Dual Reciprocity Boundary Element Method for Magma Ocean Simulations

Tyler W. Drombosky drombosk@math.umd.edu
Saswata Hier-Majumder saswata@umd.edu †

† Department of Geology

28 April 2010
Physical Motivation

- Earth’s early history is marked by a giant impact with a Mars-sized object
- This led to a substantial amount of interior melting followed by rapid crystallization of this ‘magma ocean’
- How this crystallization took place and crystals settled provide insight into the planet’s rate of cooling
Abstract

- Understanding how crystals settle in a magma ocean is critical for answering questions about Earth’s early history.
- Experiments have been performed that have given insight into this behavior.
- The ability to simulate this behavior numerically is not available.
- Over the course of AMSC663 and AMSC664, a numerical solver based on the Dual Reciprocity Method has been built to address this problem.
1. Brief review of the Dual Reciprocity Method
2. Restatement and evaluation of original goals
3. Summary of where the project is now and where it will go in the future
Table of Contents

1. Brief review of the Dual Reciprocity Method
2. Restatement and evaluation of original goals
3. Summary of where the project is now and where it will go in the future
# Table of Contents

1. Brief review of the Dual Reciprocity Method
2. Restatement and evaluation of original goals
3. Summary of where the project is now and where it will go in the future
The Dual Reciprocity Method solves partial differential equations in the form:

\[-\Delta u = \hat{b} \quad \text{in} \quad \Omega \quad (1)\]

with Dirichlet, Neumann, or mixed boundary conditions.

- The source term, \( \hat{b} \) is potentially non-linear in \( u \).
- In DRM, \( \hat{b} \) is referred to as the residual term.
Approximation of the Residual Term

- Approximate the residual term, by a linear combination of basis functions \( \hat{f}_k \)

\[
\hat{b} = \sum_k \beta_k \hat{f}_k
\]  

(2)

where

\[
\hat{f}_k = 1 + r_k \quad r_k = |x - x_k|
\]  

(3)

- With this basis function we can find \( \hat{u}_k \) such that

\[
-\Delta \hat{u}_k = \hat{f}_k \quad \text{thus} \quad \hat{u}_k = -\frac{r^2}{4} - \frac{r^3}{9}
\]  

(4)
Some Notation

- Denote the flux density as:
  \[ q = \nu \cdot \nabla u \]  
  \hspace{1cm} (5)

  where \( \nu \) is the outward facing normal vector.

- Define the fundamental solution \( u_i^* \) to the Poisson in \( \mathbb{R}^2 \):
  \[ \Delta u_i^* = -\delta(x - x_i) \text{ so } u_i^* = -\frac{1}{2\pi} \ln(r_i) \]  
  \hspace{1cm} (6)
Integration by Parts: LHS

After multiplying the general DRM PDE by the fundamental solution and integrating over the domain, use integration by parts twice:

\[
\int_{\Omega} -\Delta uu_i^* = \int_{\Gamma} -qu_i^* + \int_{\Gamma} \nabla u \cdot \nabla u_i^* \quad (7)
\]

\[
\quad = \int_{\Gamma} -qu_i^* + uq_i^* - \int_{\Omega} \Delta u_i^* u \quad (8)
\]

\[
\quad = \int_{\Gamma} uq_i^* - qu_i^* + c(x_i)u(x_i) \quad (9)
\]

\[c(x_i)\] is a function that indicates what “fraction” of the singularity is in the domain.
Recalling the approximation of the residual term, apply a similar procedure to the RHS:

\[
\int_\Omega \hat{b}u_i^* = \sum_k \beta_k \int_\Omega \hat{f}_k u_i^* \tag{10}
\]

\[
= \sum_k \beta_k \int_\Omega -\Delta \hat{u}_k u_i^* \tag{11}
\]

\[
= \sum_k \beta_k \left[ \int_\Gamma \hat{u}_k q_i^* - \hat{q}_k u_i^* + c(x_i)\hat{u}_k(x_i) \right] \tag{12}
\]
Combining the LHS and RHS:

\[
\int_{\Gamma} u q_i^* - q u_i^* + c(x_i) u(x_i) = \sum_k \beta_k \left[ \int_{\Gamma} \hat{u}_k q_i^* - \hat{q}_k u_i^* + c(x_i) \hat{u}_k(x_i) \right]
\] (13)

Using the method of collocation to distribute nodes on the boundary and cubic spline interpolation to interpolate between boundary nodes, the discretized equation becomes:

\[
H u + G q = (H \hat{U} + G \hat{Q}) \beta
\] (14)
Goals of Project

1. Integrate existing code used to solve the Stokes equation and Poisson equation:

\[-\nabla P + \mu \Delta \vec{v} + \rho \vec{b} = 0\]  \quad \text{and} \quad  \[-\Delta u = b \quad (15)\]

where \( b \) is harmonic

2. Use DRM to solve the Poisson equation for general \( \hat{b} = b \)

3. Use DRM to solve the heat equation

\[\frac{\partial}{\partial t} + \vec{v} \cdot \nabla u - \Delta u = b \quad \text{where} \quad \hat{b} = b - \frac{\partial}{\partial t} - \vec{v} \cdot \nabla u (16)\]
Goals of Project

1. Integrate existing code used to solve the Stokes equation and Poisson equation:

\[-\nabla P + \mu \Delta \vec{v} + \rho \hat{b} = 0 \quad \text{and} \quad -\Delta u = b \quad (15)\]

where \( b \) is harmonic

2. Use DRM to solve the Poisson equation for general \( \hat{b} = b \)

3. Use DRM to solve the heat equation

\[\frac{\partial}{\partial t} + \vec{v} \cdot \nabla u - \Delta u = b \quad \text{where} \quad \hat{b} = b - \frac{\partial}{\partial t} - \vec{v} \cdot \nabla u (16)\]
Goals of Project

1. Integrate existing code used to solve the Stokes equation and Poisson equation:

$$-\nabla P + \mu \Delta \vec{v} + \rho \vec{b} = 0$$
and

$$-\Delta u = b$$

(15)

where $b$ is harmonic

2. Use DRM to solve the Poisson equation for general $\hat{b} = b$

3. Use DRM to solve the heat equation

$$\frac{\partial}{\partial t} + \vec{v} \cdot \nabla u - \Delta u = b$$

where

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

(16)
Goal 1: Integrate Solvers with Data Structures

- Used to transfer information between solvers
- Very lean, contains only essential data
- Built to inspire code structure

(mesh)

(ptcl 1)

(ptcl n)

(whole space properties)

((x,y))

((x_{int},y_{int}))

((u,q))

((v,v_{int}))

(Single particle properties)
Goal 1: Integrate Solvers with Subroutines

- Each code had different subroutines for the same task
  - Some functions appear hard coded multiple times
  - Required different inputs
  - Returned incompatible outputs
- Unified assembly routines
  - Matrices for the Stokes and heat equations are built in a similar manner
  - Learn one code, learn them both
- Replaced custom and Numerical Recipes methods with those found in LAPack and BLAS
Goal 2: Use DRM to Solve Poisson Equation

- Used to validate assembly of DRM matrices
- Spacial error was measured and analyzed
Goal 2: Use DRM to Solve Poisson Equation

- Noticed Error “bounce back"
  - Error decreases quadratically as number of nodes increased
  - Error then increases once a certain number of nodes is used
- Investigated potential causes
  - Recall $\beta = \hat{F}^{-1}b$
  - As $N$ increases, $\kappa(\hat{F})$ increases leading to inaccurate $\beta$
- Solutions
  - Switch to double precision
  - Use different basis functions
Goal 2: Use DRM to Solve Poisson Equation

- Noticed Error “bounce back"
  - Error decreases quadratically as number of nodes increased
  - Error then increases once a certain number of nodes is used

- Investigated potential causes
  - Recall $\beta = \hat{F}^{-1}b$
  - As $N$ increases, $\kappa(\hat{F})$ increases leading to inaccurate $\beta$

- Solutions
  - Switch to double precision
  - Use different basis functions
Goal 2: Use DRM to Solve Poisson Equation
Goal 3: Use DRM to Solve Heat Equation

- Recall the DRM matrix equation

\[ Hu + Gq = (H\hat{U} + G\hat{Q})\hat{F}^{-1}\hat{b} \] (17)

where \( \hat{b} = b - \frac{\partial}{\partial t}u - \vec{v} \cdot \nabla u \)

- We want to discretize \( \vec{v} \cdot \nabla u \)

- Davis et al. showed the approximation

\[ \frac{\partial}{\partial x}U = \frac{\partial \hat{F}}{\partial x}\hat{F}^{-1}U \quad \text{and} \quad \frac{\partial}{\partial y}U = \frac{\partial \hat{F}}{\partial y}\hat{F}^{-1}U \] (18)

are valid
Goal 3: Use DRM to Solve Heat Equation

The DRM matrix equation for the heat equation becomes

\[ Hu + Gq = (H\hat{U} + G\hat{Q})\hat{F}^{-1} \left( b - \frac{\partial}{\partial t} U - V_x \cdot \frac{\partial \hat{F}}{\partial x} \hat{F}^{-1} U - V_y \cdot \frac{\partial \hat{F}}{\partial y} \hat{F}^{-1} U \right) \] (19)

- Notice that the linear system is a first order ordinary differential equation with variable \( t \) and unknown \( U \)
- Use Crank-Nicolson method to time step
Goal 3: Use DRM to Solve Heat Equation

- Used to validate time stepping method
- Time stepping error was measured and analyzed

Error Plot coming soon
Goal 3: Use DRM to Solve Heat Equation

Single particle movie: Still trying to figure out the technicalities of getting this in the slides
Goal 3: Use DRM to Solve Heat Equation

Multiparticle movie
Implementation Details

- Rewrote code to handle singular integrals
  - Generalized for use in Stokes and heat solvers
  - Optimized and debugged
- Implemented OpenMP
  - Matrix Assembly explicitly parallelized
  - Used the Intel Math Kernel Library for LAPack and BLAS which is parallelized with -openmp compile time flag
- Wrote domain back solver
  - Once boundary values are computed, domain values can be quickly back calculated
  - Separate program that shares many of the same libraries
- Provided detailed comments on code using Doxygen
Future Work

- Stokes solver currently not running
  - Separate modifications being one on the solver
  - Waiting until all modification to be complete before adding it solver
- No Fast Multipole Method
  - Makes the DRM linear systems sparse
  - Greatly increases code performance if systems are large
  - Current matrices from two dimensional problems are small
- Basis functions
  - Investigate using other basis functions
  - Functions with compact support
  - For whole space problems, basis functions should match asymptotically match the solution
Schedule

- Phase I (present - early November)
  - Merge Stokes flow and Poisson solver code ✔
- Phase II (November - December)
  - Test and validate steady-state code ✔
  - Optimize code ✔
- Phase III (December - early February)
  - Add Dual Reciprocity code ✔
  - Add Fast Multipole Method code (optional)
- Phase IV (February - March)
  - Test and validate Dual Reciprocity code ✔
  - Optimize code
    - Interface with Intel Math Kernel Library ✔
    - Parallel implementation with OpenMP ✔


Questions?

Multiparticle movie here