Math 226C Project 3

Theodore Kwan
tmkwan@uci.edu

June 14, 2016

Contents

1 Project Description

This project concerns the system of partial differential equations:

$$\begin{cases} \nabla \times (\nabla \times u) = f(\mathbf{x}), & \mathbf{x} \in \Omega \\ \operatorname{div}(u) = 0, & \mathbf{x} \in \Omega \\ u = g_D(\mathbf{x}), & \mathbf{x} \in \partial\Omega \end{cases} \tag{1}$$

Where:

$$\Omega = (0,1) \times (0,1) \times (-1,0) \tag{2}$$

The solution to this system can be approximated by the finite element method applied to edge elements.

1.1 Problems Analyzed

1.1.1 Problem 1

The first problem that will by analyzed has exact solution:

$$u = \begin{pmatrix} \sin(z) \\ \cos(x) \\ \sin(y) \end{pmatrix} \tag{3}$$

To find the right hand side, the curl needs to be calculated. So:

$$\nabla \times u = \begin{vmatrix} i & j & k \\ \partial_x & \partial_y & \partial_z \\ \sin(z) & \cos(x) & \sin(y) \end{vmatrix} = \begin{pmatrix} \cos(y) \\ \cos(z) \\ -\sin(x) \end{pmatrix} \tag{4}$$

And:

$$\nabla \times (\nabla \times u) = \begin{vmatrix} i & j & k \\ \partial_x & \partial_y & \partial_z \\ \cos(y) & \cos(z) & -\sin(x) \end{vmatrix} = \begin{pmatrix} \sin(z) \\ \cos(x) \\ \sin(y) \end{pmatrix} \tag{5}$$

1.1.2 Problem 2

The second problem that will by analyzed has exact solution:

$$u = \begin{pmatrix} z^2 \\ x^2 \\ y^2 \end{pmatrix} \tag{6}$$

To find the right hand side, the curl needs to be calculated. So:

$$\nabla \times u = \begin{vmatrix} i & j & k \\ \partial_x & \partial_y & \partial_z \\ z^2 & x^2 & y^2 \end{vmatrix} = \begin{pmatrix} 2y \\ -(-2z) \\ 2x \end{pmatrix} = \begin{pmatrix} 2y \\ 2z \\ 2x \end{pmatrix} \tag{7}$$

And:

$$\nabla \times (\nabla \times u) = \begin{vmatrix} i & j & k \\ \partial_x & \partial_y & \partial_z \\ 2y & 2z & 2x \end{vmatrix} = \begin{pmatrix} -2 \\ -2 \\ -2 \end{pmatrix} \tag{8}$$

2 Direct Method

This section describes and analyzes the direct method for solving the discrete system of partial differential equations (1). After discretization, the resulting system of equations is solved using the built-in mldivide method in Matlab.

2.1 Basis and Matrix System

The system of partial differential equations (1) is being approximated on the edges instead of the nodes to maintain stability. The basis functions used to calculate the stiffness and divergence matrices are given by:

$$\phi_k = \lambda_i \nabla \lambda_j - \lambda_j \nabla \lambda_i \tag{9}$$

Using the basis given in (9), we can calculate the stiffness matrix and the negative divergence matrix:

$$A_{ij} = (\nabla \times \phi_i, \nabla \times \phi_j), \quad B_{ij} = (\phi_i, \nabla \lambda_j) \tag{10}$$

Using these two matrices, we can convert the original discretization of equation (1) to the saddle-point system:

$$\begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix} \tag{11}$$

Where p is a lagrange-multiplier, and $g = \nabla \cdot u$. Using this saddle-point system, the PDE is approximated and the finite element approximation u_h is found.

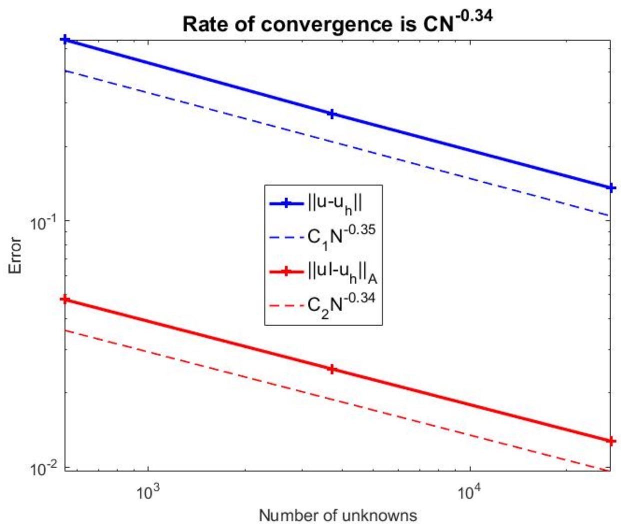
2.2 Convergence Rates with Direct Solver

We can plot the convergence rates for u and $\nabla \cdot p$, as well as that of $\nabla \cdot u$. This is done in the sections below.

2.2.1 Convergence of u

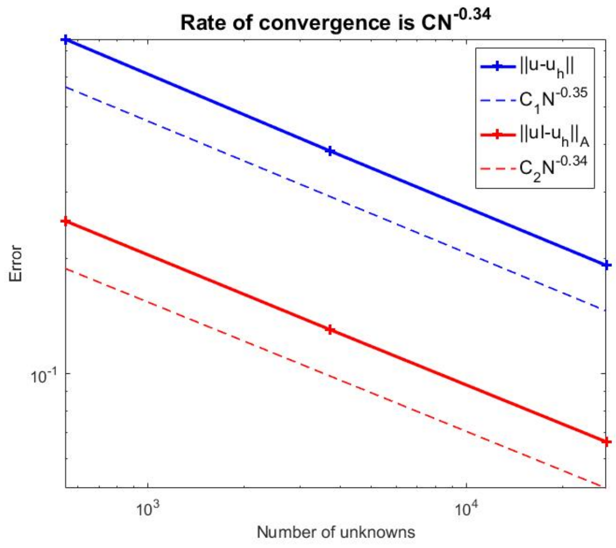
This section plots the convergence of u in the L_2 and H_1 norms. Both problems are recorded, and show the same rates of convergence.

For the first problem, the approximation for u converges at a rate of $N^{-0.35}$ in the L_2 norm, and at a rate of $N^{-0.34}$ in the H_1 norm. This is shown in the image below:



This is the expected convergence rate for this method using the direct solver. The problem is that any further refinements are too large to use mldivide since it will cause an out of memory error.

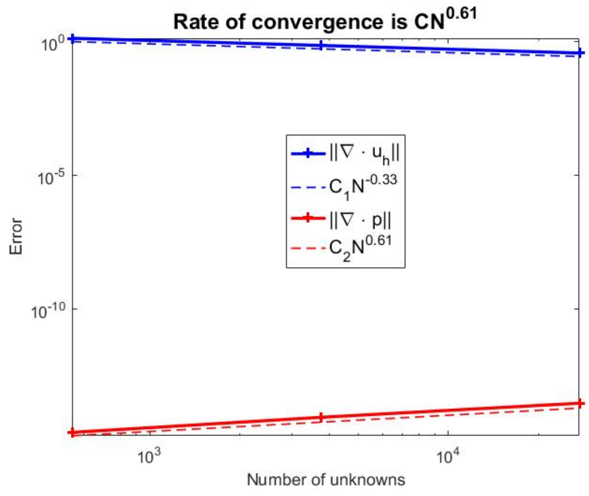
For the second problem, the convergence rates are exactly the same. This is shown in the image below:



This is the expected convergence rate for the finite element method applied to the saddle-point system using the direct solver. The problem is that any further refinements are too large to use a direct solver since it will cause an out of memory issue.

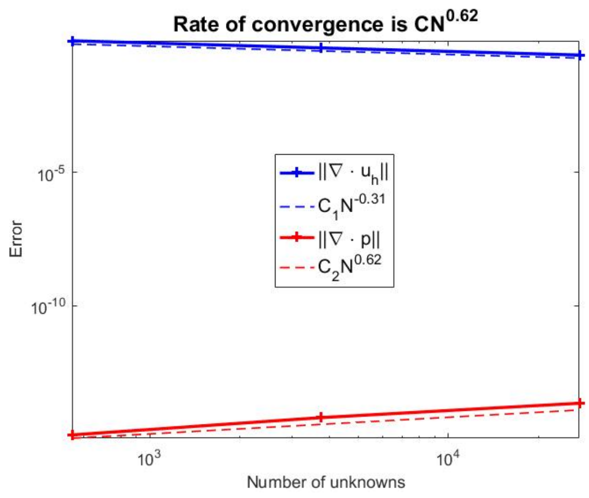
2.2.2 Convergence of p

The rates of convergence for $\nabla \cdot u$ are correct, but the convergence rate for $\nabla \cdot p$ is incorrect. This is shown in the image below:



For the second problem, the convergence rates are similar, and $\nabla \cdot p$ still diverges.

This is shown in the image below:



3 Multigrid Method

For the multi-grid method, the system has been changed to a saddle-point system which uses the Laplacian matrix while modifying the right hand side. This correction is due to the fact that:

$$-\Delta u = \nabla \times (\nabla \times u) - \nabla(\nabla \cdot u) \tag{12}$$

Using (12), we can modify the original system (11) to:

$$\begin{pmatrix} \tilde{A} & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} \tilde{f} \\ g \end{pmatrix} \tag{13}$$

Where p is a Lagrange-multiplier, and $g = -\nabla \cdot u$. In addition, the matrix $\tilde{A}$ is the discrete Laplacian matrix, and the right hand side is given by:

$$\tilde{f} = f - \nabla(\nabla \cdot u) \tag{14}$$

3.1 Distributed Gauss-Seidel Algorithm

The Distributed Gauss-Seidel method is applied to the saddle point system given above by the algorithm described in project Stokes. The algorithm is given by:

Given the current approximations $\mathbf{u}^k,\ p^k$:

$$\left[\mathbf{u}^{k+1}, p^{k+1}\right] = DGS\left(\mathbf{u}^k, p^k\right) \tag{15}$$

1. Relax the momentum equations to find the intermediate velocity field by solving the poisson equation:

$$\begin{cases} \Delta \mathbf{u}^{k+\frac{1}{2}} = \nabla p^k - f(\mathbf{x}), & \mathbf{x} \in \Omega \\ \mathbf{u}^{k+\frac{1}{2}} = g(\mathbf{x}) & \mathbf{x} \in \partial\Omega \end{cases}$$

2. Relax the residual of the continuity equation by solving the poisson equation:

$$\begin{cases} \Delta \phi = \nabla \cdot \mathbf{u}^{k+\frac{1}{2}} - g(\mathbf{x}), & (\mathbf{x}) \in \Omega \\ \frac{\partial \phi}{\partial \mathbf{n}} = 0, & (\mathbf{x}) \in \partial\Omega \end{cases}$$

3. Update the next step with the correction:

$$\begin{aligned} \mathbf{u}^{k+1} &= \mathbf{u}^{k+\frac{1}{2}} + \nabla \phi \\ p^{k+1} &= p^k - \Delta \phi \end{aligned}$$

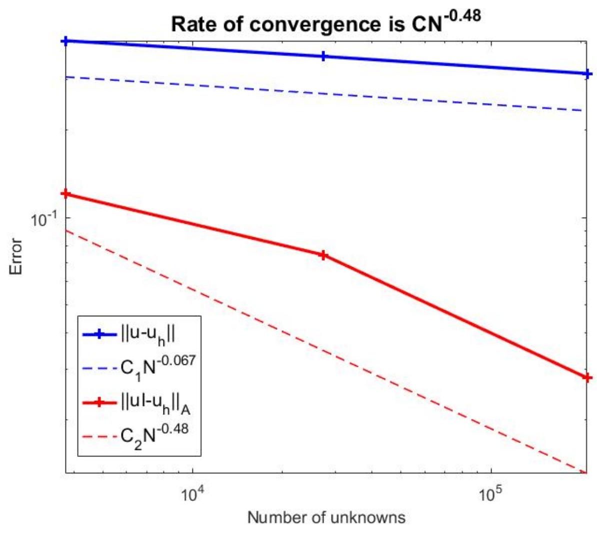
3.2 Convergence Rates with DGS and Multi-Grid

We can plot the convergence rates for u and $\nabla \cdot p$, as well as that of $\nabla \cdot u$. This is done in the sections below.

3.2.1 Convergence of u

This section plots the convergence of u in the L_2 and H_1 norms. Convergence of both methods is measured on the first problem (3), and show almost the same rates of convergence. It appears as if the multi-grid method that was tested does not work properly yet, since it does not actually reduce the number of iterations.

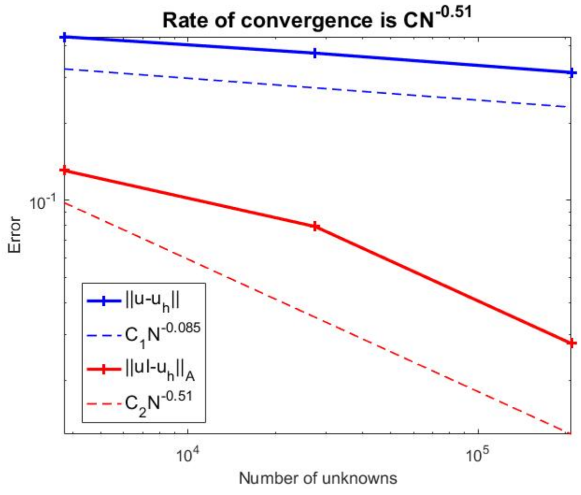
For the first problem using the multi-grid method, the approximation to u converges at a rate of $N^{-0.067}$ in the L_2 norm, and at a rate of $N^{-0.48}$ in the H_1 norm. This is shown in the image below:



These convergence rates are wrong in the L_2 norm, but acceptible in the H_1 norm. This indicates that the multi-grid method is not working as it is supposed to. The number of steps is too high for the finest grid, but the method is able to perform at a finer grid then the direct solver. This is represented in the table below:

h on fine grid	J	Tolerance	Iterations	H_1 Error	L_2 Error
0.25	2	0.050391	4	0.12049	0.40694
0.125	3	0.0097206	6	0.074254	0.3592
0.0625	4	0.001792	27	0.027975	0.31359

Using distributed gauss-seidel method on the same problem (3), the convergence rates are extremely close to the multi-grid method, but slightly higher in both the L_2 and H_1 norms. This is shown in the image below:

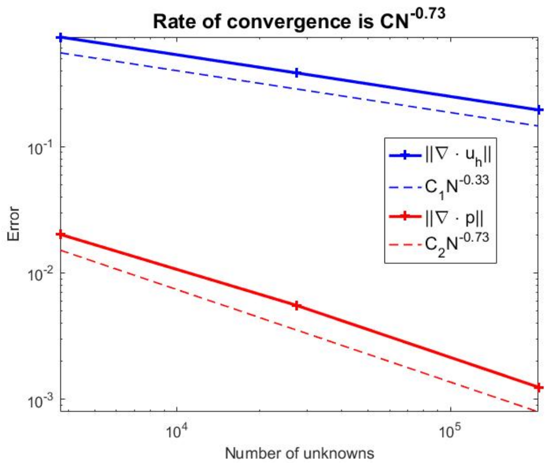


These convergence rates are wrong in the L_2 norm, but acceptible in the H_1 norm. This indicates that the DGS method is doing the same as the multi-grid method. In addition, the number of iterations is almost identical to the multi-grid method, so the multi-grid method tested does not show any improvement over the DGS method. A table of this information is given below:

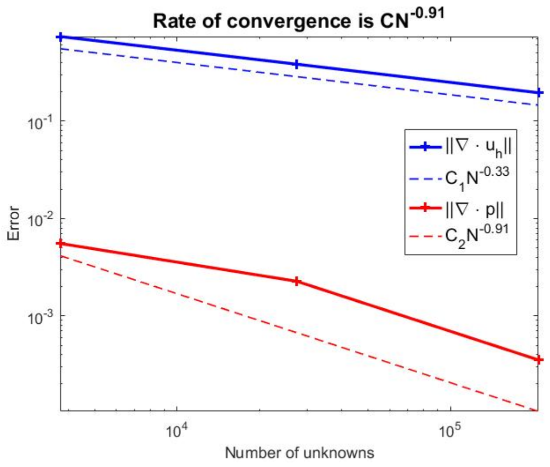
h	Tolerance	Iterations	H_1 Error	L_2 Error
0.25	0.050391	4	0.13072	0.43251
0.125	0.0097206	6	0.079172	0.37362
0.0625	0.001792	29	0.027793	0.31431

3.2.2 Convergence of p

Using the multi-grid method, the rates of convergence for the divergence of p and the divergence of u are recorded. the convergence rates are higher then the direct-solver, but the actual error in $\nabla \cdot p$ and $\nabla \cdot u$ is higher.



For the distributed gauss-seidel method, the convergence rates are similar, but $\nabla \cdot p$ converges slightly quicker. This is shown in the image below:



4 Conclusion

When solving the system of partial differential equations (1) in 3 dimensions, the amount of memory required to store the matrices is very high, as well as the amount of memory needed to solve the resulting discrete system of equations (11) or (13). The multi-grid method could be improved and debugged so that it converges at the proper rates in both the L_2 and H_1 norms. This will make the multi-grid method better, since it allows for the system of equations to be solved on a finer grid.