High Performance Techniques Applied in Partial Differential Equations Library

Shilei Lin
*College of Saint Benedict/Saint John's University*, lin.shilei@outlook.com

Follow this and additional works at: https://digitalcommons.csbsju.edu/honors_thesis

Part of the Hydraulic Engineering Commons, Numerical Analysis and Scientific Computing Commons, and the Partial Differential Equations Commons

**Recommended Citation**
https://digitalcommons.csbsju.edu/honors_thesis/40

This Thesis is brought to you for free and open access by DigitalCommons@CSB/SJU. It has been accepted for inclusion in All College Thesis Program, 2016-present by an authorized administrator of DigitalCommons@CSB/SJU. For more information, please contact digitalcommons@csbsju.edu.
High Performance Techniques Applied in Partial Differential Equations Library

AN ALL COLLEGE THESIS

College of Saint Benedict  Saint John’s University

In Partial Fulfillment
of the Requirements of Distinction
In the Department of Computer Science and Mathematics
by

Shilei Lin

May 2018
PROJECT TITLE:
High Performance Techniques Applied in Partial Differential Equations Library

BY: SHILEI LIN

Approved By:

Michael Heroux
Scientist in Residence of Computer Science

Michael Tangredi
Associate Professor of Mathematics

Imad Rahal
Associate Professor of Computer Science

Imad Rahal
Chair, Department of Computer Science

Robert Hesse
Chair, Department of Mathematics

Molly Ewing
Director, All College Thesis Program
ABSTRACT

This thesis explores various Trilinos packages to determine a method for updating the deal.ii library, which specializes in solving partial differential equations by finite element methods. It begins with introducing related concepts and general goals, followed by exploring computational and mathematical methods which are analytical solutions of one-dimensional Boussinesq equations and developing newer prototypes for solvers in deal.ii based on Trilinos packages. After demonstrating the methods, it indicates the reducing solving time in newer prototypes. Based on results from the prototype, similar methods are applied to update the deal.ii library. In the end, a testing program is exploited to demonstrate the improvement in performance for deal.ii.
## Contents

1 Introduction .......................................................... 1  
  1.1 Introduction to Parallel Computing .......................... 1  
  1.2 Introduction to Template Programming ..................... 1  
  1.3 Introduction to Boussinesq Equation ....................... 1  

2 Contributions to this Topic ........................................ 2  

3 Problem Statement .................................................. 2  

4 Background .......................................................... 3  
  4.1 One-Dimensional Boussinesq Equation ...................... 3  
  4.2 The deal.ii ..................................................... 4  
  4.3 Trilinos Packages .............................................. 4  
    4.3.1 Petra Packages ........................................... 4  
    4.3.2 Solver Packages .......................................... 5  
    4.3.3 Preconditioning Packages ............................... 5  
    4.3.4 Finite Element and Matrix Generation Package ....... 5  
    4.3.5 Summary of Trilinos Packages ......................... 6  

5 Methods ............................................................... 7  
  5.1 Solving the One Dimensional Flow Equation Analytically .. 7  
    5.1.1 The Important Properties of $H(\eta)$ .................. 9  
    5.1.2 Polynomial Approximation Of the Solution ............. 12  
  5.2 Prototype Development ........................................ 15  
  5.3 Updating deal.ii .............................................. 16  
    5.3.1 Updating Solver from AztecOO to Belos ............... 16  
    5.3.2 Converting Epetra to Tpetra ............................ 20  

6 Results ............................................................... 23  
  6.1 One-Dimensional Boussinesq Equation Result .............. 23  
  6.2 Prototypes Comparison ....................................... 25  
    6.2.1 Number of Iterations Comparison ....................... 25  
    6.2.2 Solving Time Comparison ................................ 26  
  6.3 Comparison of Trilinos Linear Solvers in deal.ii ....... 28  
    6.3.1 Solving time Comparison ................................ 28  
    6.3.2 Number of Solver Iteration Comparison ............... 30  

7 Conclusion .......................................................... 31  
  7.1 Performance of Prototypes ................................. 31  
  7.2 Performance of Updated deal.ii ............................ 31  
  7.3 Future Work .................................................... 31
8 Appendix
8.1 Source Code .................................................. 32
  8.1.1 Source Code for Prototype .............................. 32
  8.1.2 Source Code for Updated deal.ii ....................... 32
8.2 Computing Specifications .................................... 32
8.3 Prototypes Detailed Comparison Table ...................... 33
8.4 Code for Iteration ............................................ 34
8.5 Installation Notes and Scripts ................................ 34
  8.5.1 The Trilinos Packages ................................... 34
  8.5.2 The deal.ii Library ........................................ 35
  8.5.3 Project Required Libraries .............................. 35

List of Tables
1 Trilinos Packages[8] ........................................... 6
2 Epetra to Tpetra Data Type Conversion ....................... 20
3 Epetra to Tpetra Functions Conversion ....................... 22
4 Values of the Parameters for $1 < \mu < 3$. [12] ............... 23
5 Prototypes Detailed Comparison Table ....................... 33

List of Listings
1 Problem Generation for Prototypes[9]. ....................... 15
2 New Namespaces and Type Abbreviations in trilinos_solver.h 16
3 The trilinos_solver.h before modification .................... 17
4 The trilinos_solver.h after modification ..................... 17
5 solve in trilinos_solver.cc before modification ............... 17
6 solve in trilinos_solver.cc after modification ............... 17
7 do_solve in trilinos_solver.cc after modification ............ 18
8 Data types for Tpetra Structure .............................. 20
9 Specification .................................................... 32
10 Iteration.m ................................................... 34
11 Trilinos Installation and CMake Script ..................... 34
12 deal.ii Installation and CMake Script ...................... 35

List of Figures
1 Expected piezometric solution curve for $H$. [12] ............... 3
2 Modified postulated piezometric solution curve for $H$. [12] .... 9
3 Piezometric $H$ versus $\eta$ for various values of $\mu$. [12] ....... 23
4 Values of the parameters $v$ and $\eta_k$ versus $\mu$. [12] .......... 24
Values of the parameters $\lambda$ and $L$ versus $\mu$.\[12\] 

Values of the parameters $\lambda$ and $A$ versus $\mu$.\[12\]

A line chart of all trials for $MLAztecOOEpetra.cc$, $MLBelosEpetra.cc$, and $MueLuBelosTpetra.cc$ with problem sizes from $50^3$ to $400^3$. This plot shows the range in number of iterations for each file.

A column chart of 4 time trials for $MLAztecOOEpetra.cc$, $MLBelosEpetra.cc$, and $MueLuBelosTpetra.cc$ with a problem size of $50^3$.

A column chart of 4 time trials for $MLAztecOOEpetra.cc$, $MLBelosEpetra.cc$, and $MueLuBelosTpetra.cc$ with a problem size of $200^3$.

A column chart of 4 time trials for $MLAztecOOEpetra.cc$, $MLBelosEpetra.cc$, and $MueLuBelosTpetra.cc$ with a problem size of $350^3$.

A column chart of all trials for $MLAztecOOEpetra.cc$, $MLBelosEpetra.cc$, and $MueLuBelosTpetra.cc$ with a problem size from $50^3$ to $400^3$.

A scatter chart at Timestep 0 for $step-32.cc$. The measured time is solving time. This plot shows the range in time performance for Belos and AztecOO solver. The solving results listed at the bottom of this chart have the same value for both solvers.

A scatter chart at Timestep 21 for $step-32.cc$. The measured time is solving time. This plot shows the range in time performance for Belos and AztecOO solver. The solving results listed at the bottom of this chart have the same value for both solvers.

A scatter chart at Timestep 51 for $step-32.cc$. The measured time is solving time. This plot shows the range in time performance for Belos and AztecOO solver. The solving results listed at the bottom of this chart have the same value for both solvers.

A scatter chart at Timestep 101 for $step-32.cc$. The measured time is solving time. This plot shows the range in time performance for Belos and AztecOO solver. The solving results listed at the bottom of this chart have the same value for both solvers.

A table lists the number of iterations where ‘B’ stands for ‘Belos’ and ‘A’ stands for ‘AztecOO’. ‘Calling’ represents the $n^{th}$ time of calling $BlockSchurPreconditioner::vmult$ function in $step-32.cc$.
1 Introduction

Real world phenomenon in physics such as heat dynamics, fluid dynamics, and quantum mechanics can mostly be represented by systems of partial differential equations (PDEs). Also, PDE systems can be enormous since they mostly describe phenomenon observed across dimensions. Therefore, PDE systems of this size are normally solved by computational software which is capable of formulating and solving large PDE systems.

Nowadays, solving large systems is still time-consuming. Thus, scientists are developing new software and hardware to improve efficiency. One approach to improve performance is by revising data structures and solving methods mainly through two high performance techniques: parallelism and templates.

1.1 Introduction to Parallel Computing

As opposed to traditional sequential computing, parallel computing takes advantage of computer architectures in which there are multiple processors. The goal of parallel computing is to reduce running time.[8]

The distributed memory model and the shared memory model are two major types of parallel computing models. For instance, the Message Passing Interface (MPI), a distributed memory model, relies on distributed networks, which is flexible and expressive. The OpenMP, a shared memory model, relies on multi-core processors, which is easier to program and debug[11].

1.2 Introduction to Template Programming

‘Templates’ are a distinct feature in the C++ programming language that allows functions and classes to handle more data types or structures. Adopting ‘template’ programming has several benefits on my project, one of which reduces the repetition in codes and allows programs to accept user-made data types besides the predefined ones.

1.3 Introduction to Boussinesq Equation

In partial differential equations (PDEs), Boussinesq equations are often found in fluid dynamics, describing groundwater flow through an aquifer as a nonlinear parabolic PDE. The equation is named after Joseph Boussinesq, who first derived it in response to John Scott Russell’s observation of the “wave of translation” (also known as a solitary wave or soliton). Boussinesq’s 1872 paper introduces the equations now known as the Boussinesq equations[2].

\[1\]This project needs MPI-able environment.
2 Contributions to this Topic

This thesis intends to provide several contributions to the scientific computing field. First, it provides a detailed analytical solution for one-dimensional Boussinesq equations (flow equation). Secondly, it demonstrates reasons to update deal.ii by examining differences between various data services and solver abilities. Lastly, it provides the potential conversions and modifications of deal.ii with more recent Trilinos packages.

3 Problem Statement

When trying to solve a large partial differential system by using computers, a powerful software can be highly effective. Hardware technology advances rapidly, however, software takes years to catch up. When modifying and updating codes, we need to reflect the current computer architectures and programming languages.[8]

In this thesis, I propose to update the Trilinos wrapper classes in deal.ii. Currently, deal.ii (version 8.4.1) uses Epetra for its vector and matrix representations, ML and MueLu(partially) as its preconditioning, and AztecOO as its iterative linear solver. Throughout the years, Trilinos (version 12.10) has seen a newer version for data representation named Tpetra, and a newer version of linear solver named Belos.

To demonstrate the improvement of runtime performance, my intention is to use one of the tutorial programs in deal.ii ‘step-32’ which are embedded with Trilinos wrapper classes of deal.ii. The step-32 constructs and solves multi-dimensional Boussinesq equations. Thus, to have a better understanding of Boussinesq equations, my project explores analytical solution of one-dimensional Boussinesq equations as discussed in a 1984 paper written by Christos D. Tzimopoulos Panagiotis K. Tolikas, and Epaminondas G. Sidiropoulos.
4 Background

4.1 One-Dimensional Boussinesq Equation

In general, the one-dimensional Boussinesq Equation has the form

\[
\frac{\partial h}{\partial t} = K \frac{\partial}{\partial x} \left( \frac{h}{S} \frac{\partial h}{\partial x} \right) \tag{4.1}
\]

where \( K \) (in meters per second) and \( S \) (dimensionless) are hydraulic conductivity and specific yield of the aquifer, respectively; \( h \) (in meters) is the depth of water from the impermeable substratum; \( x \) is the horizontal distance from the origin; and \( t \) is the time. The boundary conditions for the equation are given by

\[
x = 0 \quad t > 0 \quad h = h_1 \tag{4.2}
\]
\[
x > 0 \quad t = 0 \quad h = h_0 \tag{4.3}
\]

where \( h_0 \) the initial depth of the water from the impermeable substratum, and \( h_1 \) the depth of water from the impermeable substratum at the origin. We consider this case as an abrupt raising of piezometric head at the origin, i.e., \( h_1 > h_0 \) [12]. We can approximate the solution as following:

![Figure 1: Expected piezometric solution curve for \( H \). [12]](image)
4.2 The deal.ii

‘Deal’ is the abbreviation of Differential Equations Analysis Library and ‘II’ indicates that it is the successor of ‘Deal’. The deal.ii library is an open source C++ library that aims to solve partial differential equations by finite element methods. It allows rapid development of modern finite element codes via providing simple meshes programming content. The deal.ii library was introduced at Universität Heidelberg, Germany, based on works by the Numerical Methods Group, which concentrated in finite element methods and error estimation.[6]

4.3 Trilinos Packages

‘Trilinos’ is a collection of open-source software libraries, called packages, intended to be used as building blocks for the development of scientific applications. The word ‘Trilinos’ is from Greek and means ‘a string of pearls’, suggesting a number of software packages linked together by a common infrastructure. Trilinos was developed at Sandia National Laboratories from a core group of existing algorithms and utilizes the functions of software interfaces such as the BLAS, LAPACK, and MPI.[5]

4.3.1 Petra Packages

Petra is the Greek word for ‘foundation’. Therefore in Trilinos, Epetra, Tpetra and Xpetra are representers for basic mathematical structures such as vectors, matrices, and graphs. In general, all Petra packages are capable of parallel execution on distributed memory machines.

1. Epetra
   The ‘E’ stands for ‘essential’ and defines the basic classes for distributed matrices and vectors, linear operators and linear problems. All packages in Trilinos support Epetra, which means that each package accepts Epetra objects as input[10]. Epetra supports double-precision numbers, floating point data, which can be extended to 64-bit indices. This allows powerful combinations among the various Trilinos functions. Epetra provides a high level of portability and stability. [8]

2. Tpetra
   The ‘T’ stands for ‘template’. Tpetra can create templated distributed linear Algebra objects. Because Tpetra uses templates technique, it allows Tpetra to accept more data types than Epetra. In particular, Tpetra is based on two major data types: ordinal type and scalar type. The ordinal type is used for storing countable items, i.e. the number of non-zero elements in a matrix. The scalar type is the type of stored data, which can be varying from a complex number to a small $2 \times 2$ matrix. [10]

---

2The MueLu package only accepts Epetra or Tpetra objects wrapped by Xpetra.
3. **Xpetra**

Xpetra is a wrapper interface which accepts both Epetra and Tpetra objects. The Xpetra also uses template technique like Tpetra. Xpetra enables algorithm developers to write to a single interface but be able to use either Epetra or Tpetra. Most importantly, Xpetra is used by MueLu preconditioner. [7]

4.3.2 **Solver Packages**

AztecOO and Belos are both iterative solver packages in Trilinos. The linear system has the form $Ax = B$, where $A$ is the left-hand side $n$ by $n$ matrix, $X$ is the solution, and $B$ is the right-hand side.

1. **AztecOO**

   AztecOO is a linear solver package based on preconditioned Krylov methods. It only supports Epetra objects. [10]

2. **Belos**

   Belos provides next-generation iterative linear solvers and a powerful linear solver developer framework. Belos supports both Epetra and Tpetra objects. Belos also enables template, because user can benefit from Belos-defined abstract base classes, which can be considered as more efficient.[10]

4.3.3 **Preconditioning Packages**

ML and Muelu are both preconditioner packages in Trilinos.

1. **ML**

   ML is the algebraic (M)ulti(L)evel preconditioner package which has scalable preconditioning capabilities for a variety of problems. It can be used as the preconditioner for both AztecOO and Belos solvers, however, it only supports Epetra objects.[10]

2. **MueLu**

   MueLu is an extensible algebraic multi-grid library that is part of the Trilinos packages. MueLu only works with Xpetra interface which wraps either Epetra or Tpetra objects as long as the program is consistent with its data structure. The library is written in C++ and enables templated traits. Like Tpetra, The MueLu package allows for different ordinal and scalar types. Also, the MueLu package is designed to support various computer architectures from supercomputers to personal computers.

4.3.4 **Finite Element and Matrix Generation Package**

Galeri is the Greek word for Gallery. The Galeri can generate a variety of distributed linear systems. The Galeri packages can also generate several well-known finite element and finite difference matrices.
### 4.3.5 Summary of Trilinos Packages

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Epetra</td>
<td>Essential package for building mathematical structures. Objects based on Epetra can be used in many other solver/preconditioning packages.</td>
<td>It is currently used in deal.ii-8.4.1.</td>
</tr>
<tr>
<td>Tpetra</td>
<td>A template based kernel for building mathematical structures. Also, it is a newer generation of Epetra.</td>
<td>To replace Epetra in deal.ii-8.4.1</td>
</tr>
<tr>
<td>Xpetra</td>
<td>An interface wraps E/Tpetra.</td>
<td>It is used by MueLu. In deal.ii [ preconditioningAMGMulue ] class, Epetra objects are wrapped by Xpetra in order to use MueLu as preconditioner.</td>
</tr>
<tr>
<td>ML</td>
<td>Multigrid preconditioner (MGP). Main multigrid preconditioner package in Trilinos</td>
<td>Currently used by deal.ii.</td>
</tr>
<tr>
<td>MueLu</td>
<td>Newer generation of MGP. MueLu is a flexible, high-performance multigrid solver library.</td>
<td>Currently, it is in deal.ii but not supporting Tpetra.</td>
</tr>
<tr>
<td>AztecOO</td>
<td>Iterative solver. It allows flexible construction of matrix and vector arguments via Epetra matrix and vector classes.</td>
<td>Currently, deal.ii uses AztecOO as wrapper solver for Trilinos objects.</td>
</tr>
<tr>
<td>Belos</td>
<td>A next-generation iterative linear solvers and a powerful linear solver developer framework.</td>
<td>To replace AztecOO in deal.ii</td>
</tr>
<tr>
<td>Galeri</td>
<td>Galeri contains a suite of utilities and classes to generate a variety of (distributed) linear systems.</td>
<td>Prototypes use Galeri to generate problems. Not used in deal.ii</td>
</tr>
</tbody>
</table>

Table 1: Trilinos Packages.[8]
5 Methods

5.1 Solving the One Dimensional Flow Equation Analytically

In Section 4.1, the partial differential equation to be solved is

\[
\frac{\partial h}{\partial t} = \frac{K}{S} \frac{\partial}{\partial x} \left( h \frac{\partial h}{\partial x} \right) \tag{5.1}
\]

The boundary conditions of this partial differential equations are defined as

\[
x = 0 \quad t > 0 \quad h = h_1 \tag{5.2}
\]
\[
x > 0 \quad t = 0 \quad h = h_0 \tag{5.3}
\]

In order to solve this partial differential equation analytically, it needs to be reduced into an ordinary differential equation by substituting \( x \) and \( t \) as,

\[
\eta = \frac{x}{[(Kh_0/S)t]^{1/2}} \tag{5.4}
\]

Based on (5.2), (5.3) and (5.4), rewrite \( h(x,t) \) as \( H(\eta) \) as

\[
h(x,t) = H(\eta) = H\left( \frac{x}{[(Kh_0/S)t]^{1/2}} \right) \tag{5.5}
\]

\[
H = h/h_0 \quad \mu = h_1/h_0 \tag{5.6}
\]

Then, we can reduce (5.1) into ordinary differential equations with boundary condition (5.6) by applying chain-rule. It follows

\[
\frac{\partial h}{\partial t} = \frac{K}{S} \frac{\partial}{\partial x} \left( h \frac{\partial h}{\partial x} \right)
\]
\[
\frac{dH}{d\eta} \cdot \frac{\partial h}{\partial t} = \frac{K}{S} \frac{dH}{d\eta} \left( h \cdot \frac{dH}{d\eta} \cdot \frac{\partial h}{\partial x} \right) \frac{\partial \eta}{\partial x}
\]
\[
\frac{dH}{d\eta} \cdot \frac{-x}{2t \sqrt{\frac{Kth_0}{S}}} = \frac{K}{S} \frac{dH}{d\eta} \left( h \cdot \frac{dH}{d\eta} \cdot \frac{1}{\sqrt{\frac{Kth_0}{S}}} \right) \cdot \frac{1}{\sqrt{\frac{Kth_0}{S}}}
\]
\[
-\frac{dH}{d\eta} \cdot \frac{\eta}{2t} = \frac{K}{S} \frac{dH}{d\eta} \left( h \cdot \frac{dH}{d\eta} \right) \cdot \frac{S}{Kt \cdot h_0}
\]
\[
-\frac{dH}{d\eta} \cdot \frac{\eta}{2t} = \frac{d}{d\eta} \left( h \cdot \frac{dH}{d\eta} \right) \frac{1}{t \cdot h_0}
\]
\[
-\frac{dH}{d\eta} \cdot \frac{\eta}{2t} = \frac{d}{d\eta} \left( h \cdot \frac{dH}{d\eta} \right)
\]
\[
-\frac{dH}{d\eta} \cdot \frac{\eta}{2t} = \frac{d}{d\eta} \left( H \cdot \frac{dH}{d\eta} \right)
\]
Thus, the result of simplification produces the ordinary differential equation with boundary conditions (5.2) and (5.3) as
\[-\frac{\eta}{2} \frac{dH}{d\eta} = \frac{d}{d\eta} \left( H \frac{dH}{d\eta} \right) \]
(5.8)
\[\eta = 0 \quad H = \mu \]
(5.9)
\[\eta = \infty \quad H = 1 \]
(5.10)
Before we take integral to solve (5.8), we need to extract some properties from the differential equations first. From (5.8), at the origin using product rule, it follows
\[\frac{d}{d\eta} \left( H \frac{dH}{d\eta} \right) = \frac{dH}{d\eta} \cdot \frac{dH}{d\eta} + H \cdot \frac{d}{d\eta} \left( \frac{dH}{d\eta} \right) \]
(5.11)
Since we are looking at the origin when \(\eta = 0\), by simplifying (5.11) we have:
\[0 = H \frac{d^2H}{d\eta^2} + \left( \frac{dH}{d\eta} \right)^2 \]
(5.12)
\[\text{Theorem 1. There exists an inflection point in } H(\eta).\]

\[\text{Remark. Let } \eta = L \rightarrow \infty, \text{ which } L \text{ is the so-called penetration distance of the } H \text{ [12].}\]

\[\text{Proof. When } \eta = L \rightarrow \infty, H(\eta) \text{ approaches the horizontal asymptote } H(L) = 1, \text{ given by its boundary (5.10) and (5.12). At the origin the second derivate of } H(\eta) \text{ is } -\left( \frac{dH}{d\eta} \right)^2 / H < 0. \text{ Therefore, to allow } H(\eta) \rightarrow 1 \text{ when } \eta \rightarrow \infty, \text{ an inflection point must exist.} \]

Let \(\eta_k\) be the inflection point, where the second derivative of \(H(\eta)\) is zero. Thus, from (5.8) at the inflection point \(\eta_k\) it follows
\[-\frac{\eta}{2} \frac{dH}{d\eta} = H \frac{d^2H}{d\eta^2} + \left( \frac{dH}{d\eta} \right)^2 = 0 + \left( \frac{dH}{d\eta} \right)^2 \]
(5.13)
or
\[-\frac{\eta_k}{2} = \frac{dH}{d\eta} \bigg|_{\eta=\eta_k} \]
(5.14)
Thus, let \(H(\eta_k) = v\) and the main characteristics of \(H(\eta)\) are
\[H(\eta_k) = v - \frac{\eta_k}{2} = \frac{dH}{d\eta} \bigg|_{\eta=\eta_k} 0 = \frac{d^2H}{d\eta^2} \bigg|_{\eta=\eta_k} H(0) = \mu \quad H(L) = 1 \]
(5.15)
Then, in the interval \(\eta_k \leq \eta \leq L\), a way to express \(H(\eta)\) by interpolating \(H\) approximately as a polynomial. We need to fit this polynomial respects characteristics above. Thus, we have the form
\[H = v - \frac{\eta_k}{2} (\eta - \eta_k) + A(\eta - \eta_k)^\lambda \]
(5.16)
where the unknowns are \(\lambda, \eta_k, v, A, L, \mu\).
5.1.1 The Important Properties of $H(\eta)$

This subsection demonstrates details in solving the ordinary differential equation form of $H$ by find the properties of the unknowns via various algebraic manipulations.

- **Property 1: $A+C=B+C$**

The following figure from previous section (Figure 1) is modified to highlight the area under the postulated solution curve. We extract the first property from this figure.

![Modified postulated piezometric solution curve for $H$.](image)

Integrate (5.8) from $\eta = \eta_k$ to $L \to \infty$ yields

$$
\int_{\eta_k}^{L} \frac{\eta}{2} \frac{dH}{d\eta} \, d\eta = \int_{\eta_k}^{L} \frac{d}{d\eta} \left( H \frac{dH}{d\eta} \right) \, d\eta
$$

(5.17)

Applying integration by parts, the left-hand side of (5.17) yields

$$
\int_{\eta_k}^{L} \frac{\eta}{2} \frac{dH}{d\eta} \, d\eta = -\frac{\eta}{2} \cdot H \bigg|_{\eta_k}^{L} + \frac{1}{2} \int_{\eta_k}^{L} H \, d\eta
$$

(5.18)

$$
= -\frac{L}{2} + \frac{\eta_k \cdot v}{2} + \frac{1}{2} \int_{\eta_k}^{L} H \, d\eta
$$
Similarly, the right-hand side of (5.17) yields
\[ \int_{\eta_k}^{L} \frac{d}{d\eta} \left( H \frac{dH}{d\eta} \right) d\eta = H \frac{dH}{d\eta} \bigg|_{\eta_k}^{L} = 1 \cdot 0 - v \cdot \left( -\frac{\eta_k}{2} \right) = \frac{v\eta_k}{2} \] (5.19)

Thus, putting (5.18) and (5.19) together, it follows
\[ \frac{L \cdot 1}{2} + \frac{v\eta_k}{2} + \frac{1}{2} \int_{\eta_k}^{L} H d\eta = \frac{v\eta_k}{2} \]
\[ \eta_k - L + \int_{\eta_k}^{L} H d\eta = \eta_k \] (5.20)
\[ \int_{\eta_k}^{L} (H - 1) d\eta + \int_{\eta_k}^{L} H d\eta = \eta_k \]
or
\[ \int_{\eta_k}^{L} (H - 1) d\eta = \eta_k \] (5.21)

This equation can be interpreted as the area of A is the same as B from Figure 2. Thus, the area bounded by the H solution curve and the lines H = 1 and \( \eta = \eta_k \) is equal to \( \eta_k \).

**Theorem 2.** In Figure 2, Area(C + B) = Area(C + A)

**Proof.** Based on Calculus knowledge, (5.21) yields,
\[ \int_{\eta_k}^{L} (H - 1) d\eta = \int_{0}^{L} (H - 1) d\eta - \int_{0}^{\eta_k} (H - 1) d\eta \] (5.22)
or
\[ \int_{0}^{L} (H - 1) d\eta = - \int_{0}^{\eta_k} (H - 1) d\eta + \int_{\eta_k}^{L} (H - 1) d\eta \]
\[ = \int_{0}^{\eta_k} (H - 1) d\eta + \eta_k \]
\[ = \int_{0}^{\eta_k} (H - 1) d\eta + \int_{0}^{\eta_k} 1 d\eta \]
\[ = \int_{0}^{\eta_k} H d\eta \] (5.23)

Thus, we have
\[ \int_{0}^{\infty} (H - 1) d\eta = \int_{0}^{\eta_k} H d\eta \] (5.24)

It suggests that the area between the H solution curve and the line H = 1 equals the area bounded by the H curve, the axes H = 0 and \( \eta = 0 \), and the line \( \eta = \eta_k \). Thus, Area(C + B) = Area(C + A)[12].

---

\(^3\)The first derivative of H equals to 0 when \( \eta \to \infty \), for H approaches horizontal asymptote.
Property 2: Two Integral Values at Inflection Points

Then, multiply (5.8) by $\eta$ and integrate it from $\eta = 0$ to $\eta = \eta_k$ yields

\[
\int_0^{\eta_k} \frac{-\eta^2}{2} \frac{dH}{d\eta} d\eta = \int_0^{\eta_k} \eta \frac{d}{d\eta} \left( H \frac{dH}{d\eta} \right) d\eta
\]  
(5.25)

Apply integration by parts on left-hand sides of (5.25), it follows

\[
\int_0^{\eta_k} \frac{-\eta^2}{2} \frac{dH}{d\eta} d\eta = -\frac{\eta^2}{2} \cdot H \bigg|_0^{\eta_k} + \int_0^{\eta_k} \eta H d\eta = -\frac{v\eta_k^2}{2} + \int_0^{\eta_k} \eta H d\eta
\]  
(5.26)

Similarly, right-hand sides it follows

\[
\int_0^{\eta_k} \eta \frac{d}{d\eta} \left( H \frac{dH}{d\eta} \right) d\eta = \left. H \frac{dH}{d\eta} \right|_0^{\eta_k} - \int_0^{\eta_k} \left( H \frac{dH}{d\eta} \right) d\eta
\]  
(5.27)

\[
= -\frac{v\eta_k^2}{2} - \frac{H^2}{2} \bigg|_0^{\eta_k}
\]

\[
= -\frac{v\eta_k^2}{2} - \frac{v^2 - \mu^2}{2}
\]

Then, putting (5.26) and (5.27) together, we have

\[
\int_0^{\eta_k} \eta H d\eta = \frac{\mu^2 - v^2}{2}
\]  
(5.28)

Similarly, multiply (5.8) by $\eta$ and integrate it from $\eta = \eta_k$ to $\eta = \infty$ yields

\[
\int_{\eta_k}^{\infty} -\frac{\eta^2}{2} \frac{dH}{d\eta} d\eta = \int_{\eta_k}^{\infty} \eta \frac{d}{d\eta} \left( H \frac{dH}{d\eta} \right) d\eta
\]  
(5.29)

Apply integration by parts on (5.29), it yields

\[
\int_{\eta_k}^{\infty} \frac{-\eta^2}{2} \frac{dH}{d\eta} d\eta = \int_{\eta_k}^{\infty} \eta \frac{d}{d\eta} \left( H \frac{dH}{d\eta} \right) d\eta
\]

\[
\left( \frac{\eta_k^2}{2} \right) - \frac{L^2}{2} + \int_{\eta_k}^{\infty} \eta H d\eta = -\frac{1 + v^2}{2} + \left( \frac{\eta_k^2}{2} \right)
\]

\[
\int_{\eta_k}^{\infty} -\eta d\eta + \int_{\eta_k}^{\infty} \eta H d\eta = -\frac{1 + v^2 + \eta_k^2}{2}
\]

or

\[
\int_{\eta_k}^{\infty} \eta (H - 1) d\eta = \frac{v^2 - 1 + \eta_k^2}{2}
\]  
(5.31)
5.1.2 Polynomial Approximation Of the Solution

Now, we are ready to find all the unknowns of

\[ H = v - \frac{\eta_k}{2} (\eta - \eta_k) + A(\eta - \eta_k)^\lambda \] (5.32)

We need to determine constants \( A \) and \( \lambda \). Based on all boundary conditions, we can make following boundary conditions based on (5.32)

\[ H(L) = 1 = v - \frac{\eta_k}{2} (L - \eta_k) + A(\eta - \eta_k)^\lambda \] (5.33)

\[ \frac{dH}{d\eta} \bigg|_{\eta=L} = 0 = -\frac{\eta_k}{2} + A \cdot \lambda(L - \eta_k)^{\lambda-1} \] (5.34)

In (5.21), we can substitute \( H(\eta) \) by the expression in (5.32). On the left-hand side of (5.21) we have

\[
\int_0^\infty (H - 1) \, d\eta = \int_0^\infty \left( v - \frac{\eta_k}{2} (\eta - \eta_k) + A(\eta - \eta_k)^\lambda - 1 \right) \, d\eta \\
= \left( -\eta + v\eta - \frac{\eta^2 \eta_k}{4} + \frac{\eta \eta_k^2}{2} + A(\eta - \eta_k)^\lambda \left( \frac{\eta}{1 + \lambda} - \frac{\eta_k}{1 + \lambda} \right) \right) \bigg|_0^\infty \\
= -\eta \bigg|_0^\infty + v\eta \bigg|_0^\infty - \frac{\eta^2 \eta_k}{4} \bigg|_0^\infty + \frac{\eta \eta_k^2}{2} \bigg|_0^\infty + \frac{A(\eta - \eta_k)^{1+\lambda}}{1 + \lambda} \bigg|_0^\infty \\
= -L + vL - \frac{L^2 \eta_k}{4} + \frac{L \eta_k^2}{2} + \frac{A(L - \eta_k)^{1+\lambda}}{1 + \lambda} \\ (5.35)
\]

while on the right-hand side we have

\[
\int_0^{\eta_k} H \, d\eta = \int_0^{\eta_k} \left( v - \frac{\eta_k}{2} (\eta - \eta_k) + A(\eta - \eta_k)^\lambda \right) \, d\eta \\
= \left( v\eta - \frac{\eta^2 \eta_k}{4} + \frac{\eta \eta_k^2}{2} + A(\eta - \eta_k)^\lambda \left( \frac{\eta}{1 + \lambda} - \frac{\eta_k}{1 + \lambda} \right) \right) \bigg|_0^{\eta_k} \\
= v\eta \bigg|_0^{\eta_k} - \frac{\eta^2 \eta_k}{4} \bigg|_0^{\eta_k} + \frac{\eta \eta_k^2}{2} \bigg|_0^{\eta_k} \\
= v\eta_k + \frac{\eta_k^3}{4} \\ (5.36)
\]

So put (5.35) and (5.36) together

\[
(\eta_k) - L + vL - \frac{L^2 \eta_k}{4} + \frac{L \eta_k^2}{2} + \frac{A(L - \eta_k)^{1+\lambda}}{1 + \lambda} = v\eta_k + \frac{\eta_k^3}{4} (+\eta_k) \\
(v - 1) (L - \eta_k) - \frac{\eta_k(L - \eta_k)^2}{4} = \eta_k \\ (5.37)
\]
Similarly, from (5.32), and (5.21),
\[
\int_{\eta_k}^{\infty} \eta (H - 1) \, d\eta = \int_{\eta_k}^{\infty} \eta (H - 1) \, d\eta - \int_{\eta_k}^{\infty} \eta_k (H - 1) \, d\eta + \int_{\eta_k}^{\infty} \eta_k (H - 1) \, d\eta
\]
\[
= \eta_k^2 + \int_{\eta_k}^{\infty} (\eta - \eta_k) (H - 1) \, d\eta
\]
\[
= \eta_k^2 + \int_{\eta_k}^{\infty} (\eta - \eta_k) \left( v - \frac{\eta_k}{2} (\eta - \eta_k) + A (\eta - \eta_k)^3 - 1 \right) \, d\eta
\]
\[
= \eta_k^2 + \left( \frac{A (\eta - \eta_k)^{\lambda + 2}}{\lambda + 2} - \frac{\eta_k^3}{6} + \frac{\eta_k^3}{2} + \frac{\eta_k^2}{2} + \eta \eta_k + \frac{\eta^2 v}{2} - \eta_k v \right) \bigg|_{\eta_k}^{\infty}
\]
\[
= \eta_k^2 + \left( \frac{A (L - \eta_k)^{\lambda + 2}}{\lambda + 2} - \frac{L^3 \eta_k}{6} + \frac{L^2 \eta_k^2}{2} - \frac{L^2}{2} - \frac{L \eta_k^3}{2} + \frac{L^2 v}{2} + \frac{L \eta_k}{2} - L \eta_k v \right)
\]
\[
- \left( - \frac{\eta_k^4}{6} + \frac{\eta_k^2}{2} - \frac{\eta_k^2 v}{2} \right)
\]
\[
= \frac{A (\eta - \eta_k)^{\lambda + 2}}{\lambda + 2} - \frac{1}{6} \eta_k (L - \eta_k)^3 + \frac{1}{2} (v - 1) (L - \eta_k)^2 + \eta_k^2
\]
\[
(5.38)
\]

From (5.31), we can find such equality
\[
\frac{A (\eta - \eta_k)^{\lambda + 2}}{\lambda + 2} - \frac{1}{6} \eta_k (L - \eta_k)^3 + \frac{1}{2} (v - 1) (L - \eta_k)^2 + \eta_k^2 = - \frac{\eta_k^2}{2} + \frac{v^2}{2} - \frac{1}{2}
\]
\[
(5.39)
\]

Also, in the interval of $0 \leq \eta \leq \eta_k$, the area under the $H(\eta)$ is taken approximated as a linear function on interval $[0, \eta_k]$. From the Figure 2, we learn that $H(\eta)$ across points $(0, \mu)$ and $(\eta_k, v)$. Put two points into a linear function, we have
\[
H = \frac{v - \mu}{\eta_k} \eta + \mu
\]
\[
(5.40)
\]

Thus, left hand side of (5.28) can be reduced into
\[
\int_0^{\eta_k} \left( \frac{v - \mu}{\eta_k} \eta^2 + \mu \right) \, d\eta = \eta_k^2 \cdot \frac{2v + \mu}{6}
\]
\[
(5.41)
\]

and consequently (5.28) yields
\[
\eta_k^2 = 3 \frac{\mu^2 - v^2}{\mu + 2v}
\]
\[
(5.42)
\]

Equations (5.33), (5.34), (5.37), (5.39) and (5.28) construct a five parameters systems. The unknowns are $\lambda, \eta_k, v, A, L, \mu$. Then, after some algebraic manipulations, we have
the iterative system with 4 intermediate variables $D_1, D_2, D_3$ and $D_4$, as following

$$D_1 = \frac{8\lambda}{\lambda - 1} - \frac{4\lambda^2}{(\lambda - 1)^2} + \frac{8\lambda^2}{(\lambda - 1)^2} \left( \frac{1}{\lambda(\lambda + 1)} \right)$$

$$D_2 = \frac{v - 1}{\eta_k} = \frac{2}{\sqrt{D_1}}$$

$$D_3 = L - \eta_k = \frac{2\lambda}{\lambda - 1} D_2$$

$$D_4 = D_2 D_3^2 - \frac{D_3^3}{3} + \frac{D_3^3}{\lambda(\lambda + 2)}$$

$$\eta_k = (D_4 - 2D_2)(-1 + D_2^2)^{-1}$$

$$A = \frac{1}{2\lambda D_3^{\lambda - 1}}$$

(5.43)

Up to this point, we have all the parameters worked out and the iteration can be carried out by choosing an initial value for $\lambda > 2$. simplifying it further, we have all the parameters from (5.43)

$$\eta_k = (D_4 - 2D_2)(-1 + D_2^2)^{-1}$$

$$v = \frac{2\eta_k}{\sqrt{D_1}} + 1$$

$$L = \frac{2\lambda}{\lambda - 1} D_2 + \eta_k$$

$$A = \frac{\eta_k}{2\lambda D_3^{\lambda - 1}}$$

(5.44)

Also, from (5.44) we have

$$3\mu^2 - \eta_k^2 \mu - 3v^2 - 2v\eta_k^2 = 0$$

(5.45)

To solve $\mu$, we need to use quadratic formula:

$$\mu = \left( -\eta_k^2 + \sqrt{\eta_k^4 + 12(3v^2 + 2\eta_k^2v)} \right) / 6$$

(5.46)

I use Matlab to assist for solving the system. The Matlab code is listed in Section 8.4. Also, the result is demonstrated in the Section 6.1.
5.2 Prototype Development

In previous work conducted by Emily Furst, she created three prototypes and compared them in different problem sizes: 50\(^3\), 120\(^3\), 190\(^3\), 260\(^3\), 330\(^3\) and 400\(^3\). The results in her paper suggested that MueLu + Belos\(^4\) are a suitable way to update deal.ii, which currently uses ML + AztecOO. When she used Xpetra objects, however, she instantiated Xpetra to Epetra. Thus, in my prototype, I instantiate Xpetra to Tpetra moving a step forward.

Thus, to show the differences, three different prototypes are developed, which each one solves a similar problem generated by Galeri. The mathematics problem represents by Xpetra interface, which builds a(n) either Epetra or Tpetra expressed by the command Xpetra::UseEpetra for Epetra or Xpetra::UseTpetra for Tpetra. In the following code, MueLu is used for instantiating Xpetra objects into Epetra or Tpetra objects.

Listing 1: Problem Generation for Prototypes[9].

```c++
typedef double scalar_type;
typedef int local_ordinal_type;
typedef int global_ordinal_type;
typedef KokkosClassic::DefaultNode::DefaultNodeType node_type;
typedef Tpetra::Map<local_ordinal_type,global_ordinal_type,node_type> driver_map_type;
RCP<const Map> xpetraMap =
  MapFactory::Build(Xpetra::UseEpetra,matrixParameters.GetNumGlobalElements(),
  indexBase, comm);
RCP<GaleriXpetraProblem> Pr = Galeri::Xpetra::BuildProblem<SC, LO, GO, Map,
  CrsMatrixWrap, MultiVector>(matrixParameters.GetMatrixType(), xpetraMap,
  matrixParameters.GetParameterList());
RCP<Matrix> xpetraA = Pr->BuildMatrix();
RCP<crs_matrix_type> A = MueLuUtilities::Op2NonConstEpetraCrs(xpetraA);
const driver_map_type map = MueLuUtilities::Map2EpetraMap(*xpetraMap);
```

Then, first comparison is between MLAztecOOEpetra and MLBelosEpetra. They both use Epetra as its data service, ML as its preconditioner, but first one uses AztecOO as its solver and second one uses Belos as the solver. Then, second comparison is between MLBelosEpetra and MueLuBelosTpetra. The MueLuBelosTpetra is adopted from an example from MueLu packages that uses Tpetra, MueLu as its preconditioner, and Belos. These two comparisons indicate improvements in solving time. Details of comparisons results are in Section 6.2.

\(^{4}\)It means that the program uses MueLu as the preconditioner and Belos as the solver.
5.3 Updating deal.ii

The comparison results from prototypes suggests that \textit{Tpetra+MueLu+Belos} has the best performance in large size problems. Thus, I expect a similar improvement in performance after updating deal.ii.

To begin this updating process, I take three steps. Since \textit{Epetra} is currently used by deal.ii, replacing all the data types and functionalities at once is infeasible. Furthermore, as I mention before, currently, \texttt{deal.i::TrilinosWrappers} is using \textit{AztecOO} as its linear solver and \textit{ML} as its preconditioner, which both are not compatible with \textit{Tpetra}.

Thus, I decide to start with updating the linear solver \textit{AztecOO}, which means by updating \textit{AztecOO} to \textit{Belos}, since \textit{Belos} is compatible with both \textit{Epetra} and \textit{ML}.

Next, after I successfully update the linear solver, I turn to convert \textit{Epetra} objects to \textit{Tpetra} objects.

5.3.1 Updating Solver from AztecOO to Belos

To update the linear solver in \texttt{deal.i::TrilinosWrappers} from \textit{AztecOO} to \textit{Belos}, we need to complete a list of things. It starts with modifying ‘header file’ \texttt{trilinos_solver.h}. While using Trilinos package to solve problem, a \texttt{linearProblem} object must be created first. Also, to create a Belos solver, a new \texttt{SolverManager} object and a new \texttt{SolverFactory} object need to be created. When those are finished, we can fully replace \textit{AztecOO} solver. The new solver can choose one of the \textit{Belos} solver algorithms, i.e. block GMRES, block CG, pseudo-block CG, pseudo-block GMRES. \cite{1}

The following list provides the process of modification on \texttt{trilinos Solver.h}.

1. Using new namespaces and abbreviating some variable names;

\begin{verbatim}
//namespace:
using Teuchos::ParameterList;
using Teuchos::RCP;
using Teuchos::rcp;
using Teuchos::rcpFromRef;

//Type Abbreviations or Shortname
typedef double ST;
typedef Epetra_MultiVector MV;
typedef Epetra_Operator OP;
\end{verbatim}
2. Change `Epetra_LinearProblem` into `Belos::LinearProblem<ST,MV,OP>`;

3. Change `AztecOO` into `Belos::SolverManager<ST,MV,OP>` and `Belos::SolverFactory<ST,MV,OP>`.

4. Change some functions return type from `void` to `double`, to examine residuals.

Next, modify `trilinos_solver.cc`. Before solving the problem, deal.II uses a set of different setup functions, `solve`, which initializes the `LinearProblem`. The following code show modifications on one of thos setup functions.

**Remark (Highlights).** The highlighted codes indicate important modifications.
Finally, we can update the solving function `do_solve`. The following code shows modifications. This completes the modification on the linear solver in `deal.II::TrilinosWrappers` from `AztecOO` to `Belos`.

Listing 7: `do_solve` in `trilinos_solver.cc` after modification

```cpp
void SolverBase::do_solve(const PreconditionBase &preconditioner) {
    factory = rcp(new Belos::SolverFactory<ST,MV,OP>);
    int max_iters = solver_control.max_steps();
    double tol = solver_control.tolerance();

    RCP<ParameterList> solverParams = rcp(new ParameterList());
    solverParams->set("Maximum Iterations", max_iters);
    solverParams->set("Convergence Tolerance", solver_control.tolerance());

    switch (solver_name) {
        case cg:
            //before: solver.SetAztecOption(AZ_solver, AZ_cg);
            newSolver = factory->create("CG", solverParams);
            break;
        case cgs:
            //before: solver.SetAztecOption(AZ_solver, AZ_cgs);
            newSolver = factory->create("Block CG", solverParams);
            break;
        case gmres:
            //before: solver.SetAztecOption(AZ_solver, AZ_gmres);
            //before: solver.SetAztecOption(AZ_kspace, additional_data.gmres_restart_parameter);
            solverParams->set("Maximum Restarts",
                             additional_data.gmres_restart_parameter);
            newSolver = factory->create("GMERS", solverParams);
            break;
        case bicgstab:
            //before: solver.SetAztecOption(AZ_solver, AZ_bicgstab);
            newSolver = factory->create("bicgstab", solverParams);
            break;
        case tfqmr:
            //before: solver.SetAztecOption(AZ_solver, AZ_tfqmr);
            newSolver = factory->create("TFQMR", solverParams);
            break;
        default:
            Assert(false, ExcNotImplemented());
    }

    /*Before:
    if (preconditioner.preconditioner.use_count()!=0){
        ierr = solver.SetPrecOperator(const_cast<Epetra_Operator
          *>(preconditioner.preconditioner.get()));
        AssertThrow (ierr == 0, ExcTrilinosError(ierr));
    }
    */
```

18
```cpp
else
    solver.SetAztecOption(AZ_precond, AZ_none);
*/
if (preconditioner.preconditioner.use_count() != 0) {
    RCP<Epetra_Operator> MLPrec = rcpFromRef(*(const_cast<Epetra_Operator (*)(_const_cast<Epetra_Operator>() & preconditioner.preconditioner.get())));
    RCP<Belos::EpetraPrecOp> RP = rcp(new Belos::EpetraPrecOp(MLPrec));
    linear_problem->setRightPrec(RP);
}
linear_problem->setProblem();
newSolver->setProblem(linear_problem);
/* Before:
   ierr = solver.Iterate (solver_control.max_steps(),
     solver_control.tolerance());
*/
Belos::ReturnType result = newSolver->solve();
switch (ierr) {
  case -1:
    AssertThrow (false, ExcMessage("AztecOO::Iterate error code -1: 
    "option not implemented");
  case -2:
    AssertThrow (false, ExcMessage("AztecOO::Iterate error code -2: 
    "numerical breakdown");
  case -3:
    AssertThrow (false, ExcMessage("AztecOO::Iterate error code -3: 
    "loss of precision");
  case -4:
    AssertThrow (false, ExcMessage("AztecOO::Iterate error code -4: 
    "GMRES Hessenberg ill-conditioned");
  default:
    AssertThrow (ierr >= 0, ExcTrilinosError(ierr));
}
  if (result == Belos::Unconverged)
    AssertThrow(false, ExcMessage("Belos::ReturnType Unconverged!");
  //before: solver_control.check (solver.NumIters(), solver.achievedTol());
  solver_control.check(newSolver->getNumIters(), actTol);
  if (solver_control.last_check() != SolverControl::success)
    AssertThrow(false, SolverControl::NoConvergence(solver_control.last_step(),
      solver_control.last_value()));
```
5.3.2 Converting Epetra to Tpetra

Converting Epetra to Tpetra, we not only need to change data types, but also their corresponding functions. Therefore, I create the Table 2 and Table 3 to provide more details. There are few remarks for the tables.

Remark (Arguments). The tables ignore all function arguments.

Remark (Highlights). The blue functions suggest a direct conversion is missing. The red functions suggests that those are the functions with the same name in different structures.

Remark (Italic Form). The italic form is only using for Epetra::FECrsGraph and Epetra::FECrsMatrix, which only implement in Epetra. During conversions, they are converted into Tpetra::CrsGraph and Tpetra::CrsMatrix respectively.

Remark (Template Parameters for Trilinos). The following table are data types specifications for Tpetra Structure.

Listing 8: Data types for Tpetra Structure

```cpp
typedef double SC;
typedef int LO;
typedef int GO;
typedef KokkosClassic::DefaultNode::DefaultNodeType NT;
```

The following table is conversion table for the data types. [3][4][10]

<table>
<thead>
<tr>
<th>Structure</th>
<th>Epetra Structure</th>
<th>Tpetra Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maps</td>
<td>Epetra::Map</td>
<td>Tpetra::Map&lt;SC, LO, GO&gt;</td>
</tr>
<tr>
<td></td>
<td>Epetra::BlockMap</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Epetra::LocalMap</td>
<td>Tpetra::LocalMap&lt;SC, LO, GO&gt;</td>
</tr>
<tr>
<td>Export</td>
<td>Epetra::Export</td>
<td>Tpetra::Export&lt;SC, LO, GO&gt;</td>
</tr>
<tr>
<td>Import</td>
<td>Epetra::Import</td>
<td>Tpetra::Import&lt;SC, LO, GO&gt;</td>
</tr>
<tr>
<td>Operator</td>
<td>Epetra::Operator</td>
<td>Tpetra::Operator&lt;SC, LO, GO, NT&gt;</td>
</tr>
<tr>
<td>Matrices</td>
<td>Epetra::CrsMatrix</td>
<td>Tpetra::CrsMatrix&lt;SC, LO, GO, NT&gt;</td>
</tr>
<tr>
<td></td>
<td>Epetra::FECrsMatrix</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Epetra::RowMatrix</td>
<td>Tpetra::RowMatrix&lt;SC, LO, GO, NT&gt;</td>
</tr>
<tr>
<td>Graphs</td>
<td>Epetra::CrsGraph</td>
<td>Tpetra::CrsGraph&lt;SC, LO, GO, NT&gt;</td>
</tr>
<tr>
<td></td>
<td>Epetra::FECrsGraph</td>
<td></td>
</tr>
<tr>
<td>Vectors</td>
<td>Epetra::Vector</td>
<td>Tpetra::Vector&lt;SC, LO, GO, NT&gt;</td>
</tr>
<tr>
<td></td>
<td>Epetra::MultiVector</td>
<td>Tpetra::MultiVector&lt;SC, LO, GO, NT&gt;</td>
</tr>
<tr>
<td>MPI</td>
<td>Epetra::Comm</td>
<td>Teuchos::RCP&lt;Teuchos::Comm&lt;int&gt;&gt;</td>
</tr>
<tr>
<td></td>
<td>Epetra::MpiComm</td>
<td>Tpetra::MpiPlatform&lt;int&gt;</td>
</tr>
<tr>
<td></td>
<td>Epetra::SerialComm</td>
<td>Tpetra::SerialPlatform&lt;int&gt;</td>
</tr>
</tbody>
</table>

Table 2: Epetra to Tpetra Data Type Conversion
The following table is the conversion table for functions. [3][4][10]

<table>
<thead>
<tr>
<th>E/Tpetra Structure</th>
<th>Epetra Function</th>
<th>Tpetra Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Map</td>
<td>MinAllGID()</td>
<td>getMaxAllGlobalIndex()</td>
</tr>
<tr>
<td></td>
<td>MaxAllGID()</td>
<td>getMinAllGlobalIndex()</td>
</tr>
<tr>
<td></td>
<td>MinMyGID()</td>
<td>getMaxGlobalIndex()</td>
</tr>
<tr>
<td></td>
<td>MaxMyGID()</td>
<td>getMinGlobalIndex()</td>
</tr>
<tr>
<td></td>
<td>MinLID()</td>
<td>getMaxLocalIndex()</td>
</tr>
<tr>
<td></td>
<td>MaxLID()</td>
<td>getMinLocalIndex()</td>
</tr>
<tr>
<td></td>
<td>NumMyElements()</td>
<td>getNodeNumElements()</td>
</tr>
<tr>
<td></td>
<td>NumGlobalElements()</td>
<td>getGlobalNumElements()</td>
</tr>
<tr>
<td></td>
<td>IndexBase()</td>
<td>getIndexBase()</td>
</tr>
<tr>
<td></td>
<td>DistributedGlobal()</td>
<td>isDistributed()</td>
</tr>
<tr>
<td></td>
<td>LinearMap()</td>
<td>isContiguous()</td>
</tr>
<tr>
<td></td>
<td>MyGlobalElements()</td>
<td>getMyGlobalIndices()</td>
</tr>
<tr>
<td></td>
<td>MyGID()</td>
<td>isNodeGlobalElement()</td>
</tr>
<tr>
<td></td>
<td>MyLID()</td>
<td>isNodeLocalElement()</td>
</tr>
<tr>
<td></td>
<td>LID()</td>
<td>getElement()</td>
</tr>
<tr>
<td></td>
<td>GID()</td>
<td>getGlobalElement()</td>
</tr>
<tr>
<td></td>
<td>IsOneToOne()</td>
<td>isOneToOne()</td>
</tr>
<tr>
<td></td>
<td>UniqueGIDs()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RangeMap()</td>
<td>getRangeMap()</td>
</tr>
<tr>
<td></td>
<td>RowMap()</td>
<td>getRowMap()</td>
</tr>
<tr>
<td></td>
<td>DomainMap()</td>
<td>getDomainMap()</td>
</tr>
<tr>
<td></td>
<td>GlobalLength()</td>
<td>getGlobalLength()</td>
</tr>
<tr>
<td></td>
<td>ColMap()</td>
<td>getColMap()</td>
</tr>
<tr>
<td></td>
<td>graph()</td>
<td>getGraph()</td>
</tr>
<tr>
<td></td>
<td>FillComplete()</td>
<td>fillComplete()</td>
</tr>
<tr>
<td></td>
<td>Filled()</td>
<td>isFillComplete()</td>
</tr>
<tr>
<td></td>
<td>IndexBase()</td>
<td>getIndexBase()</td>
</tr>
<tr>
<td></td>
<td>MaxNumIndices()</td>
<td>getNodeMaxNumRowEntries()</td>
</tr>
<tr>
<td></td>
<td>NumMyRows ()</td>
<td>getNodeNumRows()</td>
</tr>
<tr>
<td></td>
<td>NumMyIndices()</td>
<td>getNumEntriesInLocalRow()</td>
</tr>
<tr>
<td></td>
<td>NumGlobalCols ()</td>
<td>getGlobalNumCols()</td>
</tr>
<tr>
<td></td>
<td>NumGlobalIndices()</td>
<td>getNumEntriesInGlobalRow()</td>
</tr>
<tr>
<td></td>
<td>GlobalMaxNumIndices()</td>
<td>getGlobalMaxNumRowEntries()</td>
</tr>
<tr>
<td></td>
<td>ReplaceGlobalValues()</td>
<td>replaceGlobalValues()</td>
</tr>
<tr>
<td></td>
<td>OperatorRangeMap()</td>
<td>getRangeMap()</td>
</tr>
<tr>
<td></td>
<td>OperatorDomainMap()</td>
<td>getDomainMap()</td>
</tr>
<tr>
<td></td>
<td>MaxNumEntries()</td>
<td>getNodeMaxNumRowEntries()</td>
</tr>
<tr>
<td></td>
<td>NumMyEntries()</td>
<td>getNumEntriesInLocalRow()</td>
</tr>
<tr>
<td></td>
<td>LRID()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GRID()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>InsertGlobalValues()</td>
<td>insertGlobalValues()</td>
</tr>
<tr>
<td></td>
<td>GlobalAssemble()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PutScalar()</td>
<td>putScalar()</td>
</tr>
<tr>
<td></td>
<td>Map()</td>
<td>getMap()</td>
</tr>
<tr>
<td></td>
<td>Norm1()</td>
<td>norm1()</td>
</tr>
<tr>
<td></td>
<td>Norm2()</td>
<td>norm2()</td>
</tr>
<tr>
<td></td>
<td>MeanValue()</td>
<td>meanValue()</td>
</tr>
<tr>
<td></td>
<td>Update()</td>
<td>update()</td>
</tr>
</tbody>
</table>

21
<table>
<thead>
<tr>
<th>PutScalar()</th>
<th>putScalar()</th>
</tr>
</thead>
<tbody>
<tr>
<td>MinValue()</td>
<td>putScalar()</td>
</tr>
<tr>
<td>MaxValue()</td>
<td>putScalar()</td>
</tr>
<tr>
<td>ExtractView()</td>
<td>putScalar()</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Operator</th>
<th>OperatorRangeMap()</th>
<th>getRangeMap()</th>
</tr>
</thead>
<tbody>
<tr>
<td>OperatorDomainMap()</td>
<td>getDomainMap()</td>
<td></td>
</tr>
<tr>
<td>Apply()</td>
<td>Apply()</td>
<td></td>
</tr>
<tr>
<td>ApplyInverse()</td>
<td>Apply()</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Solver</th>
<th>reset()</th>
<th>setProblem()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iterate()</td>
<td>solve()</td>
<td></td>
</tr>
<tr>
<td>NumIters()</td>
<td>getNumIters()</td>
<td></td>
</tr>
<tr>
<td>TrueResidual()</td>
<td>achievedTol()</td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Epetra to Tpetra Functions Conversion
6 Results

6.1 One-Dimensional Boussinesq Equation Result

In section 5.1 we have all the parameters for $H(\eta)$ which defined in (5.32) as

$$H(\eta) = v - \frac{\eta_k}{2}(\eta - \eta_k) + A(\eta - \eta_k)^\lambda,$$

where $\eta = \frac{x}{(Kh_0/S)^{1/2}}$ \hspace{1cm} (6.1)

Then we need to find all the parameters by iterations. Thus, iterated through (5.44), we have the following table:\footnote{The code for iterations is listed in Section 8.4}

<table>
<thead>
<tr>
<th>$H$</th>
<th>$\lambda$</th>
<th>$v$</th>
<th>$\eta_k$</th>
<th>$\mu$</th>
<th>$A$</th>
<th>$L$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$H_1$</td>
<td>2.1000</td>
<td>1.2571</td>
<td>0.2923</td>
<td>1.2996</td>
<td>0.0184</td>
<td>3.6498</td>
</tr>
<tr>
<td>$H_2$</td>
<td>2.2000</td>
<td>1.5283</td>
<td>0.5931</td>
<td>1.7010</td>
<td>0.0326</td>
<td>3.8590</td>
</tr>
<tr>
<td>$H_3$</td>
<td>2.3000</td>
<td>1.8135</td>
<td>0.9033</td>
<td>2.2082</td>
<td>0.0435</td>
<td>4.0898</td>
</tr>
<tr>
<td>$H_4$</td>
<td>2.4000</td>
<td>2.1127</td>
<td>1.2240</td>
<td>2.8258</td>
<td>0.0519</td>
<td>4.3408</td>
</tr>
</tbody>
</table>

Table 4: Values of the Parameters for $1 < \mu < 3$.\cite{12}

Then plot the $H(\eta)$ by those parameters in Table 4 we have

![Figure 3: Piezometric $H$ versus $\eta$ for various values of $\mu$.\cite{12}](image)

The following Figure 4, 5 and 6 are representing numerical results for all the basis parameters $v, \eta_k, \lambda, A, and L$ versus $\mu$, $1 < \mu < 3$.\footnote{The code for iterations is listed in Section 8.4}
Figure 4: Values of the parameters $v$ and $\eta_k$ versus $\mu$. [12]

Figure 5: Values of the parameters $\lambda$ and $L$ versus $\mu$. [12]
6.2 Prototypes Comparison

6.2.1 Number of Iterations Comparison

Figure 6: Values of the parameters $\lambda$ and $\Lambda$ versus $\mu$. [12]

Figure 7: A line chart of all trials for MLAztecOOEpetra.cc, MLBelosEpetra.cc, and MueLuBelosTpetra.cc with problem sizes from $50^3$ to $400^3$. This plot shows the range in number of iterations for each file.
6.2.2 Solving Time Comparison

Figure 8: A column chart of 4 time trials for \texttt{MLAztecOOEpetra.cc}, \texttt{ML-BelosEpetra.cc}, and \texttt{MueLuBelosTpetra.cc} with a problem size of $50^3$.

Figure 9: A column chart of 4 time trials for \texttt{MLAztecOOEpetra.cc}, \texttt{ML-BelosEpetra.cc}, and \texttt{MueLuBelosTpetra.cc} with a problem size of $200^3$. 
Figure 10: A column chart of 4 time trials for *MLAztecOOEpetra.cc*, *MLBelosEpetra.cc*, and *MueLuBelosTpetra.cc* with a problem size of $350^3$.

Figure 11: A column chart of all trials for *MLAztecOOEpetra.cc*, *MLBelosEpetra.cc*, and *MueLuBelosTpetra.cc* with a problem size from $50^3$ to $400^3$.
6.3 Comparison of Trilinos Linear Solvers in deal.ii

6.3.1 Solving time Comparison

Figure 12: A scatter chart at Timestep 0 for step-32.cc. The measured time is solving time. This plot shows the range in time performance for Belos and AztecOO solver. The solving results listed at the bottom of this chart have the same value for both solvers.

Figure 13: A scatter chart at Timestep 21 for step-32.cc. The measured time is solving time. This plot shows the range in time performance for Belos and AztecOO solver. The solving results listed at the bottom of this chart have the same value for both solvers.
Figure 14: A scatter chart at Timestep 51 for *step-32.cc*. The measured time is solving time. This plot shows the range in time performance for *Belos* and *AztecOO* solver. The solving results listed at the bottom of this chart have the same value for both solvers.

Figure 15: A scatter chart at Timestep 101 for *step-32.cc*. The measured time is solving time. This plot shows the range in time performance for *Belos* and *AztecOO* solver. The solving results listed at the bottom of this chart have the same value for both solvers.
### Number of Solver Iteration Comparison

#### Table of Number of Iterations at Timestep 0, 21, 51, 101

<table>
<thead>
<tr>
<th>Timestep</th>
<th>0</th>
<th>21</th>
<th>51</th>
<th>101</th>
<th>0</th>
<th>21</th>
<th>51</th>
<th>101</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calling</td>
<td>B</td>
<td>A</td>
<td>B</td>
<td>A</td>
<td>B</td>
<td>A</td>
<td>B</td>
<td>A</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>10</td>
<td>15</td>
<td>15</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>10</td>
<td>15</td>
<td>15</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>10</td>
<td>15</td>
<td>15</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>10</td>
<td>15</td>
<td>15</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>10</td>
<td>15</td>
<td>15</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>10</td>
<td>15</td>
<td>15</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>10</td>
<td>15</td>
<td>15</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>8</td>
<td>10</td>
<td>10</td>
<td>15</td>
<td>15</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>9</td>
<td>10</td>
<td>10</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>17</td>
<td>17</td>
<td>53</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>10</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>17</td>
<td>17</td>
<td>54</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
<td>10</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>17</td>
<td>17</td>
<td>54</td>
</tr>
<tr>
<td>12</td>
<td>10</td>
<td>10</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>17</td>
<td>17</td>
<td>54</td>
</tr>
<tr>
<td>13</td>
<td>10</td>
<td>10</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>17</td>
<td>17</td>
<td>54</td>
</tr>
<tr>
<td>14</td>
<td>10</td>
<td>10</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>17</td>
<td>17</td>
<td>54</td>
</tr>
<tr>
<td>15</td>
<td>10</td>
<td>10</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>17</td>
<td>17</td>
<td>54</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>10</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>17</td>
<td>17</td>
<td>54</td>
</tr>
<tr>
<td>17</td>
<td>8</td>
<td>8</td>
<td>15</td>
<td>15</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>18</td>
<td>8</td>
<td>8</td>
<td>15</td>
<td>15</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>19</td>
<td>8</td>
<td>8</td>
<td>15</td>
<td>15</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>20</td>
<td>8</td>
<td>8</td>
<td>15</td>
<td>15</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>21</td>
<td>9</td>
<td>9</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>22</td>
<td>9</td>
<td>9</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>23</td>
<td>9</td>
<td>9</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>24</td>
<td>9</td>
<td>9</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>25</td>
<td>10</td>
<td>10</td>
<td>15</td>
<td>15</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>26</td>
<td>10</td>
<td>10</td>
<td>15</td>
<td>15</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>27</td>
<td>10</td>
<td>10</td>
<td>15</td>
<td>15</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>28</td>
<td>10</td>
<td>10</td>
<td>15</td>
<td>15</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>29</td>
<td>10</td>
<td>10</td>
<td>15</td>
<td>15</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>30</td>
<td>10</td>
<td>10</td>
<td>15</td>
<td>15</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>31</td>
<td>10</td>
<td>10</td>
<td>15</td>
<td>15</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>32</td>
<td>10</td>
<td>10</td>
<td>15</td>
<td>15</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>33</td>
<td>9</td>
<td>9</td>
<td>13</td>
<td>13</td>
<td>10</td>
<td>10</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>34</td>
<td>9</td>
<td>9</td>
<td>13</td>
<td>13</td>
<td>10</td>
<td>10</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>35</td>
<td>9</td>
<td>9</td>
<td>13</td>
<td>13</td>
<td>10</td>
<td>10</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>36</td>
<td>9</td>
<td>9</td>
<td>13</td>
<td>13</td>
<td>10</td>
<td>10</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>37</td>
<td>9</td>
<td>9</td>
<td>13</td>
<td>13</td>
<td>10</td>
<td>10</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>38</td>
<td>9</td>
<td>9</td>
<td>13</td>
<td>13</td>
<td>10</td>
<td>10</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>39</td>
<td>9</td>
<td>9</td>
<td>13</td>
<td>13</td>
<td>10</td>
<td>10</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>40</td>
<td>9</td>
<td>9</td>
<td>13</td>
<td>13</td>
<td>10</td>
<td>10</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>41</td>
<td>9</td>
<td>9</td>
<td>10</td>
<td>10</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>42</td>
<td>9</td>
<td>9</td>
<td>10</td>
<td>10</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>43</td>
<td>9</td>
<td>9</td>
<td>10</td>
<td>10</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>44</td>
<td>9</td>
<td>9</td>
<td>10</td>
<td>10</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
</tr>
</tbody>
</table>

Figure 16: A table lists the number of iterations where ‘B’ stands for ‘Belos’ and ‘A’ stands for ‘AztecOO’. ‘Calling’ represents the $n^{th}$ time of calling $B.jlockSchurPreconditioner::vmult$ function in `step-32.cc`. 
# Conclusion

Overall, this project demonstrates the analytical solution of an one-dimensional Boussinesq equation, a workable prototype for updating deal.ii, a new solver for deal.ii and a conversion table for converting Epetra objects to Tpetra objects.

## 7.1 Performance of Prototypes

The results from Section 6.2 confirms that using MueLu as the preconditioner, Belos as solver and Tpetra as basic mathematical structure result in shorter solving time and a smaller number of iterations. A real world problem that deals.ii solves can potentially have the large size of matrices and vectors. Thus, I believe the results from prototypes suggest that improvement can be expected after successfully updating deal.ii as proposed.

Noticeably, the intermediate prototype, $MLEpetraBelos$, does not result in faster solving time with smaller sizes of the problems. Thus, the same results apply to the current updated solver version of deal.ii-8.4.1. Since I have only updated the solver, and ‘step-32’ uses ML as preconditioner, the program result in a longer runtime with such a combination.

## 7.2 Performance of Updated deal.ii

The results from Section 6.3 reflect Section 6.2, that ML as preconditioner, Belos as solver with Epetra do not produce shorter solving time and a smaller number of iterations. Actually, the table in Figure 16 suggests that there is no difference in the number of iterations at Timestep 0,21,51, and 101. Since this version of deal.ii only updated its Trilinos solver, based on the results from 6.3, I believe that improvement can be expected after a complete update.

## 7.3 Future Work

Currently, this project finishes updating the deal.ii Trilinos solver. The next phase of converting Epetra to Tpetra is still ongoing. Also, the conversion table is not entirely complete, which means that there is still much to do.

For starters, the conversion tables are not yet completed, maintaining the conversion table is critical for the future of this project. The table needs regular updates since Trilinos packages are constantly renewed. Also, it may be possible to encapsulate some Xpetra objects into the project because Xpetra objects are more friendly when working with MueLu.
8 Appendix

8.1 Source Code

8.1.1 Source Code for Prototype

The complete source code of the prototypes can be found at the git repository https://github.com/s1lin/ug_thesis_prototypes or clone as https://github.com/s1lin/ug_thesis_prototypes.git.

8.1.2 Source Code for Updated deal.ii

The source code of the ongoing updating process of deal.ii-8.4.1 can be found at the git repository https://github.com/s1lin/deal.ii-8.4.1-Tpetra.git or clone as https://github.com/s1lin/deal.ii-8.4.1-Tpetra.git. The complete source code for deal.ii with ‘Belos’ solver can be checked out as a branch.

8.2 Computing Specifications

In this project, the computing node has a Dual Intel Xeon CPU E5-2420 Sandy Bridge @ 1.90GHz 15MB L3 Cache 95W Six-Core (12 cores per node, with 2 HW thread states per core). The following are the detailed specifications.

Listing 9: Specification

```plaintext
vendor_id : GenuineIntel
cpu family : 6
model : 45
model name : Intel(R) Xeon(R) CPU E5-2420 @ 1.90GHz
stepping : 7
microcode : 0x710
cpu : 0
cache size : 15360 KB
physical id : 1
siblings : 12
core id : 5
cpu cores : 6
apicid : 43
initial apicid : 43
fpu : yes
fpu_exception : yes
cpuid level : 13
wp : yes
flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat
        pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb
        rdtscp lm constant_tsc arch_perfmon pebs bts rep_good nopl xtopology
        nonstop_tsc aperfmperf eagerfpu pni pclmulqdq dtes64 monitor ds_cpl vmx
        smx est tm2 ssse3 cx16 xptr pdcm pcid dca sse4_1 sse4_2 x2apic popcnt
        tsc_deadline_timer aes xsave avx lahf_lm ida arat epb pln pts dtherm
        tpr_shadow vnmi flexpriority ept vpid xsaveopt
```
### 8.3 Prototypes Detailed Comparison Table

**Remark.** In following table ‘I’ stands for number of iterations, ‘size’ stands for the problem size, and ‘Residual’ stands for the result residual. The time is measuring in seconds. The actual problem size is the cube power to the number in the ‘size’ column.

<table>
<thead>
<tr>
<th>Size</th>
<th>MueLuBelosEpetra</th>
<th>MLBelosEpetra</th>
<th>MLArzeOOEpetra</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time</td>
<td>I Residual</td>
<td>Time</td>
</tr>
<tr>
<td>50</td>
<td>0.13832</td>
<td>13 1.12E-07</td>
<td>0.07621</td>
</tr>
<tr>
<td>50</td>
<td>0.12876</td>
<td>13 1.12E-07</td>
<td>0.05581</td>
</tr>
<tr>
<td>50</td>
<td>0.13895</td>
<td>13 1.12E-07</td>
<td>0.05359</td>
</tr>
<tr>
<td>50</td>
<td>0.12423</td>
<td>13 1.12E-07</td>
<td>0.05636</td>
</tr>
<tr>
<td>100</td>
<td>0.66452</td>
<td>13 1.17E-05</td>
<td>1.27561</td>
</tr>
<tr>
<td>100</td>
<td>0.57198</td>
<td>13 1.17E-05</td>
<td>0.96262</td>
</tr>
<tr>
<td>100</td>
<td>0.66596</td>
<td>13 1.17E-05</td>
<td>0.95775</td>
</tr>
<tr>
<td>100</td>
<td>0.55718</td>
<td>13 1.17E-05</td>
<td>0.96385</td>
</tr>
<tr>
<td>150</td>
<td>2.03450</td>
<td>13 5.53E-05</td>
<td>4.41518</td>
</tr>
<tr>
<td>150</td>
<td>2.00938</td>
<td>13 5.53E-05</td>
<td>4.41267</td>
</tr>
<tr>
<td>150</td>
<td>2.01293</td>
<td>13 5.53E-05</td>
<td>4.41944</td>
</tr>
<tr>
<td>150</td>
<td>2.02474</td>
<td>13 5.53E-05</td>
<td>4.09682</td>
</tr>
<tr>
<td>200</td>
<td>5.30855</td>
<td>14 1.81E-03</td>
<td>14.0855</td>
</tr>
<tr>
<td>200</td>
<td>5.30083</td>
<td>14 1.81E-03</td>
<td>14.0809</td>
</tr>
<tr>
<td>200</td>
<td>5.29882</td>
<td>14 1.81E-03</td>
<td>14.0751</td>
</tr>
<tr>
<td>200</td>
<td>5.28961</td>
<td>14 1.81E-03</td>
<td>14.0423</td>
</tr>
<tr>
<td>250</td>
<td>12.0674</td>
<td>16 4.28E-03</td>
<td>37.7675</td>
</tr>
<tr>
<td>250</td>
<td>12.0437</td>
<td>16 4.28E-03</td>
<td>37.6013</td>
</tr>
<tr>
<td>250</td>
<td>12.0514</td>
<td>16 4.28E-03</td>
<td>37.5571</td>
</tr>
<tr>
<td>250</td>
<td>12.0406</td>
<td>16 4.28E-03</td>
<td>37.5610</td>
</tr>
<tr>
<td>300</td>
<td>20.7422</td>
<td>16 4.28E-03</td>
<td>82.4681</td>
</tr>
<tr>
<td>300</td>
<td>20.7682</td>
<td>16 8.02E-03</td>
<td>85.9746</td>
</tr>
<tr>
<td>300</td>
<td>20.7546</td>
<td>16 8.02E-03</td>
<td>87.8332</td>
</tr>
<tr>
<td>300</td>
<td>20.7720</td>
<td>16 8.02E-03</td>
<td>89.5426</td>
</tr>
<tr>
<td>350</td>
<td>33.0793</td>
<td>16 2.52E-02</td>
<td>177.653</td>
</tr>
<tr>
<td>350</td>
<td>32.8929</td>
<td>16 2.52E-02</td>
<td>173.226</td>
</tr>
<tr>
<td>350</td>
<td>33.1230</td>
<td>16 2.52E-02</td>
<td>169.977</td>
</tr>
<tr>
<td>350</td>
<td>33.4689</td>
<td>16 2.52E-02</td>
<td>161.084</td>
</tr>
<tr>
<td>400</td>
<td>50.0645</td>
<td>16 6.24E-02</td>
<td>429.628</td>
</tr>
<tr>
<td>400</td>
<td>49.9835</td>
<td>16 6.24E-02</td>
<td>324.990</td>
</tr>
<tr>
<td>400</td>
<td>49.6237</td>
<td>16 6.24E-02</td>
<td>303.448</td>
</tr>
<tr>
<td>400</td>
<td>49.4960</td>
<td>16 6.24E-02</td>
<td>291.795</td>
</tr>
</tbody>
</table>

Table 5: Prototypes Detailed Comparison Table
8.4 Code for Iteration

Listing 10: Iteration.m

```matlab
function [u,v,nk,l,L] = Solve()

l(1) = 2;
m = 2;
index = 1;
tol = 1e-6;
while (m<3)
    l(index) = m;
    D1 = 8*m/(m-1)-4*m*m/(m-1)^2+8*m*m/(m-1)^2*(1/(m*(m+1)));
    D3 = 2/sqrt(D1);
    D4 = D3*D2^2-D4^3/3+D4^3/(m*(m+2));
    nk(index) = (D2 - 2*D3)*(-1+D3^2)^(-1);
    v(index) = D3*nk(index)+1;
    L(index) = D3*2*m/(m-1) + nk(index);
    A(index) = nk(index)/(2*m*(L(index)-nk(index))^(m-1));
    b = -nk(index)^2;
    delta = b^2-12*(-3*v(index)^2-2*v(index)*nk(index)^2);
    u(index) = (-b + sqrt(delta))/6;
    m = l(index) + 1e-8;
    index = index + 1;
end
end
```

8.5 Installation Notes and Scripts

8.5.1 The Trilinos Packages

The following is the CMake script to install Trilinos. The latest Trilinos version is 12.10, which can be downloaded from the public Github repository.

Listing 11: Trilinos Installation and CMake Script

```bash
mkdir trilinos_build
cd trilinos_build

cmake
-DTrilinos_ENABLE_ALL_OPTIONAL_PACKAGES=OFF
-DTrilinos_ENABLE_Amesos=ON
-DTrilinos_ENABLE_Epetra=ON
-DTrilinos_ENABLE_Tpetra=ON
-DTrilinos_ENABLE_Xpetra=ON
-DTrilinos_ENABLE_EpetraExt=ON
```

---

6To successfully install Trilinos with MPI gFortran is required. The installation guide is listed in Section 8.5.3.
7To get Trilinos, clone from https://github.com/trilinos/Trilinos.git
8.5.2 The deal.ii Library

The following is a CMake script to install the deal.ii, version 8.4.1.

Listing 12: deal.ii Installation and CMake Script

```
mkdir dealii_build

cd dealii_build

make
```

8.5.3 Project Required Libraries

1. gFortran

---

8Installation of Trilinos are required in reproducing this project and should be installed prior to deal.ii installation.
2. **BLAS, LAPACK** \(^9\)

3. **P4EST** \(^{10}\)

---

\(^{9}\)BLAS: Basic Linear Algebra Subprograms and LAPACK: Linear Algebra PACKage

\(^{10}\)The P4EST enables the dynamic management of a collection of adaptive octrees, conveniently called a forest of octrees. [http://www.p4est.org](http://www.p4est.org)


References


