Also, an effort was made to unify the basic subroutines shared between the solvers.

Despite the work done to integrate the two solvers, the Stokes flow solver was never fully implemented along side the DRM code for solving the heat equation. There are several reasons for the lack of implementation. First, the project's main focus has always been to implement the DRM heat equation solver. This led to neglect in fixing programming errors and validating the Stokes solver. Second, when using the Stokes solver, we wish to advect the particles over time. There are several implementation details of the advection that are currently being researched by other students in the geology department. It was therefore decided to postpone fully implementing the Stokes flow solver until all required research is completed.

5.2 Choice of Basis Functions

The corner stone to DRM is the approximation of the residual term using functions that satisfy (19). The choice most popular in literature is the radial cone presented in section 3.4. While approximations using the radial cones worked well for the problems sampled in section 4, they may lack the robustness needed for more complicated problems such as the end goal of modeling the settling of tens of thousands of crystals.

The first issue with the radial cone functions is the lack of knowing if the radial cones do indeed form a mathematical basis for continuous functions on the required domains. If the functions do not form a basis, the approximation to the residual term may degrade as more computation nodes are used. In dynamic domains, such as those seen when the domain is subject to advection from a viscous Stokes flow, it may not be immediately clear what number and placement of radial cones will provide an accurate estimation of the residual term. If the set of functions chosen to approximate the residual term is known to form a mathematical basis, this issue is completely avoided.

In addition to the choice of basis function, their optimal positioning is also a topic of further investigation. Section 4.1 and 4.2 showed that the error in the approximation was lower when the interior nodes were distributed on a lattice compared to when the nodes were distributed randomly. It is important to recall that DRM uses the interior nodes to incorporate information on the residual term's behavior on the domain. Therefore we wish to distribute the interior nodes in such a way that captures the desired information. On a dynamic domain, it may become too computationally costly to distribute interior nodes based on a structured scheme. Thus, it is important to continue analyzing the accuracy of problems where interior nodes are randomly distributed to see if it is a viable distribution scheme.

5.3 Fast Multipole Method

The linear systems used in DRM are generally dense and asymmetric. The lack of structure limits the number of computational nodes that can be used in any simulation as general solvers run in $\mathcal{O}(\mathcal{N}^3)$ where $\mathcal{N}$ is the total number of computational nodes. If the linear system were sparse, the system could be solved using an iterative solver that runs in $\mathcal{O}(k\mathcal{N}\log\mathcal{N})$ where k is the number of iterations performed.

The Fast Multipole Method [17] takes advantage of the geometric structure of the problem along with the algebraic properties of the matrix entries to approximate matrix vector multiplications in $\mathcal{O}(p\mathcal{N})$ where $1 \leq p \leq \mathcal{N}$ is a parameter related to how much error is acceptable in the matrix vector multiplication. In the case of approximating solutions to PDEs, exchanging a little bit of error for computational speed is generally favorable as the approximations are expected to contain some error.

Implementing Fast Multipole Method was listed as an optional addition to the DRM solver. However, due to time constraints, it was not added. Also, the linear systems in the problems examined were very manageable, the largest being 450×450. The small size of linear systems encountered minimized the immediate necessity to implement the Fast

Multipole Method. As the solver is used for larger systems containing tens of thousands of particles, it will be necessary to implement the Fast Multipole Method.

6 Conclusion

Over the course of AMSC663 and AMSC664, a successful approach for integrating an existing Stokes flow and Poisson solver was developed. The Poisson solver was modified to use DRM. The addition of DRM removed the restriction that required the original solver to only use harmonic functions as source terms. Next the DRM Poisson solver code was modified to solve the heat equation. Along the way, the existing code was cleaned and modified extensively. Subroutines that were redundant or hard coded were removed and replaced by calls to unified methods. In addition, all linear algebra work was implemented using calls to the BLAS and LAPack libraries and OpenMP was used to explicitly parallelize the assembly stage of the code.

The code was validated at several stages and a thorough error analysis was performed. Also, to ensure the code is usable in the future by others, extensive documentation was included. The documentation includes inline comments as well as an electronic reference manual for the code generated by Doxygen. Finally, open issues were addressed. Along with the possibilities of adding new methods to the code such as FMM, existing implementation details were questioned and suggestion were made for future investigation.

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

[9] C. Pozrikidis, "Interfacial Dynamics for Stokes Flow", *Journal of Computational Physics*, Issue 169, 2001, Pages 250-301

[10] 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

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

[12] 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

[13] L. Gaul, M. K$'$ogl, M. Wagner, *Boundary Element Methods for Engineers and Scientists*, Springer, Berlin, 2003

[14] 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

[15] P. W. Partridge and C. A. Brebbia, "Computer implementation of the BEM dual reciprocity method for the solution of general field equations", *Communications in Applied Numerical Methods*, Issue 6, 1990, Pages 83-92

[16] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes: The Art of Scientific Computing*, Cambridge University Text, New York, NY, 2007

[17] 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