2013

Numerical Simulations For The Flow Of Rocket Exhaust Through A Granular Medium

Kristina Kraakmo
University of Central Florida

NUMERICAL SIMULATIONS FOR THE FLOW OF ROCKET EXHAUST THROUGH A GRANULAR MEDIUM

by

KRISTINA KRAAKMO
B.S. University of Central Florida, 2009

A thesis submitted in partial fulfilment of the requirements
for the degree of Master of Science
in the Department of Mathematics
in the College of Sciences
at the University of Central Florida
Orlando, Florida

Fall Term
2013

Major Professor: Brian Moore

ABSTRACT

Physical lab experiments have shown that the pressure caused by an impinging jet on a granular bed has the potential to form craters. This poses a danger to landing success and nearby spacecraft for future rocket missions. Current numerical simulations for this process do not accurately reproduce experimental results. Our goal is to produce improved simulations to more accurately and efficiently model the changes in pressure as gas flows through a porous medium. A two-dimensional model in space known as the nonlinear Porous Medium Equation as it is derived from Darcy's law is used. An Alternating-Direction Implicit (ADI) temporal scheme is presented and implemented which reduces our multidimensional problem into a series of one-dimensional problems. We take advantage of explicit approximations for the nonlinear terms using extrapolation formulas derived from Taylor-series, which increases efficiency when compared to other common methods. We couple our ADI temporal scheme with different spatial discretizations including a second-order Finite Difference (FD) method, a fourth-order Orthogonal Spline Collocation (OSC) method, and an N^{th}-order Chebyshev Spectral method. Accuracy and runtime are compared among the three methods for comparison in a linear analogue of our problem. We see the best results for accuracy when using an ADI-Spectral method in the linear case, but discuss possibilities for increased efficiency using an ADI-OSC scheme. Nonlinear results are presented using the ADI-Spectral method and the ADI-FD method.

ACKNOWLEDGMENTS

I would like to express my sincerest gratitude to my advisor Dr. Brian Moore for continuous guidance and support through my journey here at UCF. His constant encouragement and advising was essential to my accomplishments as a graduate student. I would like to thank Dr. Phil Metzger for providing inspiration, collaboration, and experimental prowess which brought this project to life. I also thank Dr. Brennan and Dr. Rollins for serving on my committee and for editing and contributing suggestions for my research. I thank Brian Brennan and Whitney Keith for their large contributions to this project and their support of my work. A special thanks is given to my peer and friend Maria Strawn for her collaboration and assistance. Finally, thanks to my mother Ellen Kelleher and to Shannon Dickson for their constant support throughout my graduate career.

This project was supported in part by the National Aeronautics and Space Administration through the University of Central Florida's Florida Space Consortium. I also would not have been able to pursue and complete my degree without the financial assistance from the UCF McNair Fellowship and the STATESS scholarship.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

1.1 History and Motivation

Soil displacement caused by the continuous firing of an impinging jet landing on a porous surface can cause dangerous effects including as cratering and visual impairment. Understanding the conditions under which these circumstances will occur has recently become an important concern to NASA [16]. Upon the landing of the Apollo 12 lunar module, eroded soil was sprayed at a velocity estimated to be as high as 2000 m/s which caused damage to the Surveyor 3 spacecraft that was 155m away from the landing site [16]. Due to the firing of the module's engine at a distance of a few feet above the surface during the Apollo 15 lunar mission, soil was blowing so severely that it completely impaired visibility. This caused the module to land along the edge of a crater at a 15 degree angle, with only three of its four legs on the surface [14], [19].

While previous research of these blast effects caused by landings has contributed to successful Apollo and Viking missions, it has been noted that future missions may suffer from more serious situations caused by such severe blowing of granular material. Planetary regolith can be defined as the outer layer of loose granular material that can be found on the moon and Mars. With larger payloads and the possibility of landing multiple spacecraft near one another, the blowing of regolith caused by rocket exhaust may pose serious challenges [16]. These include the possibility of crater formation and soil erosion. In an effort to prevent such complications, NASA has developed a program to research plume and soil mitigation techniques. This research serves the purpose of engineering hardware near the launch and landing sites for future missions to ensure both the safety of the crew and success of the mission. In order to do so, it is imperative that the behavior of regolith under mission conditions is sufficiently understood. For example, it has been suggested that barriers be engineered around the landing site to prevent flying regolith from damaging nearby

spacecraft or outposts. This type of prevention mechanism is largely dependent upon the accurate calculation of the angle at which regolith will be blown during landing. While this type of calculation is not difficult given the physical parameters, predictions may be off by orders of magnitude if the regolith exhibits cratering. If any type of cratering occurs, the material may be blasted at such a severe angle that some of it may completely overshoot the height of the barrier. Other potential problems include the loss of visibility upon landing the module due to the blowing material, and even damage to the lander itself.

Physically simulating these experiments realistically in a lab requires large amounts of lunar simulant and a hypersonic engine in a large vacuum at reduced gravity to account for the moon's lack of atmosphere. Due to the high cost of such experiments and the lack of necessary equipment, researchers have turned to numerical simulations for understanding how the blast of gas from a rocket will flow through planetary regolith and the displacement effects it has on the porous material.

1.2 Overview

The bulk of this thesis is dedicated to the study of numerically simulating the pressure flow through a porous medium. In Chapter 4, we describe the coupling of our results with equations of elasticity that determine how the flow of gas actually displaces the regolith. When the linear elasticity equation fails, we can infer that the regolith itself has failed. We conclude that regolith which fails to act like an elastic material will begin to act like a plastic and display cratering behavior. We discuss in detail the specific tests that can be performed to determine whether or not a crater is formed.

Various cratering phenomena have been observed to occur due to the pressure exerted upon a granular medium under rocket landing conditions. In particular, two of the identified mechanisms are known to be *bearing-capacity failure* (BCF) and *diffusion-driven flow* (DDF) [15]. We can

think of BCF as the downward shoving of material caused by the pressure exceeding the bearing capacity of the soil which forms a vertical depression. On the other hand, DDF describes the forced shearing of soil caused by the flow of pressure through the pore spaces within the medium. Geometrically, observations have shown that material is moved vertically by BCF and horizontally by DDF. When we use our simulations to predict crater formation, we focus primarily on the formation due to BCF or DDF.

A model has been created and analyzed in [24] to couple the pressure and displacement equations. For pressure simulations, this model uses the second-order Crank-Nicholson method in time and the Chebyshev Spectral method in space. Displacement results were obtained using a finite element method. However, tests for cratering using this model were inconsistent with cratering results from two experimental cases which we present in Section 1.3. Furthermore, another model has been implemented to solve this problem using a high-performance finite-element method for the Porous Medium Equation. While these results are more promising, they still do not completely reproduce the experimental results. This has motivated us to further research numerical techniques to more accurately and efficiently simulate pressure for our situation.

In this thesis, a model for simulating the pressure due to rocket exhaust is proposed and analyzed. The contributions to the general study of porous medium displacement under rocket landing conditions that are made in this thesis can be listed as follows:

- the implementation of an efficient second-order time-stepping scheme for this problem that can be easily parallelized to run computationally expensive simulations,
- the set-up of a fourth-order spatial discretization that allows for elegant domain decomposition to simulate steep-gradient conditions,
- the development of MatLab code that can be used and extended for similar problems,
- a comparison of three common spatial discretizations to help identify current modeling in-

accuracies.

1.3 The Problem

The porous medium equation

$$\partial_t P = \Delta(P^m), \qquad \text{for } m > 1, \tag{1.1}$$

is a well known nonlinear parabolic equation [22]. While this equation is used to describe a plethora of physical dynamic phenomena, one of its main physical applications is modeling the flow of an ideal gas through a porous medium [22]. We derive a form of (1.1), with $m = 2$, that serves as the mathematical model for our physical problem.

We begin by considering Darcy's Law describing diffusive flow through a permeable medium

$$\mathbf{v} = -\frac{\kappa}{\eta\epsilon}\nabla \cdot P, \tag{1.2}$$

where $\mathbf{v}$ is the gas velocity in the medium, P is the pressure of the field, k is the permeability of the medium, η is the viscosity of the gas, and ϵ is the porosity of the medium. The Ideal Gas law states

$$\gamma = \frac{P}{RT}, \tag{1.3}$$

where γ is the gas density, R is the ideal gas constant, and T is temperature. By combining equations (1.2) and (1.3) with the conservation of mass

$$\nabla \cdot (\gamma \mathbf{v}) = -\frac{\partial \gamma}{\partial t}, \tag{1.4}$$

we can perform the following derivation:

$$
\begin{aligned}
& -\frac{\kappa}{\eta\epsilon RT}\nabla\cdot(P\nabla P) = -\frac{1}{RT}\frac{\partial P}{\partial t} \\
\implies \quad & \nabla(P\nabla\cdot P) = \frac{\eta\epsilon}{\kappa}\frac{\partial P}{\partial t} \\
\implies \quad & \nabla^2\cdot P^2 = \frac{2\eta\epsilon}{\kappa}\frac{\partial P}{\partial t}.
\end{aligned} \tag{1.5}
$$

We use this equation to model the time evolution of rocket exhaust as it diffuses through regolith. The aim is to predict the formation of craters in cases that are unknown through numerical simulation. By doing this, we can determine the rocket exhaust and material conditions under which cratering will occur. The following case studies serve as a means to verify that our simulations correspond to reality.

Lunar Case -an actual lunar landing where no crater was formed that is spatially unbounded

Lab Case -an experiment conducted within a box-shaped apparatus at Kennedy Space Center's Granular Mechanics and Regolith Operations Laboratory where a crater was formed.

All experiments must be done using three dimensions in order to physically simulate reality. However, due to the symmetric nature of our experiments, we make a coordinate shift from cylindrical coordinates to Cartesian coordinates. It can be seen in Figure (1.1) that there are only two degrees of freedom in the three dimensional case. That is, the values of the pressure are independent of the θ direction. Therefore, we make a shift down to two dimensions which also simplifies our computations. The cylinder on the left in Figure (1.1) represents the three dimensional initial pressure exerted by the rocket nozzle. The cylinder on the right shows a slice representing the spatial domain in two dimensions.

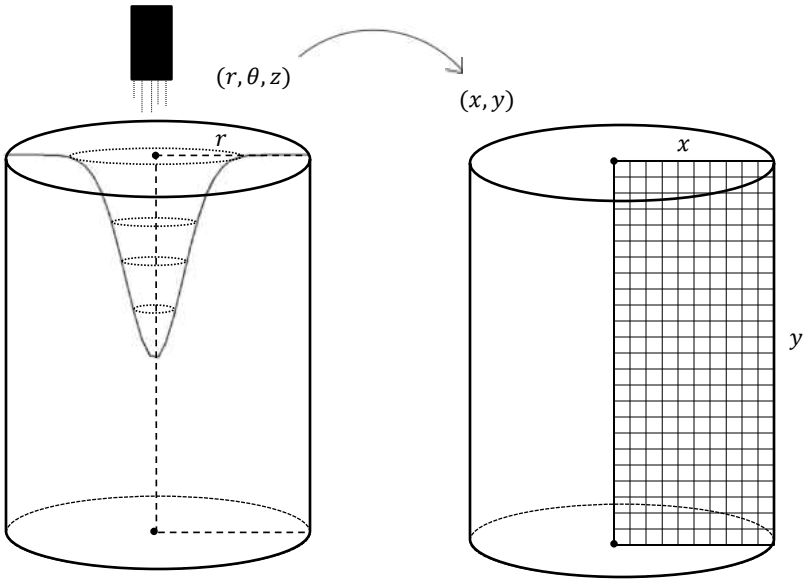


Figure 1.1: Coordinate shift from cylindrical to Cartesian coordinates.

It is helpful to first consider the initial condition representing the firing of rocket exhaust into the medium at $t = 0$. Following the lead of [19], we model the initial condition for our numerical model using a Gaussian function of the form

$$g_0(r, z) = E \cdot e^{-\frac{1}{\sigma^2}\left(x^2+y^2\right)}, \tag{1.6}$$

where σ is the radius of the jet nozzle in meters. The variables x and y denote the horizontal and vertical spatial dimensions. The amplitude E is calculated in terms of σ and the thrust of the rocket by

$$E = \frac{\text{Thrust}}{\pi\sigma^2}.$$

A list of all parameters used for both experiments are included in Table (1.1).

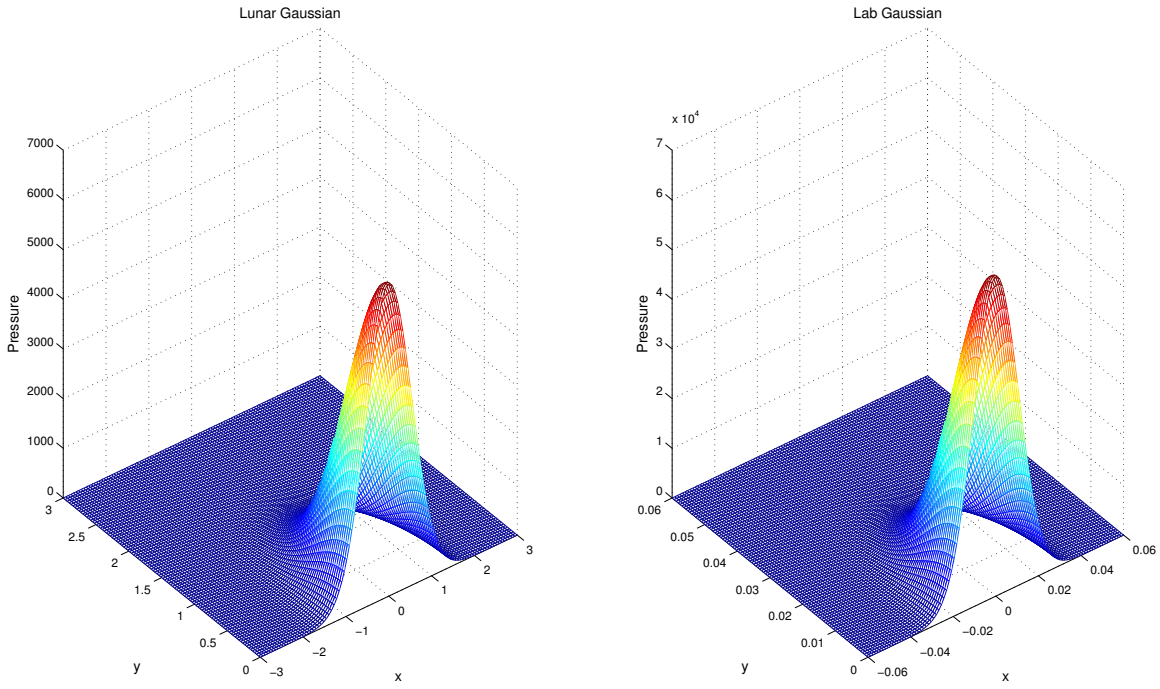
Table 1.1: Experimental Parameters

Parameters	units	Description	Lunar Case	Lab Case
Thrust	N	thrust of the rocket	11,120	45.5
σ	m	radius of the nozzle	0.75	0.015
p_0	psi	characteristic pressure of environment	1	14.7
κ	m^2	permeability of the medium	10^{-12}	4.68×10^{-10}
η	Ns/m^2	viscosity of the gas	5×10^{-5}	6.7×10^{-5}
ϵ	-	porosity of the medium	0.51	0.5
E	N/m^2	exhaust from the rocket nozzle	68	63,662

We plot the initial conditions for each test case in Figures (1.2(a)) and (1.2(b)). The domain is chosen to be $[-4\sigma, 4\sigma] \times [0, 4\sigma]$ to account for boundary conditions. While these two Gaussians may look similar, we note that the distribution in Figure (1.2(b)) is much steeper than that of Figure (1.2(a)). The lunar distribution has an amplitude of about 6,000m on domain [-3, 3]×[-3, 3], while the lab distribution has an amplitude of about 64,000m on the much smaller domain [-0.06, 0.06]×[-0.06, 0.06]. This is expected since there is no atmosphere in the lunar case so the plume should be more spread out while the lab case acts more like a jet. Under these conditions, it is easily seen that the less steep lunar case did not create a crater while cratering definitely occured in the lab. We see the challenge for the lab case when attempting to approximate derivatives of such a steep function.

If we think about this distribution in terms of cylindrical coordinates, we can see that the evolution of diffusion is radially symmetric. In order to take advantage of this property, we rewrite (1.5) using cylindrical coordinates

$$\frac{\partial^2 p^2}{\partial z^2} + \frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial p^2}{\partial r}\right) = \frac{2\epsilon\eta}{\kappa}\frac{\partial p}{\partial t}. \tag{1.7}$$



(a) (b)

Figure 1.2: Gaussian initial conditions for (a) the lunar case and (b) the lab case. Parameters for (a) are Thrust = 11,120N, $\sigma = .75$m. Parameters for (b) are Thrust = 45.5N, $\sigma = .015$m. The domain in each case is $[-4\sigma, 4\sigma] \times [0, 4\sigma]$, where σ is the standard deviation of the distribution.

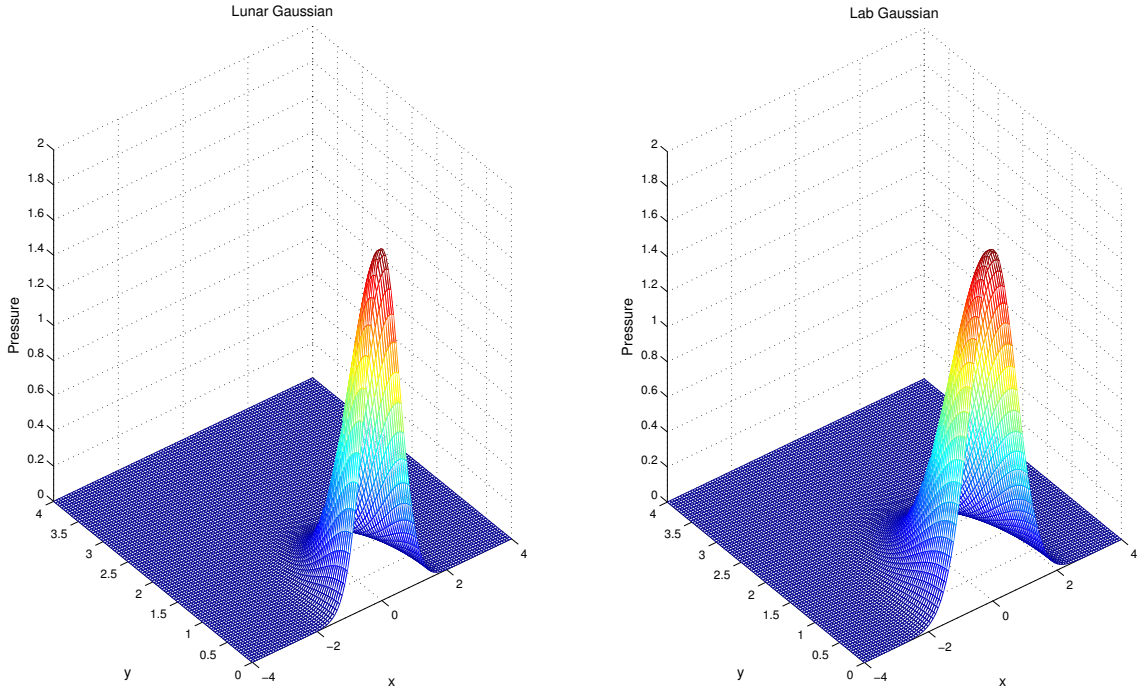
By implementing the linear change of variables

$$p \to pp_0, \qquad r \to rr_0, \qquad z \to zr_0, \qquad t \to \frac{2\epsilon\eta r_0^2}{\kappa p_0} t, \tag{1.8}$$

where p_0 is the characteristic pressure with units of N/m^2 and r_0 is the radius of the jet nozzle in meters, we arrive at the dimensionless equation

$$p_t = \frac{\partial}{\partial z}(pp_z) + \frac{\partial}{\partial r} pp_r + \frac{1}{r} pp_r, \tag{1.9}$$

where we utilize the notation $p_t = \dfrac{\partial p}{\partial t}$, $p_z = \dfrac{\partial p}{\partial z}$, and $p_r = \dfrac{\partial p}{\partial r}$. The is the form of the equation



(a) (b)

Figure 1.3: Rescaled Gaussian initial conditions for (a) the lunar case and (b) the lab case.

which we will be solving in our nonlinear simulations.

Expanding (1.9) yields

$$p_t = pp_{zz} + pp_{rr} + (p_z)^2 + (p_r)^2 + \frac{1}{r}pp_r. \tag{1.10}$$

Once we make the change of variables in (1.8), our initial conditions are smoothed out greatly. We plot the scaled versions of both the lunar and the lab initial conditions in Figures (1.3(a))-(1.3(b)).

CHAPTER 2: DISCRETIZATION METHODS

2.1 Explicit Finite Difference Method

One of the first numerical studies of this problem, conducted in [19], arose when preparing to land the Apollo Lunar Module on the moon. At this time, the existence of a grandular medium was known to be present on the lunar surface and potential hazards during landing had been identified. One hazard considered was damage to the spacecraft caused by the blowing of regolith. Another was such severe exhaust plume that would cause the module to overturn. Preliminary lab experiments were conducted to test the firing of a Surveyor engine and analyze the erosion and depressions left in the soil for different porous media. While these lab experiments attempted to simulate lunar conditions, accounting for the lunar atmosphere and reduced gravity was not possible. Due to the physical constraints of lab experimentation on Earth, researchers turned to numerical simulations of this process.

To numerically model this problem, the authors of [19] presented an explicit five-point finite-difference method. They begin by making the following forward in time and centered in space finite difference discretizations:

$$p_t(x,y,t_n) \approx \frac{P_{i,j}^{n+1} - P_{i,j}^n}{\Delta t}, \tag{2.1}$$

$$p_r^2(x,y,t_n) \approx \frac{(P_{i+1,j}^n)^2 - (P_{i-1,j}^n)^2}{2\Delta r}, \tag{2.2}$$

$$p_{rr}^2(x,y,t_n) \approx \frac{(P_{i+1,j}^n)^2 - 2(P_{i,j}^n)^2 + (P_{i-1,j}^n)^2}{(\Delta r)^2}, \tag{2.3}$$

$$p_{zz}^2(x,y,t_n) \approx \frac{(P_{i,j+1}^n)^2 - 2(P_{i,j}^n)^2 + (P_{i,j-1}^n)^2}{(\Delta z)^2}, \tag{2.4}$$

where P is the discrete approximation of the continuous function p. By plugging these discretiza-

tions into the expanded dimensionless form of equation (1.7),

$$\frac{\partial^2 p^2}{(\partial z)^2} + \frac{1}{r}\frac{\partial p^2}{\partial r} + \frac{\partial^2 p^2}{(\partial r)^2} = \frac{\partial p}{\partial t}, \tag{2.5}$$

the following discrete difference equation can be formulated

$$\begin{aligned} P_{i,j}^{n+1} = \quad & P_{i,j}^{n} + \frac{\Delta t}{(\Delta z)^2}\left[(P_{i,j+1}^{n})^2 - 2(P_{i,j}^{n})^2 + (P_{i,j-1}^{n})^2\right] + \\ & \frac{\Delta t}{(\Delta r)^2}\left[\left(1 + \frac{\Delta r}{2r}\right)(P_{i+1,j}^{n})^2 + \left(1 - \frac{\Delta r}{2r}\right)(P_{i-1,j}^{n})^2 - 2(P_{i,j}^{n})^2\right], \end{aligned} \tag{2.6}$$

where $r \in [r_1, r_{N_r+1}]$, $z \in [z_1, z_{N_z+1}]$, and $t \in [t_0, t_f]$. We have that $\Delta t = t_n - t_{n-1}$, where $n = 0, 1, \ldots, t_f$, $\Delta z = z_j - z_{j-1}$ for $j = 1, \ldots, N_z + 1$, and $\Delta r = r_i - r_{i-1}$ for $i = 1, \ldots, N_r + 1$. The spatial domain is restricted to $r \in [-1, 1]$, $z \in [0, 1]$. The domain for the $r-$dimension is chosen so the singularity when $r = 0$ is avoided. This also significantly simplifies the boundary conditions which is made clear in Section 4.2.1. This type of treatment is considered in [21]. We denote by $P_{i,j}^{n}$ the point $P(r_i, z_j, t_n)$.

A uniform spatial mesh is imposed by setting $\Delta r = \Delta z$ and $M = \dfrac{\Delta t}{(\Delta z)^2}$. By combining like terms, equation (2.6) can be rewritten as

$$P_{i,j}^{n+1} = P_{i,j}^{n} + M\left[(P_{i,j+1}^{n})^2 + (P_{i,j-1}^{n})^2 + \left(1 + \frac{\Delta r}{2r}\right)(P_{i+1,j}^{n})^2 + \left(1 - \frac{\Delta r}{2r}\right)(P_{i-1,j}^{n})^2 - 4(P_{i,j}^{n})^2\right]. \tag{2.7}$$

Impervious boundary conditions where $r = r_1$, $r = r_{N_r+1}$, and $z = z_{N_z+1}$ are the most representative of the physics for our problem. For such conditions, the pressure along one layer outside of each boundary is exactly the same as the pressure along one layer inside the boundary. These boundary conditions simulate zero flow of gas across the boundary, so we account for them by setting

$$r_1 = r_2, \qquad r_{N_r+1} = r_{N_r}, \qquad z_{N_z+1} = z_{N_z}.$$

When implementing a difference approximation, the solution has a finite domain of dependence. That is, each point in the numerical solution is dependent upon an interval of points on the original mesh. In order for the solution of (2.5) to converge using the difference method (2.7), the choice of Δt and Δz must satisfy the Courant-Friedrichs-Lewy (CFL) condition for stability,

$$\frac{\Delta t}{(\Delta z)^2} < \frac{1}{4}. \tag{2.8}$$

It is clear that a better spatial approximation to the differential equation can be obtaind by using a smaller step-size. However, since we must adhere to the constraint (2.8), using smaller spatial step size requires the use of an even smaller time step size. This relationship makes it extremely difficult to simulate accurate approximations with sufficient efficiency. For this reason, we turn to a linearly implicit method that is not restricted by such a strict constraint as (2.8) allowing for increased efficiency.

2.2 Alternating-Direction Implicit Finite Difference Method

Alternating-Direction Implicit (ADI) methods have been used to efficiently solve problems in multiple dimensions [2]. The efficiency of an ADI method comes from its ability to step forward by a timestep of $\Delta t/2$ implicitly in the first dimension, say along horizontal lines of a 2D matrix, and explicitly in the second, along vertical lines of a 2D matrix. Then the method steps forward in time again by a timestep of $\Delta t/2$ implicitly in the second dimension and explicitly in the first. We make use of Figure (2.1) to explain the general behavior of the ADI time-stepping scheme. First, the method carries out the explicit calculation after one-half of a timestep (the right-hand side of (2.12a)) on points such as those marked by a $\bigcirc$ and then performs the implicit calculation after one-half of a timestep (the left-hand side of (2.12a)) at nodes such as those marked with a $\square$. Then the method performs the explicit calculation (the right-hand side of (2.12b)) at points such

as those marked by a ○ and then performs the implicit operation (the right-hand side of (2.12b)) after one-half of a timestep at nodes such as those marked with a □. Boundary conditions for the

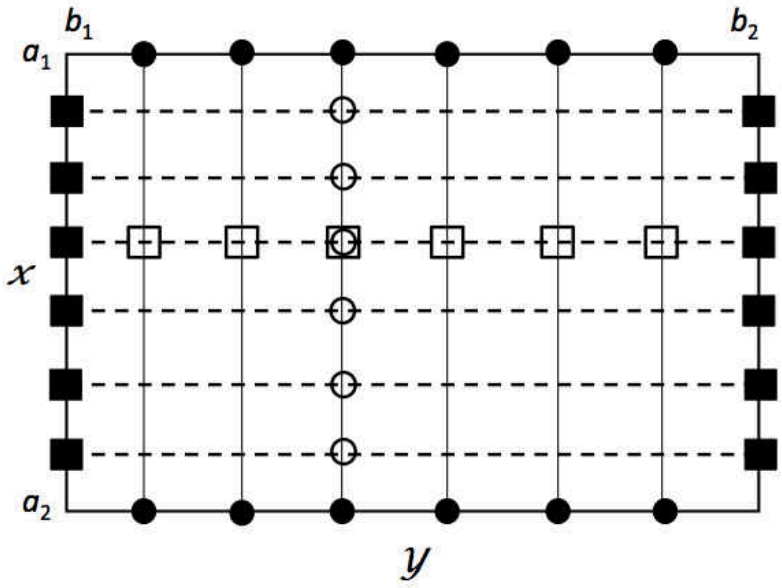


Figure 2.1: Rectangular grid with general domain $M = [a_1, a_2] \otimes [b_1, b_2]$ for an ADI method. In the figure, x remains constant along horizontal dashed lines, and y is constant along vertical solid lines.

$x-$direction are needed at points marked with a ●, and boundary conditions for the $y-$direction are needed at points marked with a ■.

This allows a two-dimensional problem to be broken down into a series of one-dimensional problems [2]. This scheme proves to be much more efficient and easier to implement than the commonly used second-order accurate Crank-Nicolson scheme [17], which requires the simultaneous solving of systems of equations implicitly in both dimensions. The Crank-Nicolson scheme used in [24] requires at least $2N^4$ long operations while an ADI scheme requires only $2N^2$. While an implicit method is more computationally expensive than the explicit scheme from Section 2.1, the time-step size for an ADI scheme is not limited by (2.8). However, similar to the Crank-Nicolson

scheme, we must still satisfy the condition

$$\frac{\Delta t}{(\Delta z)^2} < 1 \tag{2.9}$$

in order to ensure accuracy. We note that this restriction allows for a larger time-step given the same number of spatial nodes than the finite-difference restriction (2.8).

Let us first consider the linear analogue to our problem

$$u_t = \nabla^2 u. \tag{2.10}$$

Let U^n be a matrix approximation to the solution $u(x, y, t_n)$. That is, the entries of the matrix U^n are the values for the approximation of u at the spatial grid points at the n^{th} time step. An ADI scheme for this problem implicitly approximates the solution in the $x-$dimension at an intermediate level, $U^{n+1/2}$, with an explicit approximation in the $y-$dimension. Then the approximate solution U^n is computed implicitly in the $y-$dimension and explicitly in the $x-$dimension. We write this two-step scheme as

$$\frac{U^{n+1/2} - U^n}{\Delta t/2} = \delta_x^2 U^{n+1/2} + \delta_y^2 U^n, \tag{2.11a}$$

$$\frac{U^{n+1} - U^{n+1/2}}{\Delta t/2} = \delta_x^2 U^{n+1/2} + \delta_y^2 U^{n+1}, \tag{2.11b}$$

where δ_x^2 and δ_y^2 are differentiation matrices which approximate second-derivatives by $\delta_x^2 U \approx U_{xx}$ and $\delta_y^2 U \approx U_{yy}$. Some rearrangement yields

$$(I - \frac{\Delta t}{2}\delta_x^2)U^{n+1/2} = (I + \frac{\Delta t}{2}\delta_y^2)U^n, \tag{2.12a}$$

$$(I - \frac{\Delta t}{2}\delta_y^2)U^{n+1} = (I + \frac{\Delta t}{2}\delta_x^2)U^{n+1/2}, \tag{2.12b}$$

where I is the identity matrix. With this arrangement it is easily seen that all terms on the right-

hand side of (2.12a) are explicitly dependent only upon the approximation U^n computed during the previous time-step. Also, the implicit computation of (2.12a) must be done in only the $x-$direction. Similarly, all terms on the right-hand side of (2.12b) are dependent solely upon the approximation $U^{n+1/2}$ which is already computed in (2.12a). The implicit computation of (2.12b) must only be done in the $y-$direction.

We now go back to our nonlinear problem and set up our ADI scheme. We can express the nonlinear porous medium equation (1.10) on a rectangular polygon $M = (-1, 1) \times (0, 1)$ in the general form

$$p_t = (L_x + L_y)p + f(x, p, p_x, p_y), \tag{2.13}$$

with the initial condition

$$p(x, y, 0) = g(x, y), \quad (x, y) \in M, \tag{2.14}$$

and boundary conditions

$$p_x(-1, y, t) = 0, \qquad p_x(1, y, t) = 0, \tag{2.15}$$

$$p(x, 0, t) = p(x, 0, 0), \qquad p_y(x, 1, t) = 0. \tag{2.16}$$

The Neumann boundary conditions at $x = -1$, $x = 1$, and $y = 1$ simulate the impervious boundary conditions discussed in Section 2.1. By imposing a Dirichlet boundary condition when $y = 0$ we are assuming that the pressure exerted by the jet on the medium is constant until shutoff.

We let L_x and L_y be the differential operators given by

$$L_x p = p p_{xx}, \quad L_y p = p p_{yy}, \tag{2.17}$$

and $f(x, p, p_x, p_y)$ be the first-order term given by

$$f(x, p, p_x, p_y) = (p_x)^2 + (p_y)^2 + \frac{1}{x} p p_x. \tag{2.18}$$

We refer to $u_h \in M$ as the approximation to the solution $p(x, y, t)$ of (2.13)-(2.16). We first define our discrete derivative matrices, $\mathcal{L}_x^n$ and $\mathcal{L}_y^n$, to approximate L_x and L_y, respectively. When coupled with the finite-difference spatial approximation, we have that

$$\mathcal{L}_x^n\Big(\tilde{u}^n(x, y)\Big) \cdot u_h^n(x, y) = \tilde{u}^n(x, y) \cdot D_x^2 u_h(x, y), \tag{2.19}$$

$$\mathcal{L}_y^n\Big(\tilde{u}^n(x, y)\Big) \cdot u_h^n(x, y) = \tilde{u}^n(x, y) \cdot D_y^2 u_h(x, y), \tag{2.20}$$

where $\tilde{u}^n$ is computed using extrapolation formulas to give an approximation of u^{n+1}. This is explained in detail in Section 2.2.1. We have that $D_x^2 u \approx u_{xx}$ and $D_y^2 u \approx u_{yy}$ are centered-difference differentiation matrices of the form

$$D^2 = \frac{1}{h^2} \begin{pmatrix} -1 & 1 & 0 & 0 & 0 & \dots & 0 \\ 1 & -2 & 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & -2 & 1 & 0 & \dots & 0 \\ \vdots & & \ddots & \ddots & \ddots & & \vdots \\ 0 & \dots & 0 & 1 & -2 & 1 & 0 \\ 0 & \dots & 0 & 0 & 1 & -2 & 1 \\ 0 & \dots & 0 & 0 & 0 & 1 & -1 \end{pmatrix}.$$

The elements in the first and last rows account for our boundary conditions which are currently set to homogeneous Neumann conditions. To account for the Dirichlet boundary condition for $y = 0$, the first element in the first row is set to one while the remaining values in the first row are zero.

We also provide a discretization of the function f at time t_n as

$$f^n\left(x, \tilde{u}^n, \hat{u}_x^n, \tilde{u}_y^n\right) = (\tilde{u}_y^n)^2 + (\hat{u}_x^n)^2 + \frac{1}{x}\tilde{u}^n\hat{u}_x^n, \tag{2.21}$$

where $\hat{u}^n$ is also computed using extrapolation. All derivatives of $\tilde{u}$ and $\hat{u}$ are approximated using a centered-difference approximation in the finite-difference case. That is, $D_x\tilde{u} \approx \tilde{u}_x$ and $D_y\hat{u} \approx \hat{u}_y$ where D_x and D_y are differentiation matrices of the form

$$D = \frac{1}{2h}\begin{pmatrix} -1 & 1 & 0 & 0 & 0 & \dots & 0 \\ -1 & 0 & -1 & 0 & 0 & \dots & 0 \\ 0 & -1 & 0 & -1 & 0 & \dots & 0 \\ \vdots & & \ddots & \ddots & \ddots & & \vdots \\ 0 & \dots & 0 & -1 & 0 & 1 & 0 \\ 0 & \dots & 0 & 0 & -1 & 0 & -1 \\ 0 & \dots & 0 & 0 & 0 & 1 & -1 \end{pmatrix}.$$

Now that we have described the method for the linear case, we can write the ADI scheme for the nonlinear case.

$$\begin{aligned} \frac{u_h^{n+1/2} - u_h^n}{\Delta t/2} &= L_x^{n+1/2}(\tilde{u}_h)u_h^{n+1/2} + L_y^{n+1/2}(\tilde{u}_h)u_h^n \\ &\quad + f^{n+1/2}\Big(x, \tilde{u}_h, (\hat{u}_h)_x, , (\tilde{u}_h)_y\Big), \end{aligned} \tag{2.22a}$$

$$\begin{aligned} \frac{u_h^{n+1} - u_h^{n+1/2}}{\Delta t/2} &= L_x^{n+1/2}(\tilde{u}_h)u_h^{n+1/2} + L_y^{n+1/2}(\tilde{u}_h)u_h^{n+1} \\ &\quad + f^{n+1/2}\Big(x, \tilde{u}_h, (\hat{u}_h)_x, (\tilde{u}_h)_y\Big). \end{aligned} \tag{2.22b}$$

Some rearrangement yields

$$\Big[I - \frac{\Delta t}{2} L_x^{n+1/2}(\tilde{u}_h)\Big] u_h^{n+1/2} = \Big[I + \frac{\Delta t}{2} L_y^{n+1/2}(\tilde{u}_h)\Big] u_h^n + f^{n+1/2}\left(x, \tilde{u}_h, (\hat{u}_h)_x, (\tilde{u}_h)_y\right), \tag{2.23a}$$

$$\Big[I - \frac{\Delta t}{2} L_y^{n+1/2}(\tilde{u}_h)\Big] u_h^{n+1} = \Big[I + \frac{\Delta t}{2} L_x^{n+1/2}(\tilde{u}_h)\Big] u_h^{n+1/2} + f^{n+1/2}\left(x, \tilde{u}_h, (\hat{u}_h)_x, (\tilde{u}_h)_y\right). \tag{2.23b}$$

Just as in the linear case (2.12a)-(2.12b), the right-hand side of (2.23a) explicitly depends only upon the approximation at the previous time-step, u_h^n. The left-hand side of (2.23a) is only calculated implicitly in the $x-$direction. Similarly, the right-hand side of (2.23b) explicitly depends on the approximation of $u_h^{n+1/2}$ that is calculated in (2.23a). The implicit calculation on the left-hand side of (2.23b) is solely done in the $y-$direction.

2.2.1 Extrapolation

We make use of extrapolation to predict the solution at the future time step which reduces our problem to a linear problem. In order to do so, we begin by defining $\hat{u}_h^n(\cdot, y) \in [-1, 1]$ and $\tilde{u}_h^n(x, \cdot) \in [0, 1]$ which approximate $u(x, y, t_n)$ for $x \in (-1, 1)$ and $y \in (0, 1)$, respectively. If we let $\hat{u}_h^0(x, y) = g(x, y)$ and $\tilde{u}_h^0(x, y) = g(x, y)$, then we can use Taylor's theorem to define

$$\begin{aligned} \hat{u}_h^{1/2} &= g(x,y) + \frac{\Delta t}{2}\Big[f^0\big(x, g(x,y), (\hat{u}_h^0)_x(x,y), (\tilde{u}_h^0)_y(x,y)\big) \\ &\quad + \mathcal{L}_x\big(\tilde{u}^0(x,y)\big) g(x,y) + \mathcal{L}_x\big(\tilde{u}^0(x,y)\big) g(x,y), \quad \text{for all } y \in M_y \end{aligned} \tag{2.24}$$

$$\begin{aligned} \tilde{u}_h^{1/2} &= g(x,y) + \frac{\Delta t}{2}\Big[f^0\big(x, g(x,y), (\hat{u}_h^0)_x(x,y), (\tilde{u}_h^0)_y(x,y)\big) \\ &\quad + \mathcal{L}_x\big(\tilde{u}^0(x,y)\big) g(x,y) + \mathcal{L}_x\big(\tilde{u}^0(x,y)\big) g(x,y), \quad \text{for all } x \in M_x. \end{aligned} \tag{2.25}$$

In order to calculate an approximation to $u(x,y)$ at the first time step, $u_h^1(x,y)$, we must first approximate a solution to $u_h^{1/2}(x,y)$. By utilizing the second-order Taylor series expansion of $u(x,y,t)$ about $t=0$, we can write

$$\begin{aligned} u(x,y,t+\frac{\Delta t}{2}) &= u(x,y,0)+\frac{\Delta t}{2}u_t(x,y,0)+O(\Delta t^2) \\ &= g(x,y)+\frac{\Delta t}{2}(f^0(x,u,u_x,u_y)+L_xu+L_yu) \end{aligned} \tag{2.26}$$

Our calculations of $\hat{u}_h^{n+1/2}$ and $\tilde{u}_h^{n+1/2}$ for $n=1,\ldots,N_t$ depend solely upon approximations of the solution, u_h^n, at the previous time-steps using

$$\tilde{u}_h^{n+1/2}(x,\cdot) = \frac{3}{2}u_h^n(x,\cdot)-\frac{1}{2}u_h^{n-1}(x,\cdot), \tag{2.27}$$

$$\hat{u}_h^{3/2}(\cdot,y) = 3u_h^{1/2}-2\hat{u}_h^0(\cdot,y), \tag{2.28}$$

$$\hat{u}_h^{n+1/2}(\cdot,y) = 2u_h^{n-1/2}(\cdot,y)-u_h^{n-3/2}(\cdot,y). \tag{2.29}$$

These formulas are obtained from second order Taylor series expansions in time, which is consistent with the overall second-order convergence of an ADI scheme.

2.2.2 Implementation

We present a step-wise list for the implementation of the ADI scheme. This implementation coded in MatLab can be viewed in detail in Appendices A and B for the Chebyshev Spectral spatial discretization and also the OSC method.

Step 1 Initialize the following

- f^0
- $u_h^{1/2}$ according to (2.26)

- $\hat{u}_h^{1/2}$ and $\tilde{u}_h^{1/2}$ according to (2.24) and (2.25).

Step 2 Perform the following for $n = 0$

- calculate the right-hand side (RHS) of (2.23a) along rows of u_h
- calculate the left-hand side (LHS) of (2.23a) along columns of u_h
- calculate the RHS of (2.23b) along columns of u_h
- calculate the LHS of (2.23b) along rows of u_h
- calculate $\hat{u}_h^{3/2}$ according to (2.28)
- calculate $\tilde{u}_h^{3/2}$ according to (2.27)

Step 3 Loop over $n = 1, \ldots t_f$ using the same steps in Step 2 except calculate $\hat{u}_h^{n+1/2}$ and $\tilde{u}_h^{n+1/2}$ using equations (2.29) and (2.27), respectively.

2.3 Chebyshev Spectral Spatial Differentiation

Rather than a second-order accurate finite-difference spatial aproximation, we introduce a spectral differenatiation method. Spectral differentiation is said to have "spectral accuracy," that is, convergence occurs so rapidly that further improvement is prevented by rounding errors [21]. It has been shown that a spectral method for smooth functions typically converge at a rate of $O(N^{-N})$ [21], where N is the number of spatial nodes. Rather than interpolating on equispaced nodes, as done in the commonly used Fourier spectral method, we make use of unevenly-spaced nodes that are clustered near the boundary. By clustering the nodes near the boundary, we can distribute the interpolation error more evenly over the domain. This allows us to minimize the effects of the Runge phenomenon in which case the approximations fail to converge.

2.3.1 Differentiation Matrices

In this section, we set up the differentiation matrices which approximate the spatial first and second derivatives for our problem using Chebyshev Spectral differentiation.

We make use of the Chebyshev nodes spread over the interval $[-1, 1]$, defined by

$$x_j = \cos(j\pi/N_x), \qquad j = 0, 1, \ldots, N_x, \tag{2.30}$$

where $N_x + 1$ is the number of nodes used in the $x-$dimension. The n^{th}-order interpolating polynomials for this method are of the form

$$p_n(x) = \sum_{j=0}^{N_x} \prod_{k=0, k\neq j}^{N} \frac{x - x_k}{x_j - x_k}.$$

The entries of the Chebyshev differentiation matrix, $\mathcal{D}_{N_x}$, that approximate the first derivative in the $x-$dimension of a smooth function are

$$(\mathcal{D}_{N_x})_{00} = \frac{2N_x^2 + 1}{6}, \tag{2.31}$$

$$(\mathcal{D}_{N_x})_{NN} = -\frac{2N_x^2 + 1}{6}, \tag{2.32}$$

$$(\mathcal{D}_{N_x})_{jj} = \frac{-x_j}{2(1 - x_j^2)}, \qquad j = 1, \ldots, N_x - 1, \tag{2.33}$$

$$(\mathcal{D}_{N_x})_{ij} = \frac{c_i(-1)^{i+j}}{c_j(x_i - x_j)}, \qquad i \neq j, \;\; i, j = 0, 1, \ldots, N_x. \tag{2.34}$$

Since our ADI method allows us to approximate derivatives in one dimension at a time, we need not worry about constructing a larger matrix that calculates derivatives in both dimensions simultaneously. We simply have that $\mathcal{D}_{N_x} u \approx u_x$ and $\mathcal{D}_{N_y}(u)^T \approx u_y$. However, for $\mathcal{D}_{N_y}$, we must construct the matrix components given in (2.31) using the Chebyshev nodes on the domain $[0, 1]$.

These can be obtained using

$$y_j = \frac{\cos(j\pi/N_y) + 1}{2}, \qquad j = 0, 1, \ldots, N_y, \tag{2.35}$$

where $N_y + 1$ is the number of nodes used in the $y-$dimension.

The Chebyshev differentiation matrix that approximates the second derivative of a smooth function is evaluated by applying the first derivative matrix to itself, that is,

$$\begin{aligned} \mathcal{D}_{N_x}^2 &= (\mathcal{D}_{N_x})^2, \\ \mathcal{D}_{N_y}^2 &= (\mathcal{D}_{N_y})^2, \end{aligned}$$

where

$$\begin{aligned} \mathcal{D}_{N_x}^2 u &\approx u_{xx}, \\ \mathcal{D}_{N_y}^2 u &\approx u_{yy}. \end{aligned}$$

We include detailed code for the formulation of these matrices according to [21] in Appendix D.

We couple this Chebyshev Spectral method in space with the ADI temporal scheme presented in Section 2.2 by replacing the finite-difference differentiation matrices D_x^2, D_y^2, D_x and D_y with $\mathcal{D}_x^2$, $\mathcal{D}_y^2$, $\mathcal{D}_x$ and $\mathcal{D}_y$, respectively, in equations (2.19) and (2.20).

2.4 Orthogonal Spline Collocation

It has been shown that an orthogonal spline collocation method coupled with an ADI temporal scheme is more efficient than other common methods for a parabolic nonlinear equation such as (2.13) [2]. The benefits of using OSC over other methods commonly used to solve this type of

problem are:

- OSC is cheaper than FD methods for desired accuracy
- OSC does not require a stability parameter or preconditioning
- OSC does not require the approximation of integrals
- OSC is easier to implement than finite element Galerkin methods
- OSC lends itself easily to domain decomposition (beneficial for steep problems with localized activity).

We note that none of these benefits are in comparison with a Spectral method. We can use a spline collocation method to obtain an approximation, $u_h(x, y)$, to our solution $u(x, y)$ in the form of piecewise polynomials that satisfy both the given equation and its boundary conditions at specifically chosen collocation points. We utilize piecewise cubic Hermite polynomials, requiring knowledge of both the function and the first derivative at each point satisfying (2.13) at the Gauss points.

2.4.1 Notation

Let us denote $I_x = (-1, 1)$ as the domain of x and $I_y = (0, 1)$ as the domain of y. Let $d_x = \{x_i\}_{i=1}^{N_x}$ and $d_y = \{y_j\}_{j=1}^{N_y}$ be uniform partitions of $\bar{I}_x$ and $\bar{I}_y$, respectively, where N_x and N_y are positive odd integers. We denote by $\bar{I}_x$ the closure of I_x. We form a Gaussian quadrature by letting $G^x = \{\xi_{ik}^x\}_{i=1,k=1}^{N_x,2}$ be the Gauss points on I_x and $G^y = \{\xi_{jk}^y\}_{j=1,k=1}^{N_y,2}$ be the Gauss points on I_y from [25] given by

$$\xi_{i1}^x = x_{i-1} + \frac{3-\sqrt{3}}{6}h^x, \quad \xi_{i2}^x = x_{i-1} + \frac{3+\sqrt{3}}{6}h^x, \quad i = 1, \ldots, N_x, \tag{2.36}$$

$$\xi_{j1}^y = y_{j-1} + \frac{3-\sqrt{3}}{6}h^y, \quad \xi_{j2}^y = y_{j-1} + \frac{3+\sqrt{3}}{6}h^y, \quad j = 1, \ldots, N_y. \tag{2.37}$$

We have that $h^x = x_i - x_{i-1}$ and $h^y = y_j - y_{j-1}$ denote the spatial step size of the uniform mesh in each direction. Given N_x nodes in d_x and N_y nodes in d_y on a uniform mesh, we have that $h^x = 1/N_x$ and $h^y = 1/N_y$. We note that there are two collocation points between each equidistant node, yielding $2N_x$ Gauss nodes in G^x and $2N_y$ Gauss nodes in G^y. We illustrate these nodes in one dimension in Figure (2.2).

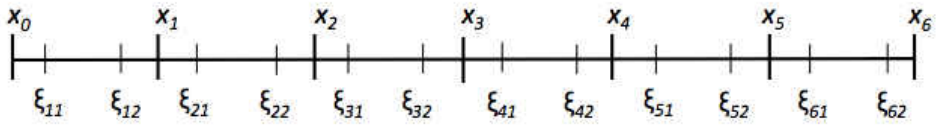


Figure 2.2: Gaussian collocation points on a finite mesh with seven equidistant nodes.

In order to consider the two-dimensional space, M, in which our solution exists, we denote by M_x and M_y the space of piecewise cubic Hermite polynomials on $\bar{I}_x$ and $\bar{I}_y$, respectively, where $M = M_x \otimes M_y$.

We utilize the value basis function v_k and the scaled slope basis function s_k to create bases for M_x and M_y as presented in [25]. These basis functions are defined by

$$v_k(x_i) = \delta_{ki}, \quad v_k'(x_i) = 0, \tag{2.38}$$

$$s_k(x_i) = 0, \quad s_k'(x_i) = \frac{\delta_{ki}}{h}, \tag{2.39}$$

where δ_{hi} is the Kronecker delta function

$$\delta_{ki} = \begin{cases} 1 & \text{for } k = i \\ 0 & \text{for } k \neq i \end{cases}.$$

We choose bases $\{\phi_i(x)\}_{i=1}^{2N_x}$ for M_x and $\{\psi_j(x)\}_{j=1}^{2N_y}$ for M_y defined by

$$\{\phi_i(x)\}_{i=1}^{2N_x+2} = \{v_0, v_1, \ldots, v_{N_x-2}, v_{N_x-1}, v_{N_x}, s_0, s_1, \ldots, s_{N_x-2}, s_{N_x-1}, s_{N_x}\}, \quad (2.40)$$

$$\{\psi_j(y)\}_{j=1}^{2N_y+2} = \{v_0, s_0, v_1, s_1, \ldots, v_{N_y-1}, s_{N_y-1}, v_{N_y}, s_{N_y}\}. \quad (2.41)$$

This choice of bases allows us to set up the form of the OSC solution, u_h, in two dimensions in such a way that all function and derivative values are stored. This allows us to operate on the solution in either dimension with collocation matrices built according to the proper bases. The two dimensional orthogonal spline collocation solution $u_h \in M_x \otimes M_y$ can then be written in terms of these bases as

$$u_h(x,y) = \sum_{i=1}^{2N_x+2} \sum_{j=1}^{2N_y+2} p_{i,j}\phi_i(x)\psi_j(y), \quad (2.42)$$

where the components of matrix P = $\{p_{i,j}\}_{i=1,\ j=1}^{2N_x+2,2N_y+2}$ are determined by the basis

$$\{\phi_i(x)\psi_j(y)\}_{i=1,\ j=1}^{2N_x+2,2N_y+2}$$

of $M_x \otimes M_y$. By representing P as a matrix in terms of the corresponding basis functions from (2.40) and (2.41), we have the $(2N_x+2) \times (2N_y+2)$ matrix

$$P = \begin{pmatrix} v_0v_0 & v_0s_0 & v_0v_1 & v_0s_1 & \ldots & v_0v_{N_y-1} & v_0s_{N_y-1} & v_0v_{N_y} & v_0s_{N_y} \\ v_1v_0 & v_1s_0 & v_1v_1 & v_1s_1 & \ldots & v_1v_{N_y-1} & v_1s_{N_y-1} & v_1v_{N_y} & v_1s_{N_y} \\ \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots \\ v_{N_x}v_0 & v_{N_x}s_0 & v_{N_x}v_1 & v_{N_x}s_1 & \ldots & v_{N_x}v_{N_y-1} & v_{N_x}s_{N_y-1} & v_{N_x}v_{N_y} & v_{N_x}s_{N_y} \\ s_0v_0 & s_0s_0 & s_0v_1 & s_0s_1 & \ldots & s_0v_{N_y-1} & s_0s_{N_y-1} & s_0v_{N_y} & s_0s_{N_y} \\ s_1v_0 & s_1s_0 & s_1v_1 & s_1s_1 & \ldots & s_1v_{N_y-1} & s_1s_{N_y-1} & s_1v_{N_y} & s_1s_{N_y} \\ \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots \\ s_{N_x}v_0 & s_{N_x}s_0 & s_{N_x}v_1 & s_{N_x}s_1 & \ldots & s_{N_x}v_{N_y-1} & s_{N_x}s_{N_y-1} & s_{N_x}v_{N_y} & s_{N_x}s_{N_y} \end{pmatrix}.$$

Here we have that the first component of each element along a row has the concatonated form of (2.41) representing the $y-$dimension, and the second component of each element along a row has the alternating form of (2.40) representing the $x-$dimension. We see the reverse behavior along a column.

According to (2.38) and (2.39), the entries of P consist of $x-$, $y-$ and $xy-$partial derivatives in the following way:

$$\begin{pmatrix}
u_{0,0} & h_y\partial_y u_{0,0} & u_{0,1} & h_y\partial_y u_{0,1} & \dots & u_{0,N_y} & h_y\partial_y u_{0,N_y} \\
u_{1,0} & h_y\partial_y u_{1,0} & u_{1,1} & h_y\partial_y u_{1,1} & \dots & u_{1,N_y} & h_y\partial_y u_{1,N_y} \\
\vdots & \vdots & \vdots & \vdots & & \vdots & \vdots \\
u_{N_x-1,0} & h_y\partial_y u_{N_x-1,0} & u_{N_x-1,1} & \partial_y u_{N_x-1,1} & \dots & u_{N_x-1,N_y} & h_y\partial_y u_{N_x-1,N_y} \\
u_{N_x,0} & h_y\partial_y u_{N_x,0} & u_{N_x,1} & h_y\partial_y u_{N_x,1} & \dots & u_{N_x,N_y} & h_y\partial_y u_{N_x,N_y} \\
h_x\partial_x u_{0,0} & h_xh_y\partial_{xy} u_{0,0} & h_x\partial_x u_{0,1} & h_xh_y\partial_{xy} u_{0,1} & \dots & h_x\partial_x u_{0,N_y} & h_xh_y\partial_{xy} u_{0,N_y} \\
h_x\partial_x u_{1,0} & h_xh_y\partial_{xy} u_{1,0} & h_x\partial_x u_{1,1} & h_xh_y\partial_{xy} u_{1,1} & \dots & h_x\partial_x u_{1,N_y} & h_xh_y\partial_{xy} u_{1,N_y} \\
\vdots & \vdots & \vdots & \vdots & & \vdots & \vdots \\
h_x\partial_x u_{N_x,0} & h_xh_y\partial_{xy} u_{N_x,0} & h_x\partial_x u_{N_x,1} & h_xh_y\partial_{xy} u_{N_x,1} & \dots & h_x\partial_x u_{N_x,N_y} & h_xh_y\partial_{xy} u_{N_x,N_y}
\end{pmatrix},$$

where $u_{i,j}$ represents $u(x_i, y_j)$.

2.4.2 Collocation Matrices

In order to approximate (2.13), we set up orthogonal spline collocation matrices for (2.19) and (2.20).

We let $\hat{\xi^x}$ be the $1 \times 2N_x$ vector of the Gauss points in the $x-$direction and $\hat{\xi^y}$ be the $1 \times 2N_y$

vector of Gauss points in the $y-$direction given by

$$\hat{\xi^x} = \{\xi^x_{11}\ \xi^x_{12}\ \xi^x_{21}\ \xi^x_{22}\ \dots\ \xi^x_{N_x,1}\ \xi^x_{N_x,2}\}, \tag{2.43}$$

$$\hat{\xi^y} = \{\xi^y_{11}\ \xi^y_{12}\ \xi^y_{21}\ \xi^y_{22}\ \dots\ \xi^y_{N_y,1}\ \xi^y_{N_y,2}\}. \tag{2.44}$$

Again, we begin by utilizing the linear case to demonstrate the general method. By approximating $\mathcal{L}^2_x \approx \delta_x$ and $\mathcal{L}^2_y \approx \delta_y$ and plugging in the form of $u_h(x,y)$ from (2.42), we have that

$$\begin{aligned}\mathcal{L}^2_x u_h + \mathcal{L}^2_y u_h &= \sum_{i=1}^{2N_x}\sum_{j=1}^{2N_y} b_{i,j}[\mathcal{L}^2_x(\phi_i)](\hat{\xi}^x_i)\psi_j(\hat{\xi}^y_j) + \sum_{i=1}^{2N_x}\sum_{j=1}^{2N_y} b_{i,j}\phi_i(\hat{\xi}^x_i)[\mathcal{L}^2_y(\psi_j)](\hat{\xi}^y_j) \\ &= \sum_{i=1}^{2N_x}\sum_{j=1}^{2N_y} b_{i,j}\phi''_i(\hat{\xi}^x_i)\psi_j(\hat{\xi}^y_j) + \sum_{i=1}^{2N_x}\sum_{j=1}^{2N_y} b_{i,j}\phi_i(\hat{\xi}^x_i)\psi''_j(\hat{\xi}^y_j), \end{aligned} \tag{2.45}$$

where $\mathcal{L}_x$ and $\mathcal{L}_y$ are orthogonal spline collocation matrices which approximate the second derivative in the $x-$ and $y-$ directions, respectively. We have that $\hat{\xi}^x_i$ is the i^{th} element of $\hat{\xi}^x$ and $\hat{\xi}^y_j$ is the j^{th} element of $\hat{\xi}^y$.

It is important to understand how these collocation matrices work when applied to a row or column of P. We have that

$$\mathcal{L}^2_x u(x,\cdot) \approx u_{xx}(\xi^x,\cdot),$$
$$\mathcal{L}^2_y u(\cdot,y) \approx u_{yy}(\cdot,\xi^y).$$

When we write $u(x,\cdot)$ we mean the approximation u at the equidistant nodes in the $x-$dimension, $x_i \in d_x$. However, these values must be a concatonation of function values and $x-$derivatives at $x_i \in d_x$, taking on the form of a column of the matrix P. Similarly, $u(\cdot,y)$ denotes the approximation u at the equidistant nodes in the $y-$dimension, $y_j \in d_y$. These values must be arranged as alternating function values and $y-$derivatives, taking the form of a row of the matrix P. The notation $u(\xi^x,\cdot)$ denotes the approximation at the Gauss points in the $x-$dimension, $\xi^x_{ij} \in \hat{\xi}^x$, while

$u(\cdot, \xi^y)$ is the approximation at the Gauss points in the $y-$dimension, $\xi_{ij}^y \in \hat{\xi}^y$. By placing a "$\cdot$" in either dimension, we mean that u is approximated at either the equidistant nodes or the Gauss nodes in the proper dimension, but the choice remains consistent for each individual operation.

We have that $\mathcal{L}_y^2$ is applied to the transpose of each row of P which is of the form

$$P(i,\cdot)^T = \begin{pmatrix} u_{i,0} & h_y\partial_y u_{i,0} & u_{i,1} & h_y\partial_y u_{i,1} & \dots & u_{i,N_y} & h_y\partial_y u_{i,N_y} \end{pmatrix}^T,$$

for $i = 0, \dots, N_x$.

We set up the $2(N_y+1) \times 2(N_y+1)$ collocation square matrix $\mathcal{L}_y^2$ in the following way according to [25]

$$\mathcal{L}_y^2 = \begin{pmatrix} [W_0^y]^{(2)} & 0 & \dots & & 0 \\ [W_1^y & Z_1^y]^{(2)} & & & \vdots \\ 0 & [W_2^y & Z_2^y]^{(2)} & & \\ \vdots & & \ddots & \ddots & 0 \\ & & & [W_{N_y}^y & Z_{N_y}^y]^{(2)} \\ 0 & \dots & & 0 & [Z_{N_y+1}^y]^{(2)} \end{pmatrix},$$

where each $[W_j^y \ Z_j^y]$ is a 2×4 matrix for $j = 1, \dots, N_y$ whose components are determined by the bases $\{\phi\}$ and $\{\psi\}$. For the bases in (2.40) and (2.41), the 2×4 matrices $[W_j^y \ Z_j^y]$ are

$$[W_j^y \ Z_j^y]^{(2)} = \begin{bmatrix} \phi''_{j-1}(\xi_{j1}) & \psi''_{j-1}(\xi_{j1}) & \phi''_j(\xi_{j1}) & \psi''_j(\xi_{j1}) \\ \phi''_{j-1}(\xi_{j2}) & \psi''_{j-1}(\xi_{j2}) & \phi''_j(\xi_{j2}) & \psi''_j(\xi_{j2}) \end{bmatrix}$$

$$= \begin{bmatrix} v''_{j-1}(\xi_{j1}) & s''_{j-1}(\xi_{j1}) & v''_j(\xi_{j1}) & s''_j(\xi_{j1}) \\ v''_{j-1}(\xi_{j2}) & s''_{j-1}(\xi_{j2}) & v''_{j-1}(\xi_{j2}) & s''_j(\xi_{j2}) \end{bmatrix},$$

for $j = 1, \ldots, N_y$. To satisfy the boundary conditions of our problem, we let

$$[W_0^y]^{(2)} = \begin{bmatrix} \nu_1^y/h & \mu_1^y \end{bmatrix}, \quad [Z_{N_y+1}^y]^{(2)} = \begin{bmatrix} \nu_2^y/h & \mu_2^y \end{bmatrix}.$$

According to explicit formulas for the value and scaled slope basis functions evaluated at the Gauss points, found in [25],

$$v_{i-1}''(\xi_{i1}) = -1/h^2 \cdot 2\sqrt{3} \quad , \quad v_{i-1}''(\xi_{i2}) = 1/h^2 \cdot 2\sqrt{3} \tag{2.46}$$

$$v_i''(\xi_{i1}) = 1/h^2 \cdot 2\sqrt{3} \quad , \quad v_i''(\xi_{i2}) = -1/h^2 \cdot 2\sqrt{3} \tag{2.47}$$

$$s_{i-1}''(\xi_{i1}) = -1/h^2 \cdot (1+\sqrt{3}) \quad , \quad s_{i-1}''(\xi_{i2}) = 1/h^2 \cdot (\sqrt{3}-1) \tag{2.48}$$

$$s_i''(\xi_{i1}) = -1/h^2 \cdot (\sqrt{3}-1) \quad , \quad s_i''(\xi_{i2}) = 1/h^2 \cdot (1+\sqrt{3}). \tag{2.49}$$

We have that $\mathcal{L}_x^2$ is applied to each column of P which is of the form

$$P(\cdot, j) = \begin{pmatrix} u_{0,j} & u_{1,j} & \ldots & u_{N_x,j} & h_x\partial_x u_{0,j} & h_x\partial_x u_{1,j} & \ldots & h_x\partial_x u_{N_x,j} \end{pmatrix}^T,$$

for $j = 0, \ldots, 2 * N_y + 1$.

We set up the $2(N_x+1) \times 2(N_x+1)$ collocation square matrix $\mathcal{L}_x^2$ in the following way:

$$\mathcal{L}_x^2 = \begin{pmatrix} \mu_1^x & 0 & \ldots & & 0 & \nu_1^x/h_x & 0 & \ldots & & 0 \\ W_1^x & & & & & Z_1^x & & & & \vdots \\ 0 & W_2^x & & & & & Z_2^x & & & \\ \vdots & & \ddots & & & & & \ddots & & \\ & & & W_{N_x-1}^x & & & & & Z_{N_x-1} & 0 \\ & & & & W_{N_x}^x & & & & & Z_{N_x} \\ 0 & \ldots & & 0 & \mu_2^x & 0 & \ldots & & 0 & \nu_2^x/h_x \end{pmatrix},$$

where each $[W_i^x]$ and $[Z_i^x]$ are 2×2 matrices whose components are determined by the bases $\{\phi\}$ and $\{\psi\}$, respectively. For the bases in (2.40) and (2.41), the matrices $[W_i^x]$ and $[Z_i^x]$ are given by

$$[W_i^x] = \begin{bmatrix} \phi''_{i-1}(\xi_{i1}) & \phi''_i(\xi_{i1}) \\ \phi''_{i-1}(\xi_{i2}) & \phi''_i(\xi_{i2}) \end{bmatrix} = \begin{bmatrix} v''_{i-1}(\xi_{i1}) & v''_i(\xi_{i1}) \\ v''_{i-1}(\xi_{i2}) & v''_i(\xi_{i2}) \end{bmatrix},$$

$$[Z_i^x] = \begin{bmatrix} \psi''_{i-1}(\xi_{i1}) & \psi''_i(\xi_{i1}) \\ \psi''_{i-1}(\xi_{i2}) & \psi''_i(\xi_{i2}) \end{bmatrix} = \begin{bmatrix} s''_{i-1}(\xi_{i1}) & s''_i(\xi_{i1}) \\ s''_{i-1}(\xi_{i2}) & s''_i(\xi_{i2}) \end{bmatrix}.$$

In order to implement this method for our problem, we must also construct the collocation matrices for approximating the first derivative in both dimensions. The structure of these first-derivative matrices, $\mathcal{L}_x$ and $\mathcal{L}_y$, are be the same as those for $\mathcal{L}_x^2$ and $\mathcal{L}_y^2$, respectively. However, for $\mathcal{L}_y$, rather than $[W_j^y \ Z_j^y]^{(2)}$, we make use of $[W_j^y \ Z_j^y]^{(1)}$, where

$$[W_j^y \ Z_j^y]^{(1)} = \begin{bmatrix} v'_{j-1}(\xi_{j1}) & s'_{j-1}(\xi_{j1}) & v'_j(\xi_{j1}) & s'_j(\xi_{j1}) \\ v'_{j-1}(\xi_{j2}) & s'_{j-1}(\xi_{j2}) & v'_j(\xi_{j2}) & s'_j(\xi_{j2}) \end{bmatrix},$$

where

$$v'_{i-1}(\xi_{i1}) = -1/h \quad , \quad v'_{i-1}(\xi_{i2}) = 1/h, \tag{2.50}$$

$$v'_i(\xi_{i1}) = -1/h \quad , \quad v'_i(\xi_{i2}) = 1/h, \tag{2.51}$$

$$s'_{i-1}(\xi_{i1}) = 1/h \cdot 1/(2\sqrt{3}) \quad , \quad s'_{i-1}(\xi_{i2}) = -1/h \cdot 1/(2\sqrt{3}), \tag{2.52}$$

$$s'_i(\xi_{i1}) = -1/h \cdot 1/(2\sqrt{3}) \quad , \quad s'_i(\xi_{i2}) = 1/h \cdot 1/(2\sqrt{3}). \tag{2.53}$$

It is also necessary to present an OSC analogue for the identity matrix. That is, we construct

matrices, I^x_{OSC} and I^y_{OSC}, such that

$$\begin{aligned} I^x_{OSC} \cdot u(x,\cdot) &= u(\xi^x,\cdot), \quad \text{for } x \in d_x,\ \xi^x \in \hat{\xi}^x, \\ I^y_{OSC} \cdot u(\cdot,y) &= u(\cdot,\xi^y), \quad \text{for } y \in d_y,\ \xi^y \in \hat{\xi}^y. \end{aligned}$$

Once again, these identity matrices have the same structure as $\mathcal{L}^2_x$ and $\mathcal{L}^2_y$, using $[W^y_j\ Z^y_j]^{(0)}$ rather than $[W^y_j\ Z^y_j]^{(2)}$. These block submatrices are built using

$$[W^y_j\ Z^y_j]^{(0)} = \begin{bmatrix} v_{j-1}(\xi_{j1}) & s_{j-1}(\xi_{j1}) & v_j(\xi_{j1}) & s_j(\xi_{j1}) \\ v_{j-1}(\xi_{j2}) & s_{j-1}(\xi_{j2}) & v_{j-1}(\xi_{j2}) & s_j(\xi_{j2}) \end{bmatrix},$$

where

$$v_{i-1}(\xi_{i1}) = (9+4\sqrt{3})/18 \quad , \quad v_{i-1}(\xi_{i2}) = (9-4\sqrt{3})/18, \tag{2.54}$$

$$v_i(\xi_{i1}) = (9-4\sqrt{3})/18 \quad , \quad v_i(\xi_{i2}) = (9+4\sqrt{3})/18, \tag{2.55}$$

$$s_{i-1}(\xi_{i1}) = (3+\sqrt{3})/36 \quad , \quad s_{i-1}(\xi_{i2}) = -(3-\sqrt{3})/36, \tag{2.56}$$

$$s_i(\xi_{i1}) = (3-\sqrt{3})/36 \quad , \quad s_i(\xi_{i2}) = -(3+\sqrt{3})/36. \tag{2.57}$$

2.4.3 Domain Decomposition

Since the majority of the change in our solution over time occurs near $r = 0$, it is highly beneficial to take advantage of a non-overlapping domain decomposition. In this way, we are able to choose a fine grid spacing scheme near $r = 0$, where the solution is the steepest, while choosing a coarse grid spacing scheme on the remainder of the domain.

Due to the symmetry in our initial condition, we discuss our method of domain decomposition in one spatial variable and assume that the extension of the decomposition algorithm to the two-

dimensional problem is clear. If we let J be the number of subdomains of I_x, we can begin by defining $\{x_j^*\}_{j=0}^J$ to be a coarse grid partition, not necessarily uniform, of I_x. That is,

$$0 = x_0^* < x_1^* < \ldots < x_{J-1}^* < x_J^* = 1.$$

Let us denote by M_j, for $j = 1, \ldots, J$, the space of Hermite piecewise cubic polynomials on the interval $[x_{j-1}^*, x_j^*]$. We represent the set of Gauss points on the domain I_x that lie in the interval $[x_{j-1}^*, x_j^*]$ by $\mathbb{G}_{\rfloor}{}^x$.

We can define $u_h^{*(j)}$, $v_h^{*(j)}$, and $w_h^{*(j)}$ on each of the J subdomains to satisfy the following problems

$$\begin{aligned} Lu_h^{*(j)}(\xi^x) &= f(\xi^x), \quad \xi^x \in \mathbb{G}_{\rfloor}{}^x, \\ u_h^{*(j)}(x_{j-1}^*) = 0, &\qquad u_h^{*(j)}(x_j^*) = 0, \end{aligned} \tag{2.58}$$

$$\begin{aligned} Lv_h^{*(j)}(\xi^x) &= 0, \quad \xi^x \in \mathbb{G}_{\rfloor}{}^x, \\ v_h^{*(j)}(x_{j-1}^*) = 1, &\qquad v_h^{*(j)}(x_j^*) = 0, \end{aligned} \tag{2.59}$$

$$\begin{aligned} Lw_h^{*(j)}(\xi^x) &= 0, \quad \xi^x \in \mathbb{G}_{\rfloor}{}^x, \\ w_h^{*(j)}(x_{j-1}^*) = 0, &\qquad w_h^{*(j)}(x_j^*) = 1. \end{aligned} \tag{2.60}$$

Then the solution $u_h(x)$ on the entire interval $[x_{j-1}^*, x_j^*]$ can be defined by

$$u_h(x) = u_h^{*(j)}(x) + u_h(x_{j-1}^*)v_h^{*(j)}(x) + u_h(x_j^*)w_h^{*(j)}(x), \tag{2.61}$$

for $j = 1, \ldots, J$, where $f(\xi^x)$ is the exact second derivative of u at the Gauss points in $\hat{\xi}^x$.

The problem (2.58) solves for the interior points on the j^{th} subdomain $[x^*_{j-1}, x^*_j]$, while (2.59) accounts for the left endpoint of the subdomain and (2.60) accounts for the right endpoint of the subdomain. To satisfy the condition that the solution u_h is continuously differentiable on its entire domain, the following condition must hold

$$u'_h(x^{*+}_j) = u'_h(x^{*-}_j), \qquad \text{for } j = 1, \dots J-1.$$

That is, there must be continuity across neighboring endpoints of each pair of subdomains. Now, we must solve for the endpoints of each subdomain, $u_h(x^*_{j-1})$ and $u_h(x^*_j)$. We solve the tridiagonal system

$$TU^*_h = \eta^*$$

to solve for these subdomain endpoints. By using the continuity condition (2.4) and the form of the solution (2.61) we build the matrix T to be

$$\begin{pmatrix} w_h'^{(1)}(x_1^*) - v_h'^{(2)}(x_1^*) & -w_h'^{(2)}(x_1^*) & & & \\ v_h'^{(2)}(x_2^*) & w_h'^{(2)}(x_2^*) - v_h'^{(3)}(x_2^*) & -w_h'^{(3)}(x_2^*) & & \\ \ddots & \ddots & \ddots & & \\ & v_h'^{(J-2)}(x^*_{J-2}) & w_h'^{(J-2)}(x^*_{J-2}) - v_h'^{(J-1)}(x^*_{J-2}) & -w_h'^{(J-1)}(x^*_{J-2}) & \\ & & v_h'^{(J-1)}(x^*_{J-1}) & w_h'^{(J-1)}(x^*_{J-1}) - v_h'^{(J)}(x^*_{J-1}) \end{pmatrix}.$$

The vector

$$U^*_h = \Big(u_h(x^*_1), u_h(x^*_2), \dots, u_h(x^*_{J-2}), u_h(x^*_{J-1})\Big)^T$$

stores the endpoints of each subdomain and

$$\eta^* = \begin{pmatrix} u_h'^{(2)}(x_1^*) - u_h'^{(1)}(x_1^*) \\ u_h'^{(3)}(x_2^*) - u_h'^{(2)}(x_2^*) \\ \vdots \\ u_h'^{(J-1)}(x_{J-2}^*) - u_h'^{(J-2)}(x_{J-2}^*) \\ u_h'^{(J)}(x_{J-1}^*) - u_h'^{(J-1)}(x_{J-1}^*) \end{pmatrix}$$

stores the difference between neighboring endpoints of each pair of subdomains.

To show the effectiveness of implementing domain decomposition, we conduct a short experiment approximating the function u for

$$\mathcal{L}_x^2 u(x) = u''(\xi^x), \tag{2.62}$$

where $u''(\xi^x)$ is known explicitly. We will be approximating the Guassian initial condition used for the lab case on the domain $[0, 1]$ centered at $x = 0.5$. We first approximate the solution to (2.62) with 20 nodes on an uniformly spaced mesh for the domain $[0, 1]$. We then split the domain into five equal subdomains and approximate the solution. We refer to these subdomains as $d_1 = [0, .2]$, $d_2 = [.2, .4]$, $d_3 = [.4, .6]$, $d_4 = [.6, .8]$, and $d_5 = [.8, 1]$, where $[0, 1]$ is made up of the union of $d_1, \ldots, d_5$. Using domain decomposition, we can choose the number of nodes to place in each subdomain. Since the majority of the steepness occurs in the center of the domain, we assign the 20 nodes in the following way: one node in the outer subdomains d_1 and d_5 where the function is relatively flat, five nodes in subdomains d_2 and d_4, and eight nodes in the center subdomain d_3 where the function is the steepest.

We begin by displaying the three different node-spacing schemes. Figure (2.3) plots

- 20 equidistant nodes on a domain with no subdomains,
- 20 equidistant nodes placed into subdomains $d_1, \ldots d_5$ using the described placement scheme

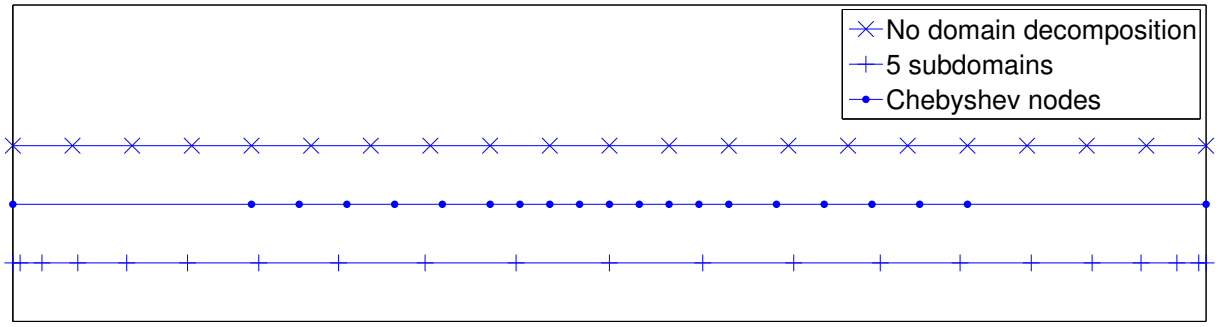


Figure 2.3: Spacing of three different spatial schemes.

- 20 Chebyshev nodes given by (2.30).

We can see clearly that the spacing with five subdomains clusters the nodes in the center of the domain, while the Chebyshev nodes are clustered near the boundary.

In Figure (2.4), we plot the exact solution and the OSC approximation using a uniform mesh as well as with domain decomposition. We include an approximation using the Chebyshev Spectral method for comparison. Since OSC takes advantage of $2(N+1)$ gauss nodes on the spatial domain, we choose $2(N+1)$ Chebyshev nodes to make an accurate comparison.

While both OSC approximations use the same amount of nodes, we can see in Figure (2.5) that the error using this five-subdomain decompostion is much lower than that for the equidistant nodes. Furthermore, the OSC approximation using domain decomposition has less error than the higher-order Chebyshev Spectral method. We see that the maximum relative error for the OSC uniform mesh is approximately 0.0273, while the maximum relative error using OSC subdomains is 0.0018. The maximum relative error for the Chebyshev Spectral method is 0.0039. It is of particular importance to point out that the error for OSC with domain decomposition is much more evenly distributed throughout the domain than the other two methods.

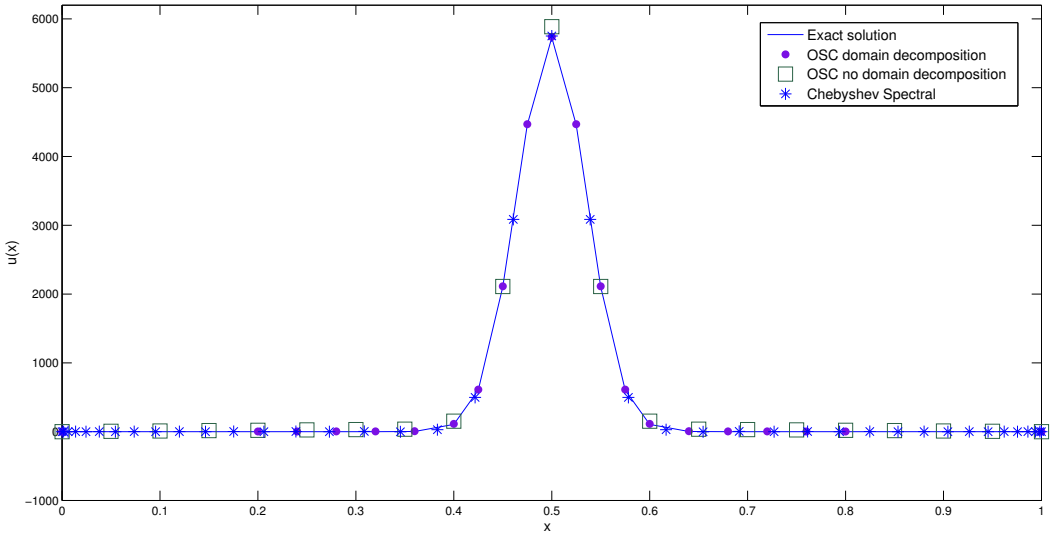


Figure 2.4: Approximations to (2.62) using OSC on a uniform mesh, OSC with domain decomposition, and the Spectral method on Chebyshev nodes.

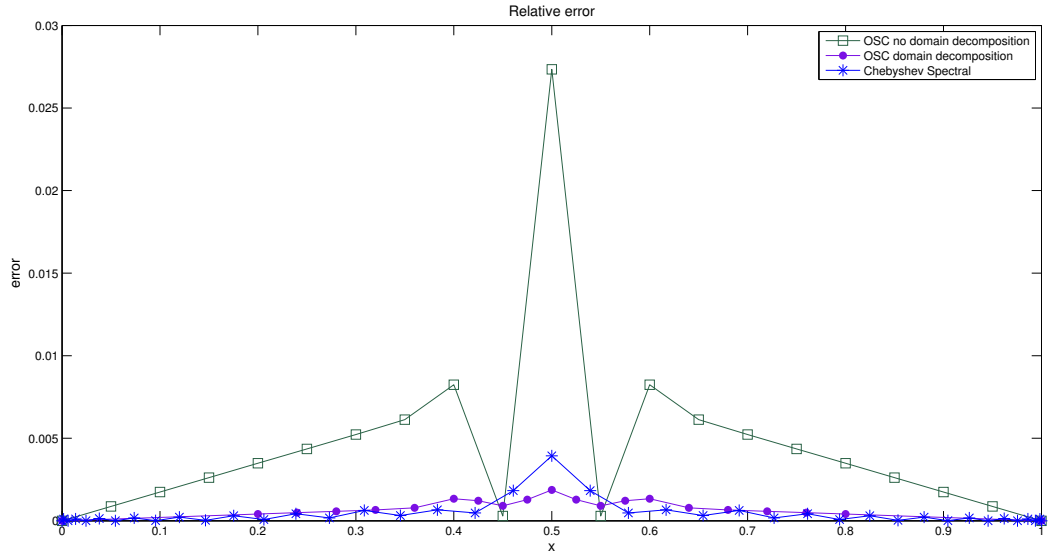


Figure 2.5: Comparison of error in approximations to (2.62) using OSC on a uniform mesh, OSC with domain decomposition, and the Spectral method on Chebyshev nodes.

For this particular experiment with $N = 20$, five subdomains was optimal. However, when the domain, the number of nodes, and the function changes it is suggested that experiments be run to determine the optimal number of subdomains.

CHAPTER 3: ERROR AND EFFICIENCY

3.1 Spatial Approximation of Derivatives

Numerically solving (2.13) requires the spatial approximation of first and second derivatives. We run experiments to compare the approximation of the second derivative of the initial condition. We claim that this is important because we generally run short-time experiments which means we do not expect the approximations to change much from the initial condition. Furthermore, due to the diffusive nature of our problem, the approximation is the steepest at the initial condition.

As presented in Sections 2.2-2.4, we have set up three spatial discretization methods for our problem. When approximating the second derivative of a Gaussian, all three methods generally attain increasingly accurate results when the number of nodes, N, is increased. We show this in Figure (3.1) by calculating the relative error using

$$\text{error} = \frac{1}{\|p_{xx}\|_\infty} \|p_{xx} - D^2 p\|_\infty, \tag{3.1}$$

where the differentiation matrix D^2 approximates the second derivative using Finite-Differences, Orthogonal Spline Collocation, or a Chebyshev Spectral method. Even when approximating derivatives of a non-steep Gaussian, we see large values in the second derivative. For example, when Thrust$= 1$ and $\sigma = 0.3$, the maximum value of the second derivative of the Gaussian reaches over 560. It is not feasible to calculate relative error component-wise since the exact second derivative contains zero values. However, it can be shown that the maximum error between the exact and approximate second derivatives occurs when the exact second derivative is the greatest. This can be seen in Figure (2.5) from Section 2.3. So, by dividing by $\|p_{xx}\|_\infty$ in (3.1), we achieve a relative error that is extremely useful when analyzing steeper cases with larger values.

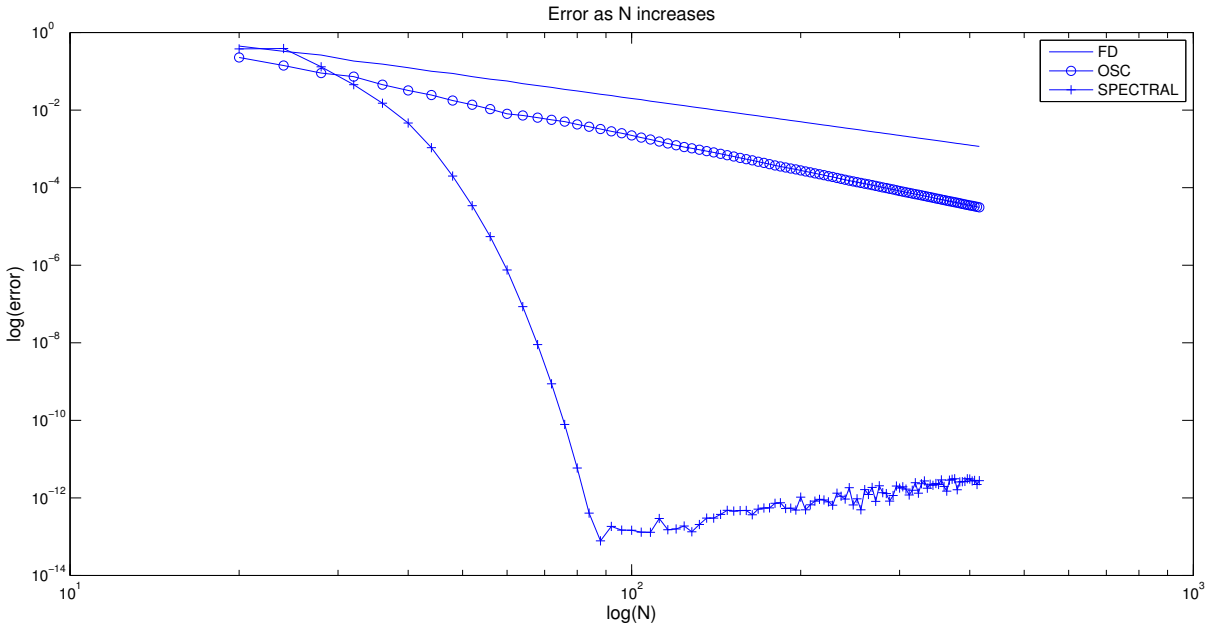


Figure 3.1: Approximation of the second derivative of the 1D Gaussian initial condition, $p_0 = Ee^{(-x^2/\sigma^2)}$, with Thrust $= 1$, $\sigma = 0.3$.

By plotting both the error and the number of nodes using a log scale, we see that the Finite-Difference method has a slope of 2 which is consistent with its second-order convergence property. We also see that OSC has a slope of 3. This convergence rate is consistent with similar experiments run in [2]. We expect to see a convergence of $O(N^{-N})$ for the Spectral method, however, since we are approximating the second derivative with $D^2 = D \cdot D$, we loose some accuracy. This $O(N^{-N})$ accuracy can be acheived for the second derivative approximation by using recurrance formulas to build the Chebyshev Spectral second derivative matrix found in [23].

It is overwhelmingly clear from Figure (3.1) that the Chebyshev Spectral method can quickly attain much better accuracy than both the finite-difference method and OSC. With 90 nodes, we can approximate the second derivative of a non-steep Gaussian with nearly 10^{-13} accuracy using the Spectral method, while OSC yields an accuracy of 10^{-3} and the finite-difference yields an accuracy of 10^{-2}. Since the Chebyshev Spectral method uses the Chebyshev nodes that are clustered on the boundaries, we expect this method to perform worse with fewer nodes, in this case when $N < 14$, than the Finite-Difference method and OSC where the nodes are evenly distributed throughout the

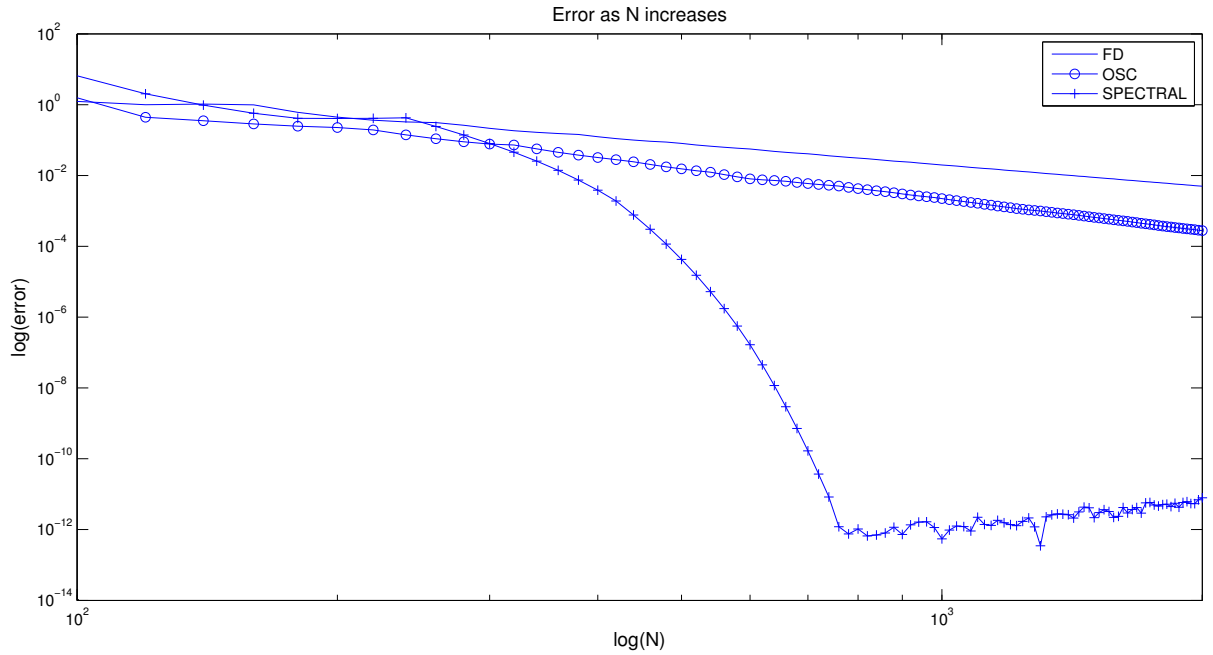


Figure 3.2: Error when approximating the second derivative of the steep 1D Gaussian initial condition, $p_0 = Ee^{(-x^2/\sigma^2)}$, with Thrust $= 45.5$, $\sigma = 0.03$.

domain. This is especially evident for a Gaussian function where the majority of the change occurs in the center of the domain.

In order to run simulations that represent our physical model, it is necessary to approximate these derivatives when the function is much more steep. We show our results for error under these extreme conditions in Figure (3.2).

As expected, both of our methods require more nodes to achieve acceptable accuracy for this approximation. Now, in order to achieve accuracy of 10^{-13} using the Spectral Method, we must use nearly 800 nodes.

3.2 Linear Problem

Now that we have an idea of how well the different spatial discretizations perform for our initial condition, we couple these methods with our ADI time-stepping scheme to simulate time-

evolution. In order to analyze and compare the ability of the different methods, we look at their ability to approximate the solution to the linear problem for which we have an exact solution. That is, we seek approximations to the linear 2D heat equation problem in Cartesian coordinates with homogeneous Dirichlet boundary conditions

$$\begin{aligned} &u_t = u_{xx} + u_{yy}, \\ &u(x_0, y, t) = 0, u(x, y_0, t) = 0, \\ &u(x_{N_x}, y, t) = 0, u(x, y_{N_y}, t) = 0, \\ &u(x, y, t_0) = E \cos\left(\frac{\pi}{2}x\right) \cos\left(\frac{\pi}{2}y\right), \end{aligned} \tag{3.2}$$

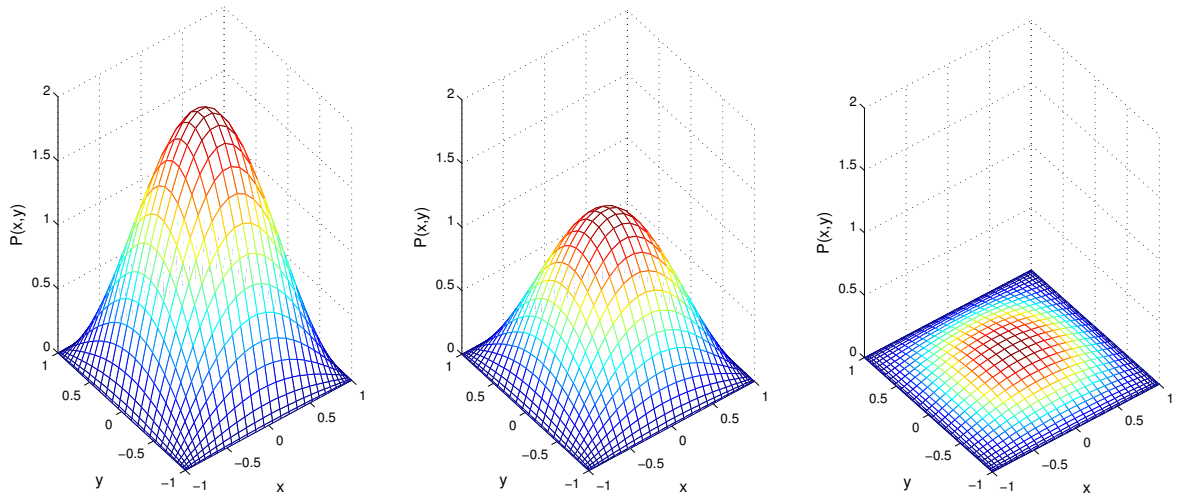
where the amplitude E is calculated the same as for the Gaussian initial condition given by (1.6). We choose this initial condition so that our problem is symmetric about $(x, y) = (0, 0)$ and has zero values on the boundaries for the domain $[-1, 1] \times [-1, 1]$. This problem is well studied and the exact solution is easily derived to be

$$u(x, y, t) = E \cos\left(\frac{\pi}{2}x\right) \cos\left(\frac{\pi}{2}y\right) e^{-\frac{\pi^2}{2}t}. \tag{3.3}$$

We compare approximations to the solution of (3.2) using the following methods

1. Backward in time, FD in space (BT-FD)
2. ADI in time, FD in space (ADI-FD)
3. ADI in time, OSC in space (ADI-OSC)
4. ADI in time, Chebyshev Spectral in space (ADI-Spectral).

First we present the behavior of the diffusion in the approximation using the ADI-Spectral method. Figure (3.3) shows the evolution of the approximation to (3.2) over time.



(a) Initial Condition (b) After 100 timesteps (c) After 500 timesteps

Figure 3.3: ADI-Spectral Approximation of (3.2) with $E = 2$, $N_x = N_y = 31$, $dt = 0.001$, $T_f = 0.5$.

We now compare all four methods' ability to approximate the solution to (3.2) as a function of the number of spatial nodes, N_x and N_y. We run the simulation using each of the four methods for increasing values of N_x and N_y and calculate the error using (3.1). We illustrate this error as a function of N_x in Figure (3.4).

Since the BT-FD and ADI-FD methods use the same spatial approximation, we see that the error using both methods overlap. It is clear from Figure (3.4) that the ADI-Spectral method performs the best for this type of problem. We see that with just 14 nodes, the ADI-Spectral method solves (3.2) with accuracy of 10^{-12}.

We also show accuracy of our approximation as a function of the time-step size. We show that the ADI method attains second-order convergence with respect to time, while the explicit backward in time method used by [19] has only first-order. This is illustrated in Figure (3.5).

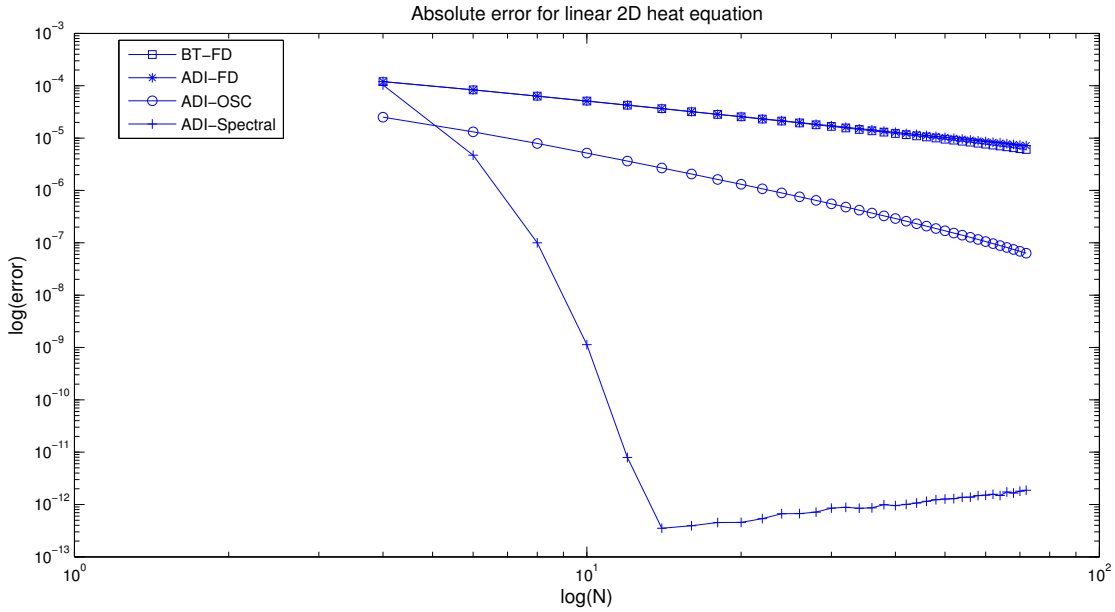


Figure 3.4: Error in approximation to (3.2) for varrying values of N_x and N_y. Parameters used are $dt = 0.0001$, $T_f = 0.002$, E = 2.

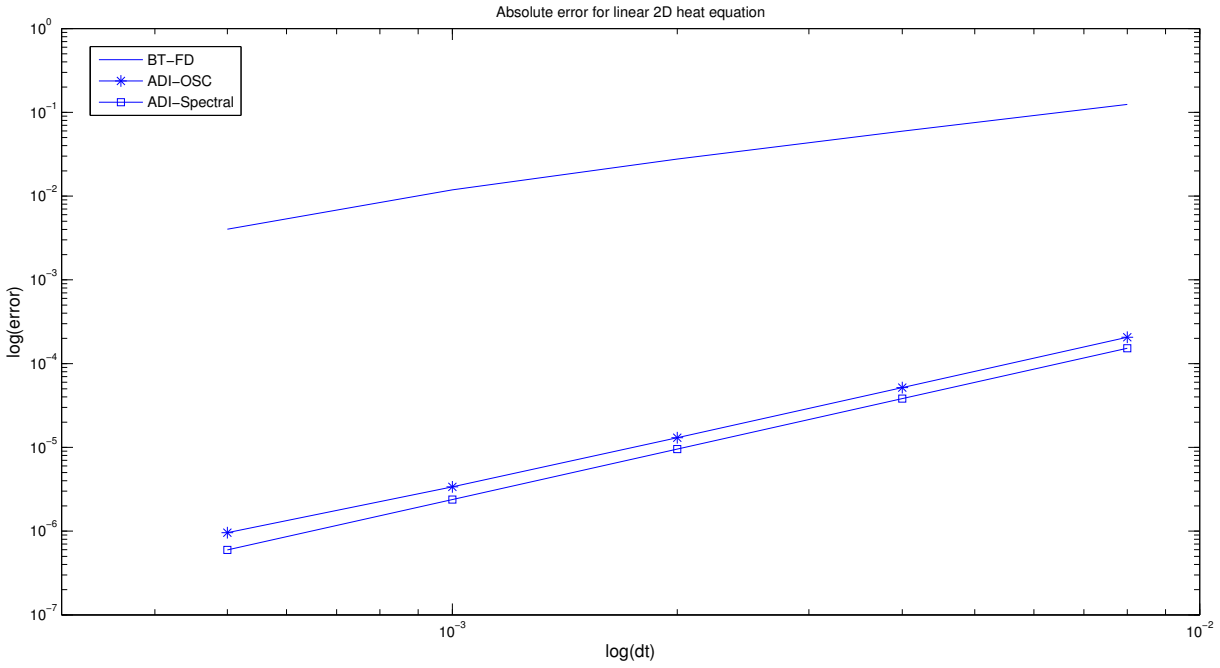


Figure 3.5: Error in approximation to (3.2) for varrying values of dt. Parameters used are $N_x = N_y = 37$, $T_f = 0.032$, E = 2.

Table 3.1: A comparison of error and runtime for the linear problem (3.2).

	Relative Error			Runtime (s)		
N	ADI-FD	ADI-OSC	ADI-Spectral	ADI-FD	ADI-OSC	ADI-Spectral
4	$1.1436 \cdot 10^{-2}$	$4.0363 \cdot 10^{-3}$	$1.061 \cdot 10^{-2}$	0.02764	0.02567	0.02686
8	$6.040 \cdot 10^{-3}$	$7.677 \cdot 10^{-4}$	$4.741 \cdot 10^{-6}$	0.05738	0.05714	0.06092
16	$3.061 \cdot 10^{-3}$	$6.446 \cdot 10^{-5}$	$3.599 \cdot 10^{-9}$	0.14484	0.15152	0.17605
32	$1.535 \cdot 10^{-3}$	$4.219 \cdot 10^{-6}$	$7.199 \cdot 10^{-9}$	0.47324	0.50055	.50713
64	$7.683 \cdot 10^{-4}$	$3.099 \cdot 10^{-7}$	$1.440 \cdot 10^{-8}$	2.6897	3.0644	3.1568
128	$3.842 \cdot 10^{-4}$	$6.136 \cdot 10^{-8}$	$2.880 \cdot 10^{-8}$	23.713	24.711	26.237

Table 3.1 shows a comparison of runtime[1] and accuracy for the ADI-FD, ADI-OSC, and ADI-Spectral methods as a function of the number of spatial nodes, N. For simplicity, we run these experiments on a uniform mesh where $N_x = N_y$. The runtime as a function of N is similar for all three methods, but we can see that we get the best error using the Chebyshev Spectral spatial discretization which is consistent with Figure (3.4). We also see that there is a point at which the error using the Spectral method no longer improves as N increases. This is interesting because while the OSC method does not initially attain as low an error as the Spectral method, it continues to improve as N increases. Furthermore, we see that when $N = 128$, the OSC method actually attains an error on the same order of magnitude as the Spectral method.

3.3 Nonlinear Simultations

Results from approximations of the linear problem (3.2) lead us to believe that the ADI-Spectral method attains the most accuracy and efficiency for approximating the nonlinear problem (2.13). Figure (3.6) shows the ADI-Spectral method's approximation after 1000 timesteps.

[1]In order to get a good estimate of runtime using MatLab, the time was calculated for five experiments and the average time is presented.

We must first point out that while using the Spectral method for spatial discretization provides the best results for both accuracy and efficiency, we see that too large of a time-step causes the exponential growth of small errors [21]. It has been shown empircally in [21] that stability in solving the 2D linear equation (3.2) using the Chebyshev Spectral method requires the following restriction on the timestep

$$\Delta t < 6.6(\Delta x)^2,$$

where we are assuming that $\Delta x \leq \Delta y$.

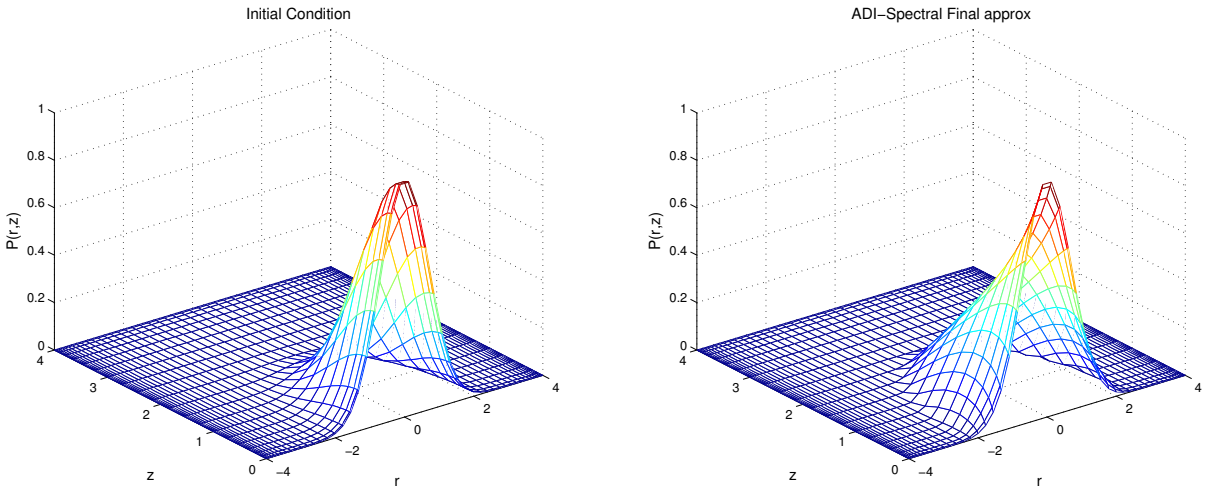


Figure 3.6: ADI-Spectral approximation to the nonlinear problem (2.13). Parameters are $N_r = 51$, $N_z = 21$, $dt = 0.0004$, $T_f = .4$, E = 1, $\sigma = 1$.

In order to make a connection between our results and the realistic experiments, we point out that the boundary $z = 0$ in Figure (3.6) represents the surface of the granular medium, while $z = 4$ represents the bottom. We see that over time the pressure spreads through the regolith while maintaining constant pressure at the top boundary. This constant pressure at $z = 0$ is achieved with the Dirichlet boundary condition. This can be better understood with the help of contour plots displayed in Figure (3.7).

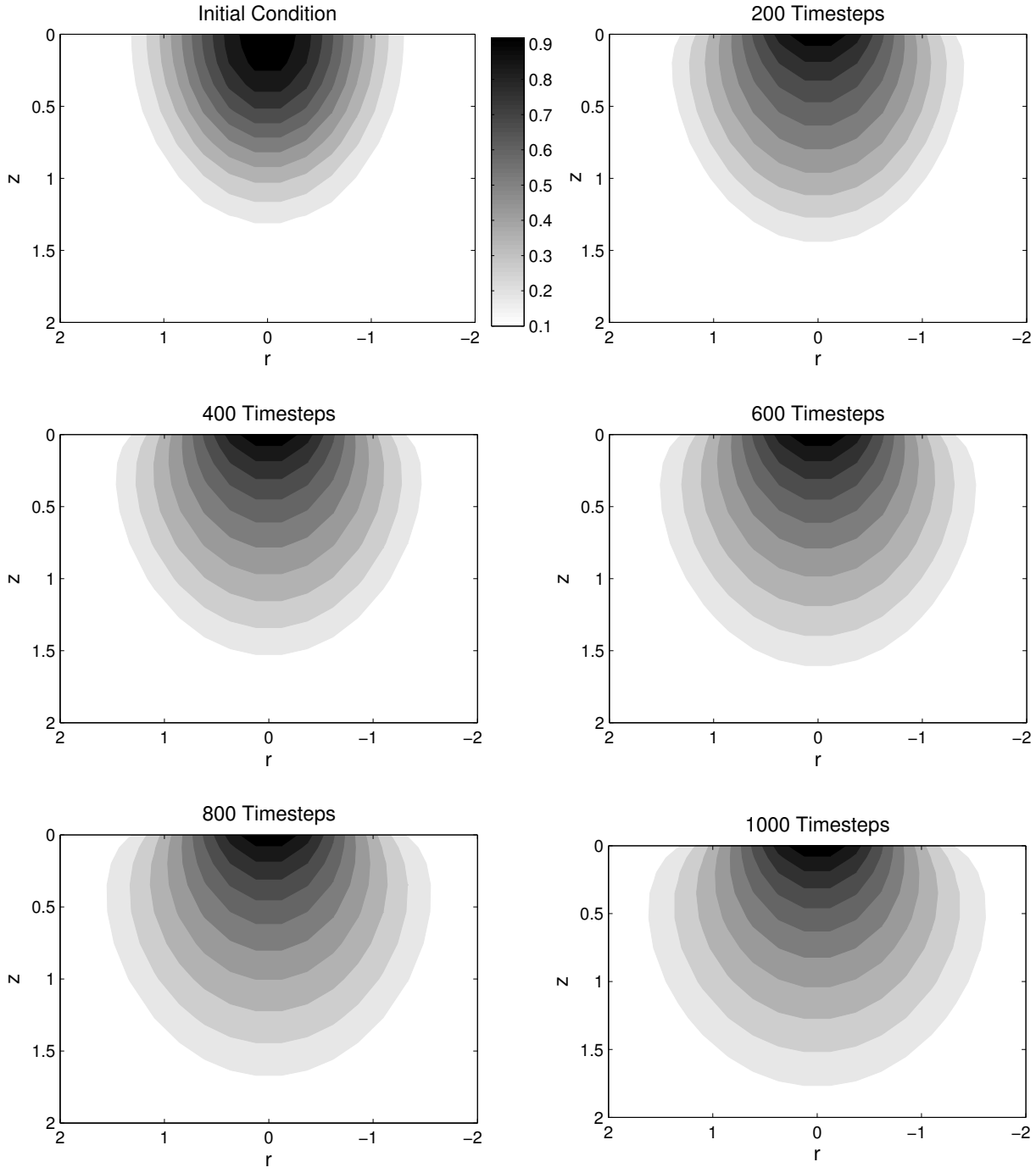


Figure 3.7: ADI-Spectral approximation to the nonlinear problem (2.13) for varrying final times. Parameters are $N_r = 51$, $N_z = 21$, $dt = 0.0004$, E = 1, $\sigma = 1$.

We present error plots in the Section 3.2 to show the order of convergence in time for the linear problem (3.2). However, no extrapolation to account for nonlinear terms was necessary for the approximation of solutions to (3.2). Therefore, we plot the the temporal error for our nonlinear method which requires extrapolation in Figure (3.8). While we have no exact solution to compare with our final approximations, we can make use of the Richardson error estimate

$$u - U(\frac{1}{2}\Delta t) \approx \frac{1}{2^p - 1}(U(\frac{1}{2}\Delta t) - U(\Delta t)), \tag{3.4}$$

where u is the unknown exact solution, and p is the order of the discretization method. Here we have that $U^n(\Delta t) = U(x, y, t_n)$, where $t_n - t_{n-1} = \Delta t$. From (3.4), we can make the comparison

$$\frac{\left\| U^n\left(\Delta t\right) - U^n\left(\frac{1}{2}\Delta t\right) \right\|}{\left\| U^n\left(\frac{1}{2}\Delta t\right) - U^n\left(\frac{1}{4}\Delta t\right) \right\|} \tag{3.5}$$

and see a slope of p for each method when plotted on a loglog scale.

Using this analysis, Figure (3.8) shows the error in time for the BT-FD, ADI-FD, and ADI-Spectral methods. We can see that the BT-FD method is first-order in time and the ADI-FD and ADI-Spectral methods are both second-order. However, since the Spectral Method converges much faster than the Finite Difference method, we have that the Richardson error for the ADI-Spectral method is lower than that of the ADI-FD method.

Figure (3.9) shows the values of the pressure for different layers under the center of the jet nozzle. By $P(0, j)$ we denote the j^{th} layer of pressure directly underneath the point where $r = 0$. Tracking the values at each of these layers over time shows us that the layers closer to the surface reach a steady state very quickly. This characteristic is consistent with experimental results where the formation of a crater happens very quickly and does not undergo much change once it is created. This serves as motivation to run short time simulations.

In Figure (3.10), we display the contour plots for the nonlinear problem using both the ADI-FD

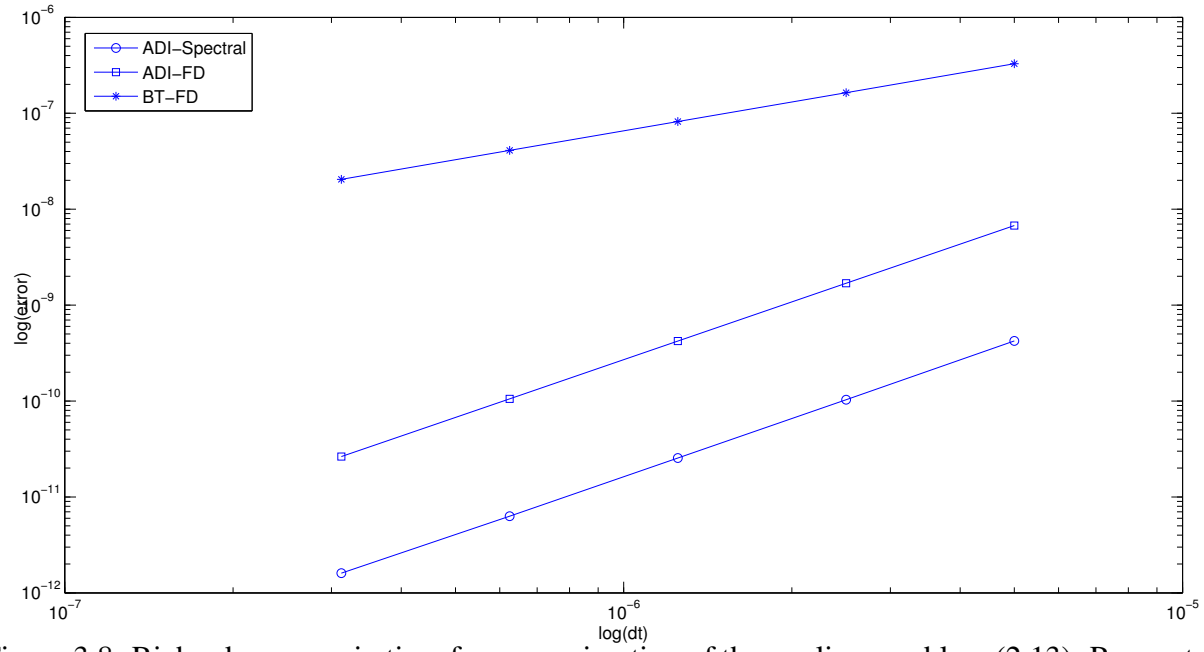


Figure 3.8: Richardson error in time for approximation of the nonliner problem (2.13). Parameters are $T_f = 0.001$, $N_r = 51$, $N_z = 21$, $E = 1$.

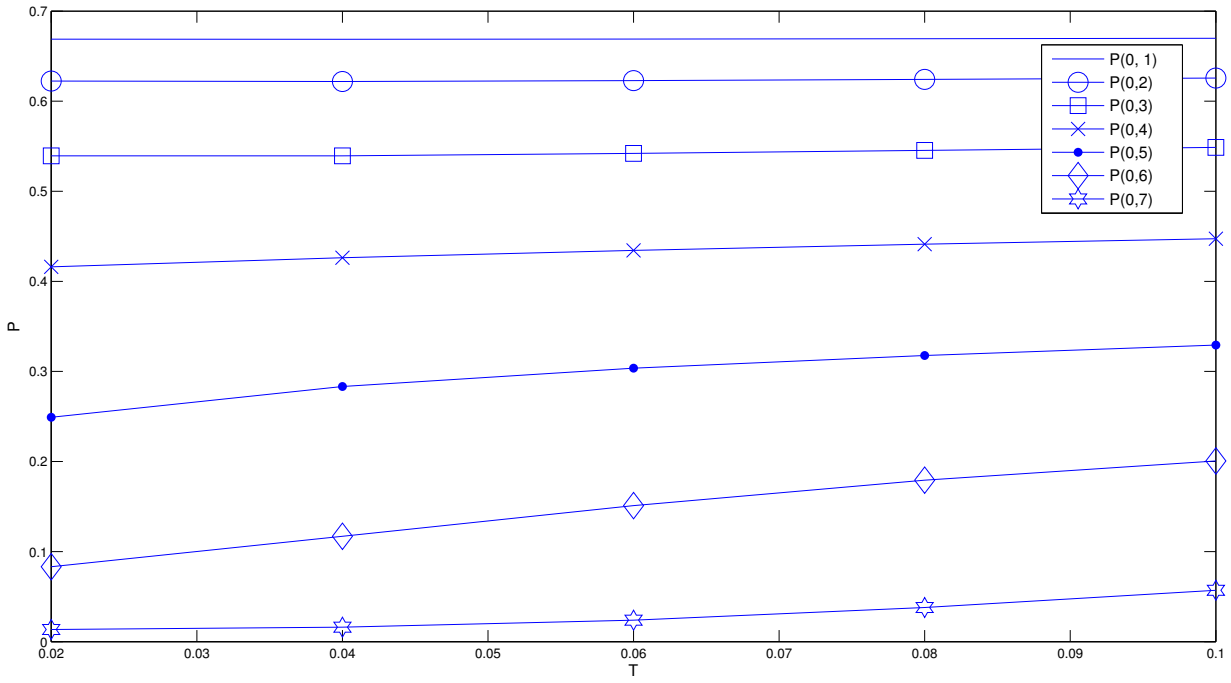


Figure 3.9: Pressure values at varrying layers below the jet nozzle tracked over time.

and ADI-Spectral methods after 1000 timesteps. We note that the two methods compare fairly well except for layers near the boundary where $z = 0$. Since the Chebyshev Spectral method is implemented on the Chebyshev nodes that are clustered near the boundaries, we get a smoother and more accurate approximation at the $z = 0$ boundary. This is displayed in the figure.

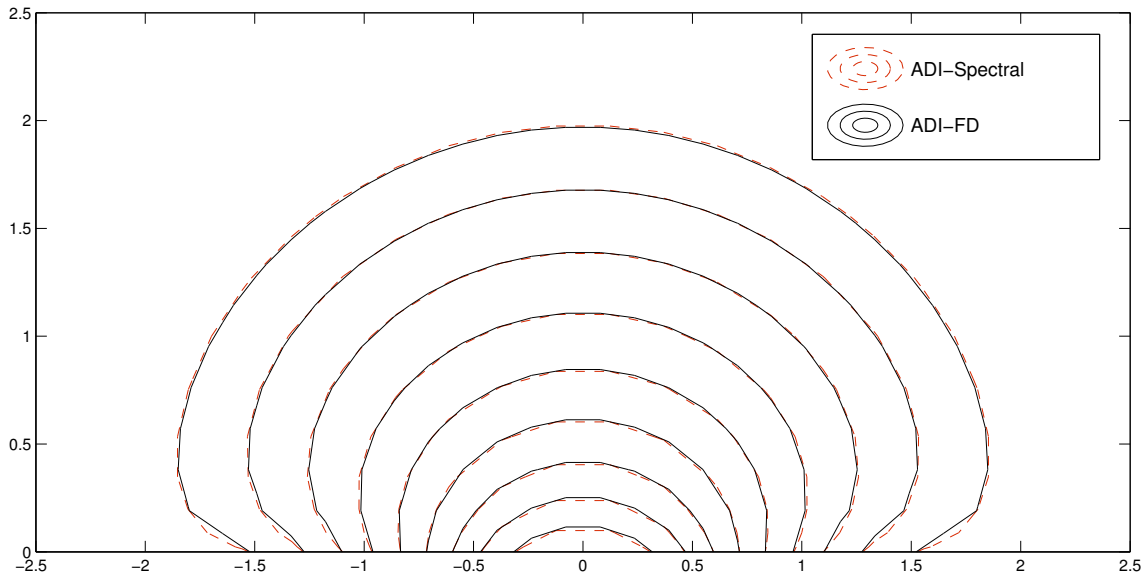


Figure 3.10: Comparison between ADI-FD and ADI-Spectral for the nonlinear problem.

We now wish to see the difference in the approximations for the two experiments presented in Section 1.1. Until now, all simulations have been implemented for the scaled case. Figures (3.11) and (3.12) present different views for the results for both experiments. In Figure (3.11), we show the final approximation of the pressure in the lunar case after 42 seconds and the final approximation of the pressure in the lab case after just 0.00096 seconds. Such a large difference in final times arises from the change of variables presented in Section 1.1. If we let T_N be the final time for our simulations and t_p be the physical time in reality, we have that

$$T_N = \beta t_p,$$

where $\beta = \dfrac{2\epsilon\eta r_0^2}{\kappa p_0}$. According to the values in Table (1.1), we have that

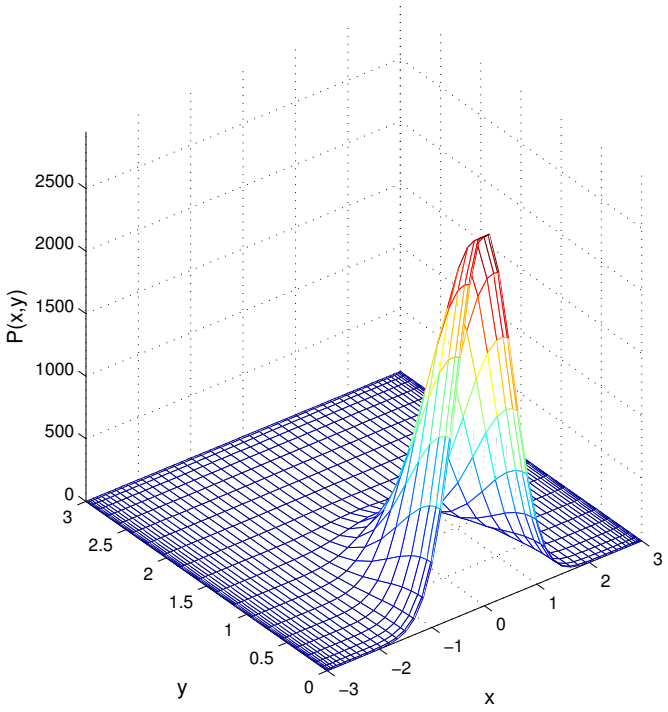
$$\beta_{lab} \approx 4.67 \times 10^{-3},$$

$$\beta_{lunar} \approx 4.16 \times 10^{3},$$

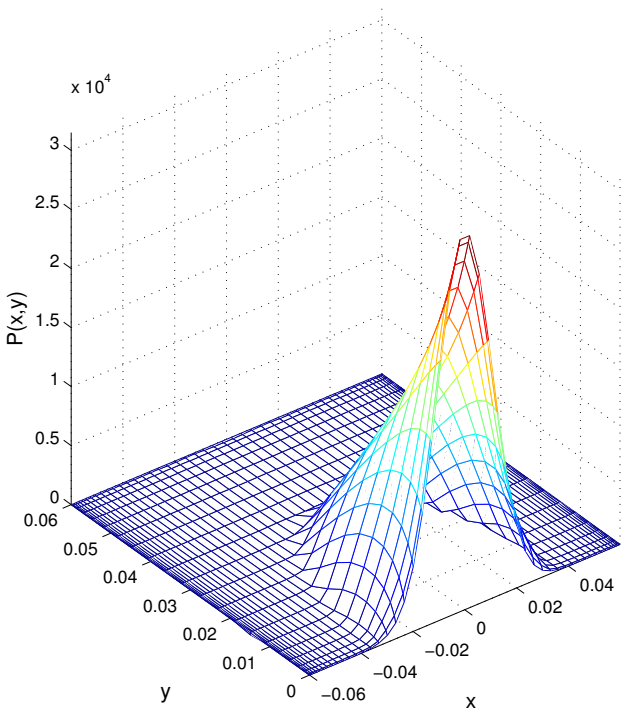
where β_{lab} and β_{lunar} are the values for β according to both the lab case and the lunar case, respectively. For example, if we let $T_N = 0.01$ for the lunar case, we are simulating the approximation after about 42 seconds. In order to approximate the pressure for the lab case after 42 seconds, we would need $T_N = 8907$. With a small time-step size of 0.001, which is necessary to ensure accuracy, we see that this type of experiment is not easily run.

Our main goal is to reproduce the experimental cratering results for both cases. No cratering occured in the lunar case, and a crater was definitely formed in the lab case. We claim that these results are consistent with our simulations, despite the difference in final times. It makes sense that the pressure is not as extreme in the lunar case, even after a longer period of time. Results from this simulation for 42 seconds show very little change in the pressure within the medium. Similarly, by showing such an extreme change in the pressure for the lab case after only a short time, we can easily see how cratering will occur.

Figure (3.12) displays this even more clearly. If we pay close attention to the boundary $z = 0$, which represents the surface of the medium, we can see a very large difference in the behavior of the lunar case when compared to that of the lab case. Since we have shown that a steady state is reached after a short amount of time, see Figure (3.9), we can assume that the results in the lunar case remain smooth and relatively unchanging as time evloves further. Overall, we claim that *these preliminary results are consistent with the experimental results for both cases*.

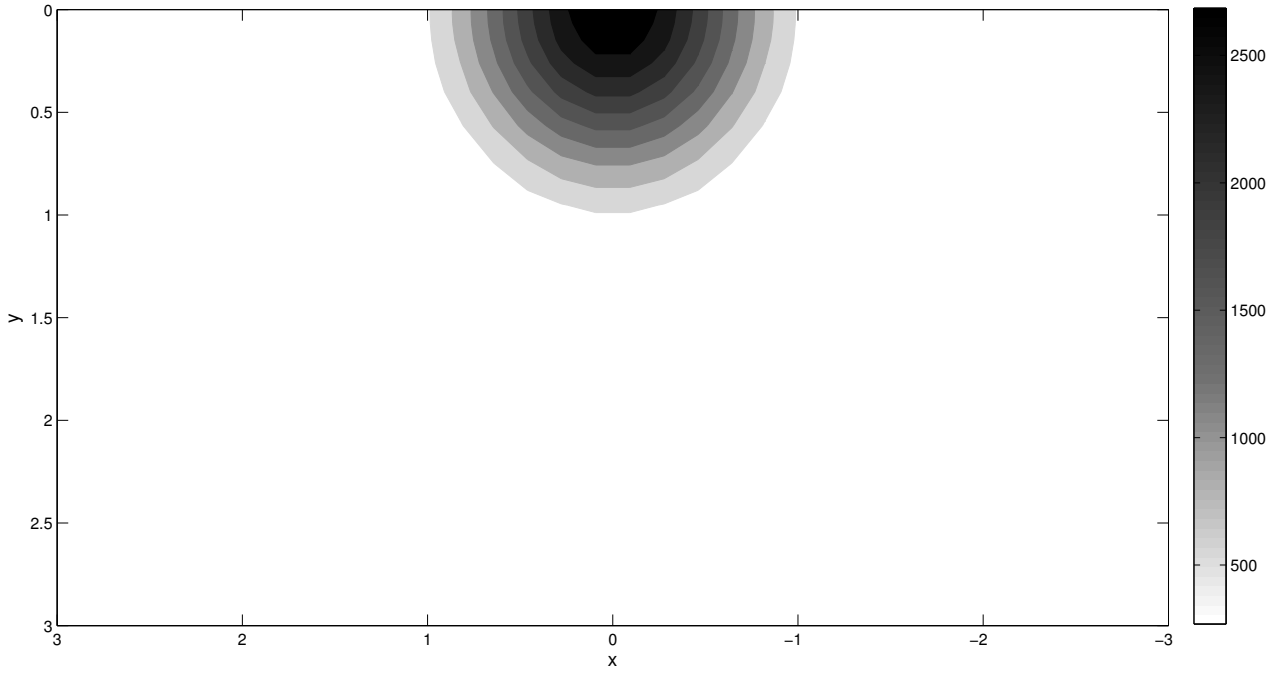


(a) Lunar

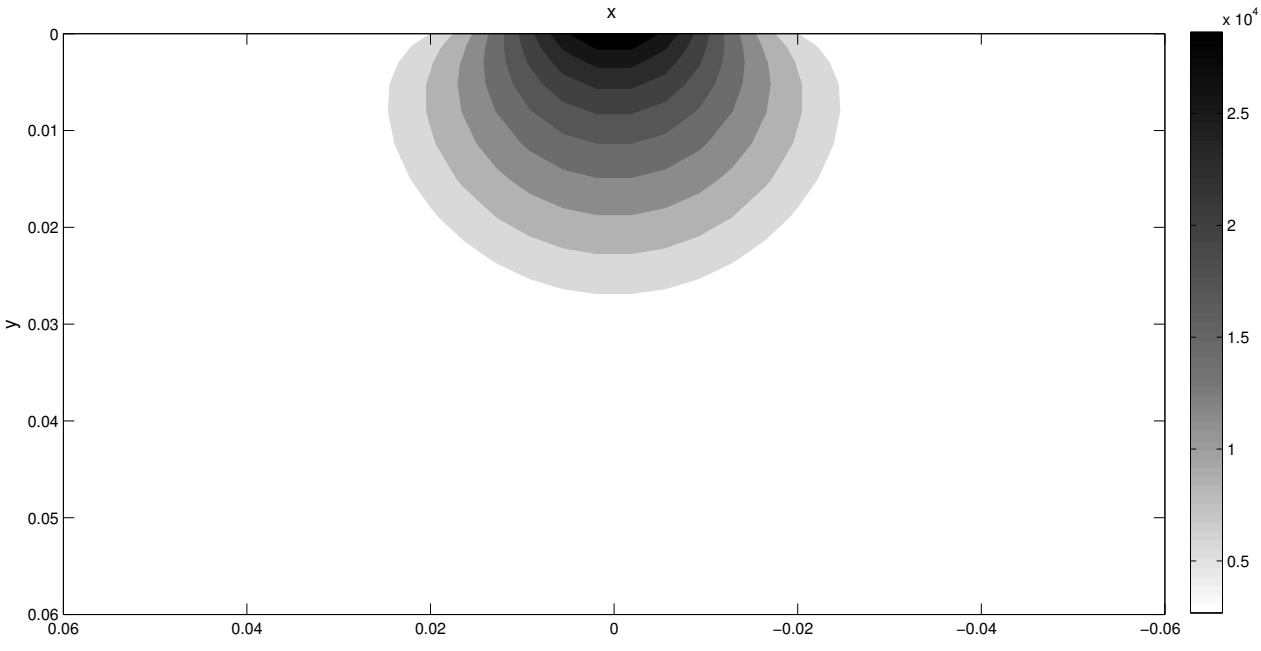


(b) Lab

Figure 3.11: Approximations of pressure for the nonlinear problem (2.13) for both the lunar and lab case.



(a) Lunar



(b) Lab

Figure 3.12: Contour plots for approximations of pressure for the nonlinear problem (2.13) for both the lunar and lab case.

CHAPTER 4: CONCLUSIONS AND FUTURE WORK

4.1 Conclusions

We have presented an Alternating-Direction Implicit temporal scheme for both the linear problem (3.2) and the nonlinear Porous Medium Equation (1.7). We have compared this ADI method with a Backward-Euler scheme to show increased accuracy, and have discussed the advantages of using this type of scheme over a Crank-Nicholson scheme. Results using a Richardson error estimate for this time discretization are presented for both the linear and nonlinear problems in Sections 3.1 and 3.2, which both show the second-order accuracy of the ADI method in time.

We also presented three spatial discretization methods to solve our problem. We presented the second-order Finite Difference method, the fourth-order Orthogonal Spline Collocation Method, and the N^{th}-order Chebyshev Spectral method. Results for all three methods coupled with and ADI time-stepping scheme are compared in Section 3.1 for the linear problem. These results show that the Spectral method attains the best accuracy. We implement this ADI-Spectral method for the nonlinear case and compare it to the ADI-FD method and also the Backward Euler method used by [19].

Experiments were run to simulate the pressure for the case where a crater was formed in a lab at KSC and for the case from a lunar mission where no cratering occurred. Our preliminary simulation results using the ADI-Spectral method agree with the results from these two cases.

4.2 Improved Efficiency

One elegant property of our problem is the presence of symmetry about the point $(0, 0)$. In our simulations, we do not take advantage of this symmetry with respect to r due to a singularity which

appears when $r = 0$. By implementing our problem on the domain $[-1, 1] \times [0, 1]$, we do not need to worry about problems arising from this singularity. However, we can with some ease derive an equation that is consistent with the order of convergence for the spatial discretization of choice to use for the layer where $r = 0$. By using half of the domain in the $r-$ dimension, we can use half as many nodes to attain the same amount of accuracy. On our current domain, the Chebyshev nodes in the $r-$dimension are clustered near the boundaries where the function is relatively flat. By chopping the $r-$domain in half, we are able to take advantage of the clustering nature of the Chebyshev nodes at the boundary for $r = 0$ where the function is the steepest.

This type of technique was derived and implemented in [19] where pressure was simulated over the domain $[0, 1] \times [0, 1]$. They make a second-order approximation to account for the singularity at the axis of symmetry that is consistent with the second-order finite-difference method used in their model. We present the derivation of this approximation to serve as an example for what will be done for higher-order discretizations.

A different approximation for $\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial p}{\partial r}\right)$ must be made to account for $r = 0$. We begin by taking a Taylor series expansion of $p_r(r, z, t)$ about the point $r = 0$ and by assuming that there is no flow of gas across this boundary, that is $\frac{\partial p}{\partial r} = 0$ when $r = r_1$,

$$\begin{aligned} p_r(r, z, t) &= p_r(0, z, t) + r p_{rr}(0, z, t) + \frac{r^2}{2} p_{rrr}(0, z, t) + \dots \\ &= r p_{rr}(0, z, t) + \frac{r^2}{2} p_{rrr}(0, z, t) + \dots, \end{aligned}$$

since $p_r(0, z, t) = 0$. Then we can write

$$\begin{aligned} \frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial p}{\partial r}\right) &= \frac{1}{r}\frac{\partial}{\partial r}\left(r^2 p_{rr}(0, z, t) + \dots\right) \\ &= \frac{1}{r}(2r p_{rr}(0, z, t) + \dots \\ &= 2p_{rr}(0, z, t) + \dots. \end{aligned}$$

Next we consider the the Taylor expansion

$$p(\Delta r, z, t) - p(0, z, t) = \frac{\Delta r^2}{2} p_{rr}(0, z, t) + \dots . \tag{4.1}$$

This implies that

$$p_{rr}(0, z, t) \approx \frac{2}{\Delta r^2}\bigg(p(\Delta r, z, t) - p(0, z, t)\bigg). \tag{4.2}$$

By combining (4.2) with the approximation $\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial p}{\partial r}\right) \approx 2p_{rr}(0, z, t)$, it follows that

$$\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial p}{\partial r}\right) \approx \frac{4}{\Delta r^2}\bigg(p(\Delta r, z, t) - p(0, z, t)\bigg), \tag{4.3}$$

when $r = 0$. Representing this in discretized form, we can write

$$P^{n+1}_{0,j} = P^n_{0,j} + M\left[(P^2_{0,j+1})^n - 2(P^2_{0,j})^n + (P^2_{0,j-1})^n\right] + 4M\left[(P^2_{1,j})^n - (P^2_{0,j})^n\right], \quad \text{for } j = 1, \dots, N_z. \tag{4.4}$$

Since our ADI-FD method is of second order in space, we can implement this second-order treatment of the boundary to approximate the solution to the nonlinear problem on the domain $[0, 1] \times [0, 1]$. Results are shown in Figure (4.1).

4.3 OSC for Nonlinear Problem

The results in this thesis show that the ADI-Spectral method performs better than the ADI-FD and ADI-OSC methods for our problem. However, we showed in Section 2.4.3 that we can achieve as good an approximation with the OSC method if we utilize the domain decomposition algorithm presented. One particular benefit of using OSC over Spectral differentiation is the *almost block diagonal* (ABD) form of the OSC differentiation matrices versus the non-sparse Spectral matrices. We can take advantage of this form by using an ABD solver which will increase efficiency while

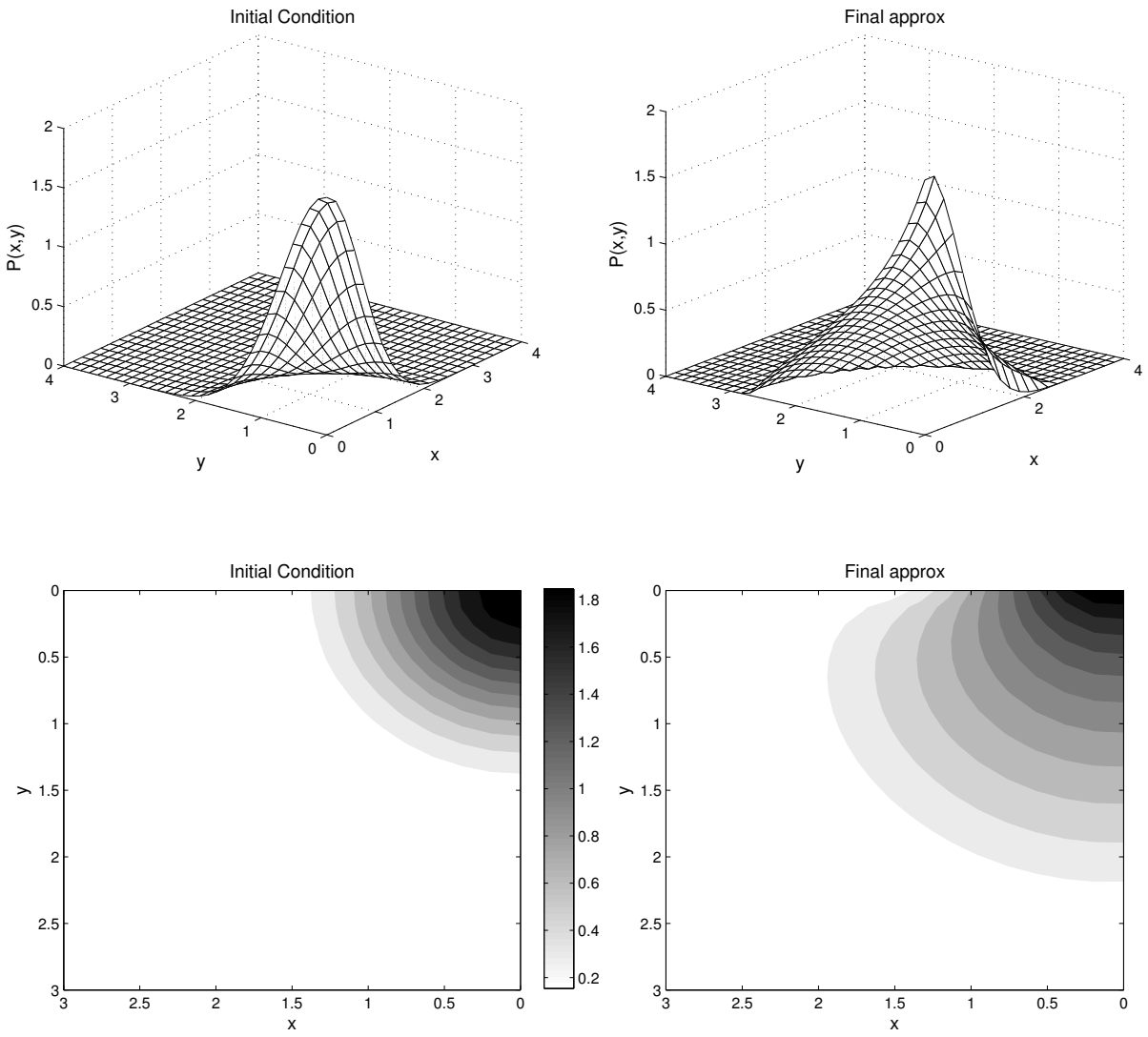


Figure 4.1: Approximation of (1.7) on $[0,1] \times [0,1]$ using ADI-FD. Parameters are $N_r = 21$, $N_z = 31$, $E = 2$, $\sigma = 1$, $dt = 0.001$, $T_f = 0.5$.

maintaing accuracy.

4.4 Displacement and Cratering

Simulating the flow of gas through a permeable membrane by approximating solutions to the Porous Medium Equation (1.5) can be extended to understand conditions for cratering. The approximations for the pressure from (1.5) can act as the body force for Navier's equation for volume displacement

$$\mu\nabla^2\mathbf{v} + (\mu+\lambda)\nabla(\nabla\mathbf{v}) + f = 0, \tag{4.5}$$

where $\mathbf{v}$ is the displacement vector field and μ and λ are Lamé parameters determined by the material. The body force $f = \rho g + \nabla P$ is determined by the pressure, P, approximated in (1.5), where ρ is the bulk density of the material and g is gravity.

The material constants μ, commonly known as the shear modulus, and λ can be calculated by

$$\mu = \frac{E_y}{2(1+w_p)}, \tag{4.6}$$

$$\lambda = \frac{w_p E_y}{(1+w_p)(1-2w_p)}. \tag{4.7}$$

For these equations E_y is Young's modulus measuring the stiffness of an elastic material while w_p is Poisson's ratio describing the strain in the material.

Once the displacement field is known, we can perform tests to determine the limits of stability of the regolith when rocket exhaust flows through it. In order to do so, the stress tensor must first be calculated by

$$T = \lambda(\text{trace}E_s)I + 2\mu E_s, \tag{4.8}$$

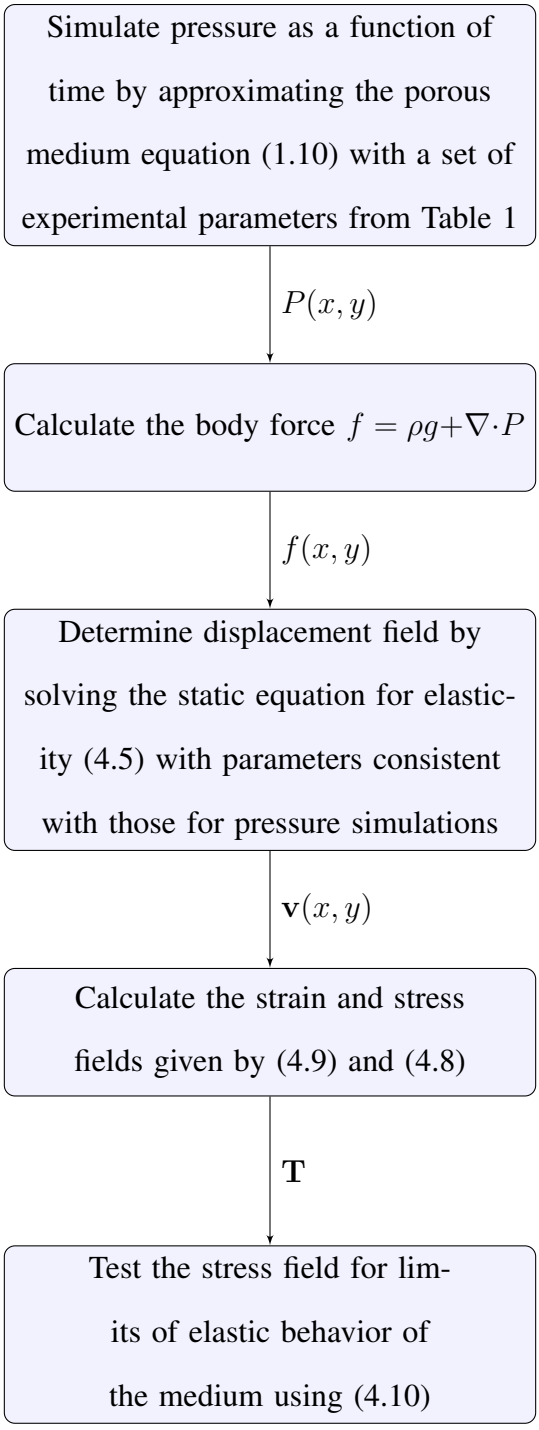
where I is the identity matrix and E_s is the strain tensor given by

$$E_s = \frac{1}{2}(\nabla\mathbf{v} + \nabla\mathbf{v}^T). \tag{4.9}$$

We can now test the capability of the regolith to support the stress by whether or not it violates the inequality

$$\tau < c + \sigma_d \tan \phi, \tag{4.10}$$

where $\tan \phi$ is given by the internal friction coefficient of the material, σ_d is any element on the diagonal of T, and τ is any off-diagonal element of T that is in the same row as σ_d. The parameter c is dependent upon the cohesion in the regolith. The failure to satisfy the condition (4.10) for any pairing of σ_d and τ for any possible rotations of T means that the material fails to act like an elastic and begins to act like a plastic. This is when cratering occurs. We have included a chart to understand the flow of steps for the simulation and testing process.


Simulate pressure as a function of time by approximating the porous medium equation (1.10) with a set of experimental parameters from Table 1
$P(x, y)$
Calculate the body force $f = \rho g + \nabla \cdot P$
$f(x, y)$
Determine displacement field by solving the static equation for elasticity (4.5) with parameters consistent with those for pressure simulations
$\mathbf{v}(x, y)$
Calculate the strain and stress fields given by (4.9) and (4.8)
$\mathbf{T}$
Test the stress field for limits of elastic behavior of the medium using (4.10)

4.5 Future Work

We now summarize the next steps to be taken to complete this project and attain the final results we seek. These steps include

- completing the implementation of an OSC spatial discretization coupled with an ADI temporal scheme for the nonlinear problem
- implementing a domain decomposition algorithm that is optimal for our problem
- deriving an appropriate 4^{th}-order boundary condition for $r = 0$
- coupling our pressure results with the displacement equation (4.5) using a finite element method presented in [24]
- running simulations for the entire model with various sets of experimental parameters to determine the conditions under which cratering occurs
- taking advantage of the alternating nature of the ADI scheme in order to parallelize the code
- running numerical experiments in the high performance computing STOKES lab at the University of Central Florida's Institue for Simulation and Training.

APPENDIX A: ADI-SPECTRAL METHOD

```
%% COMPUTATIONAL PARAMETERS
Nr = 31;      Nz = 21;
dt = .0001; T = 300*dt;
T_final = ceil(T/dt);
IR = eye(Nr+1);   IZ = eye(Nz+1);
%% PHYSICAL PARAMETERS
sigma = 1; %standard deviation of IC
E = 1;       %amplitude of IC
%% SET UP MESH WITH CHEBYSHEV NODES
domainparam = 4; %take domain out to 4*standard_deviation
r = domainparam*sigma*cos(pi*((0:Nr)/(Nr)));
r = fliplr(r); %Cheb nodes on [-4*sigma,4*sigma]
z = domainparam*sigma*((cos(pi*((0:Nz)/(Nz)))+ones(Nz+1,1)')/2);
z = fliplr(z); %Cheb nodes on [0,4*sigma]
r_inv(1:Nr+1) = 1./r(1:Nr+1);
%% INITIAL CONDITION
for i = 1:Nz+1
    u(i,1:Nr+1)= E*(exp(-((((r(1:Nr+1)).^2)/((sigma)^2))+((z(i))
        .^2)/((sigma)^2))));% + AmbPres;
    u_r(i,1:Nr+1) = -2*(r(1:Nr+1))/(sigma^2).*u(i,1:Nr+1);
    u_z(i,1:Nr+1) = -2*(z(i))/(sigma^2).*u(i,1:Nr+1);
    u_rr(i,1:Nr+1)= -2/(sigma^2).*u(i,1:Nr+1) - 2*(r(1:Nr+1))/(
        sigma^2).*u_r(i,1:Nr+1);
    u_zz(i,1:Nr+1)= -2/(sigma^2.)*u(i,1:Nr+1) - 2*(z(i))/(sigma
        ^2).*u_z(i,1:Nr+1);
```

```
end
%store initial condition and derivatives
g = u; g_r = u_r; g_rr = u_rr; g_z = u_z; g_zz = u_zz;
%% CHEBYSHEV SPECTRAL DIFFERENTIATION MATRICES
[D1Z,D2Z] = cheb(z,Nz,IZ);
[D1R,D2R] = cheb(r,Nr,IR);
%% BOUNDARY CONDITIONS
D2R(1,:) = 0; D2R(end,:) = 0; D2Z(1,:) = 0;      D2Z(end,:) = 0;
D1R(1,:) = 0; D1R(end,:) = 0; D1Z(1,:) = 0;      D1Z(end,:) = 0;
D1R(1,1) = 1; D1Z(1,1)   = 1; D1R(end,end) = 1; D1Z(end,end) = 1;
%% INITIALIZE F, P_tilde, P_tilde_z, P_hat, P_hat_r
F       = (g_r.^2 + g_z.^2 + repmat(r_inv,Nz+1,1).*g.*g_r)'; %F0
P_tilde = (g + (dt/2)*(g.*g_rr + g.*g_zz + F'))';     %P_tilde1/2
P_hat   = P_tilde';
P_hat0  = g;                                          %P_hat0
P_tilde_z = (D1Z*P_tilde')';      P_hat_r   = D1R*P_hat';
%% FIRST TIMESTEP n = 0
u_old = g;   u = u';  Q = u;
for kk = 2:Nr
    P_tilde_mat_z = repmat(P_tilde(kk,:),Nz+1,1);
    BZ = IZ + (dt/2)*(P_tilde_mat_z.*D2Z);
    Q(kk,:) = (BZ*u(kk,:)')';
    F(kk,:) = P_hat_r(kk,:).^2 + P_tilde_z(kk,:).^2 + ...
              r_inv(kk)*(P_tilde(kk,:).*P_hat_r(kk,:));
end
    F(1,:)   = 0; F(Nr+1,:) = 0;
```

```
    Q(end,:) = 0; Q(1,:)     = 0;
    u_half = Q;
for k = 2:Nz
    P_tilde_mat_r = repmat(P_tilde(:,k),1,Nr+1)';
    AR = IR - (dt/2)*(P_tilde_mat_r.*D2R);
    u_half(:,k) = linsolve(AR, Q(:,k) + ((dt/2)*F(:,k)));
end
    Q(:,end) = 0;
    u_half(:,end)=0;
    u = u_half;
for L = 2:Nr
   P_tilde_mat_z = repmat(P_tilde(L,:),Nz+1,1);
   AZ = IZ - (dt/2)*(P_tilde_mat_z.*D2Z);
   u(L,:) = linsolve(AZ, 2*u_half(L,:)' - Q(L,:)');
end
u_old_half = u_half';    u = u';
P_tilde   =  (3/2)*u - (1/2)*u_old;
P_hat     =  3*u_old_half - 2*P_hat0;    %P_hat1+1/2
P_hat_r   =  D1R*P_hat';     P_tilde_z= (D1Z*P_tilde)';
%% REMAINING TIMESTEPS
for n = 2:T_final
    u_old = u;
    u = u';
    P_tilde = P_tilde';
    Q = u;
    for kk2 = 2:Nr
```

```
        P_tilde_mat_z = repmat(P_tilde(kk2,:),Nz+1,1);
        BZ = IZ + (dt/2)*(P_tilde_mat_z.*D2Z);
        Q(kk2,:) = (BZ*u(kk2,:)')';
        F(kk2,:) = P_hat_r(kk2,:).^2 + P_tilde_z(kk2,:).^2 + ...
                   r_inv(kk2)*(P_tilde(kk2,:).*P_hat_r(kk2,:));
    end
    F(1,:)     = 0; F(Nr+1,:) = 0;
    Q(end,:)   = 0; Q(1,:)    = 0;
    u_half_new = Q;
    for k2 = 2:Nz
        P_tilde_mat_r = repmat(P_tilde(:,k2),1,Nr+1)';
        AR = IR - (dt/2)*(P_tilde_mat_r.*D2R);
        u_half_new(:,k2) = linsolve(AR, Q(:,k2)+(dt/2)*F(:,k2));
    end
    u_half_new(:,end)=0;
    Q(:,end) = 0;
    u = u_half_new  ;
    for L2 = 2:Nr
        P_tilde_mat_z = repmat(P_tilde(L2,:),Nz+1,1);
        AZ = IZ - (dt/2)*(P_tilde_mat_z.*D2Z);
        u(L2,:) = linsolve(AZ, 2*u_half_new(L2,:)' - Q(L2,:)');
    end
    u = u';
    u_half_new = u_half_new';
    P_tilde = (3/2)*u - (1/2)*u_old;
    P_hat   = 2*u_half_new - u_old_half;
```

```
    u_old_half = u_half_new;
    P_hat_r   =  D1R*P_hat';       P_tilde_r=  D1R*P_tilde';
    P_hat_z   =  D1Z*P_hat;        P_tilde_z= (D1Z*P_tilde)';
    u_half_new = u_half_new';
end
```

APPENDIX B: ADI-OSC METHOD

```
function []= Heat2D_Linear_ADI_OSC
%% COMPUTATIONAL PARAMETERS
Nr = 31; Nz = 11;
dt = .0001; T = .001;
T_final = ceil(T/dt);
%% PHYSICAL PARAMETERS
Thrust = 1;
diam   = .3;
sigma  = diam/2;
E = Thrust/(pi*diam^2/4);
%% EQUIDISTANT MESH
domparam = 4;
r=linspace(-domparam,domparam,Nr+1); z=linspace(0,domparam,Nz+1);
hr = 2*domainparam/(Nr);             hz = domparam/(Nz);
%% GAUSS POINTS
nodes_gauss = zeros(Nr,2);
c = r(1);
for index = 1:Nr
    nodes_gauss(index,1) = c + hr*(1-1/sqrt(3))/2;
    nodes_gauss(index,2) = c + hr*(1+1/sqrt(3))/2;
    c = hr + c;
end
rgauss1=Mat2Vec2(nodes_gauss);
rgauss=[r(1);rgauss1(:);r(end)]';
%% INITIAL CONDITION
```

```
for i = 1:Nz+1
    u(i,1:2*Nr+2) = E*exp(-(rgauss(1:2*Nr+2).^2/sigma^2 + ...
                        z(i).^2/sigma^2));
    u_r(i,1:2*Nr+2) = -2*(rgauss(1:2*Nr+2))/(sigma^2).*u(i,1:2*Nr
       +2);
    u_z(i,1:2*Nr+2) = -2*(z(i))/(sigma^2).*u(i,1:2*Nr+2);
end
u = u'; u_z = u_z'; u_r = u_r';
g=u;
%% STORE DERIVATIVE VALUES IN Z
bigmat = zeros(2*(Nr+1),2*(Nz+1));
bigmat(1:2*Nr+2,1:2:2*Nz+1) =u(1:2*Nr+2,1:Nz+1);
bigmat(1:2*Nr+2,2:2:2*Nz+2) =hz.*u_z(1:2*Nr+2,1:Nz+1);
%% OSC DIFFERENTIATION MATRICES
[LR,BR] = script_L_concat(hr,Nr);
[LZ,BZ] = script_L_alternating(hz,Nz);
%% BUILD MATRICES FOR RHS AND LHS OF ADI METHOD
AR = BR - (dt/2)*LR; CR = BR + (dt/2)*LR;
AZ = BZ - (dt/2)*LZ; CZ = BZ + (dt/2)*LZ;
%% BOUNDARY CONDITIONS
AR(1,1) = 1; AR(end,end)  = 1;
CR(1,1) = 1; CR(end,Nr+1) = 1;
AZ(1,1) = 1; AZ(end,end)  = 1;
CZ(1,1) = 1; CZ(end,end-1)= 1;
%% TIME-STEPPING LOOP
u = bigmat;
```

```
for n = 1:T_final
    Q=u;
    for ii = 1:2*Nr+2
        Q(ii,:) = CZ*u(ii,:)';
    end
    for k = 1:2*Nz+2
        u_half(:,k) = linsolve(AR, Q(:,k));
    end
    Q = Q';  u_half = u_half';  u = u';
    for LL = 1:2*Nz+2
        Q2(LL,:) = CR*u_half(LL,:)';
    end
    for L = 1:2*Nr+2
       u(:,L) = linsolve(AZ, Q2(:,L));
    end
    Q = Q';  u_half = u_half';  u = u';
end
%% TRANSFORM SOLUTION FROM GAUSS POINTS TO EQUIDISTANT IN R
BR(1,1) = 1;  BR(end,end) = 1;
for MM=1:2*Nz+2
    unew2(:,MM) = linsolve(BR,u(:,MM));
end
u_final = unew2(1:Nr+1,1:2:2*Nz+1); %SKIP OVER DERIVATIVES IN Z
```

APPENDIX C: OSC MATRICES

OSC_Matrices_alternating.m

```
function [I_OSC,L1,L2]=OSC_Matrices_alternating(h,N)
% For basis {v0,s0,v1,s1,...vn,sn}
%% NEUMANN BOUNDARY CONDITIONS
mu1 = 0; nu1 = 1; mu2 = 0; nu2 = 1;
%% VALUES FOR BLOCK COMPONENTS
a1 = 2*sqrt(3); a2 = 1 + sqrt(3); a3 = sqrt(3) - 1;
c1 = (9+4*sqrt(3))/18; c2 = (3+sqrt(3))/36;
c3 = (9-4*sqrt(3))/18; c4 = (3-sqrt(3))/36;
%% BUILD BLOCK COMPONENTS
A = (1/h^2).* [-a1 -a2 a1 -a3; a1 a3 -a1 a2];
B = (1/h).* [-1 1/a1 1 -1/a1; -1 -1/a1 1 1/a1];
C = [c1 c2 c3 -c4; c3 c4 c1 -c2];
%% OSC MATRICES
L2 = zeros(2*N+2, 2*N+2);
L1 = zeros(2*N+2, 2*N+2);
I_OSC = zeros(2*N+2, 2*N+2);
for i = 1:N
    L2(2*i:2*i+1, 2*i-1:2*i+2) = A;
    L1(2*i:2*i+1, 2*i-1:2*i+2) = B;
    I_OSC(2*i:2*i+1, 2*i-1:2*i+2) = C;
end
%% FIRST AND LAST ROWS TAKE IN BOUNDARY CONDITIONS
I_OSC(1,1:2)=[mu1 nu1/h]; I_OSC(end,end-1:end)=[mu2 nu2/h];
L1(1,1:2)=[mu1 nu1/h];    L1(end,end-1:end)=[mu2 nu2/h];
L2(1,1:2)=[mu1 nu1/h];    L2(end,end-1:end)=[mu2 nu2/h];
```

OSC_Matrices_alternating.m

```
function [L1,L2,ID_OSC]=OSC_Matrices_concatonated(h,N)
% For basis {v0,v1,...vn,s0,s1,...,sn}
%% NEUMANN BOUNDARY CONDITIONS
mu1 = 0; nu1 = 1; mu2 = 0; nu2 = 1;
%% %% VALUES FOR BLOCK COMPONENTS
a1 = 2*sqrt(3); a2 = 1 + sqrt(3); a3 = sqrt(3) - 1;
c1 = (9+4*sqrt(3))/18; c2 = (3+sqrt(3))/36;
c3 = (9-4*sqrt(3))/18; c4 = (3-sqrt(3))/36;
%% BUILD BLOCK COMPOENENTS
W2 = (1/h^2).*[-a1 a1;a1 -a1];Z2 = (1/h^2).*[-a2 -a3; a3 a2];
W1 = (1/h).* [-1 1; -1 1];     Z1 = (1/h).*[1/a1 -1/a1; -1/a1 1/
    a1];
Cw = [c1 c3; c3 c1];           Cz = [c2 -c4; c4 -c2];
%% OSC MATRICES
L2 = zeros(2*N+2, 2*N+2);
L1 = zeros(2*N+2, 2*N+2);
I_OSC = zeros(2*N+2, 2*N+2);
for i = 1:N
    L2(2*i:2*i+1, i:i+1)                = W2;
    L2(2*i:2*i+1,(N+1)+i:(N+1)+i+1)= Z2;
    L1(2*i:2*i+1, i:i+1)                = W1;
    L1(2*i:2*i+1,(N+1)+i:(N+1)+i+1)= Z1;
    I_OSC(2*i:2*i+1, i:i+1)                    = Cw;
    I_OSC(2*i:2*i+1, (N+1)+i:(N+1)+i+1)    = Cz;
end
```

```
%% FIRST AND LAST ROWS TAKE IN BOUNDARY CONDITIONS
ID_OSC(1,1)      = mu1; ID_OSC(1,N+2)   = nu1/h;
ID_OSC(end,N+1)= mu2; ID_OSC(end,end)= nu2/h;
L1(1,1)      = mu1;      L1(1,N+2)   = nu1/h;
L1(end,N+1)= mu2;      L1(end,end)= nu2/h;
L2(1,1)      = mu1;      L2(1,N+2)   = nu1/h;
L2(end,N+1)= mu2;      L2(end,end)= nu2/h;
```

APPENDIX D: CHEBYSHEV SPECTRAL DIFFERENTIATION MATRICES

cheb.m

```
function [D1,D2] = cheb(z,N,I)
% z is the spatial nodes
% N is the number of spatial nodes
% I is the identity matrix
cz = [2; ones(N-1,1); 2].*(-1).^(0:N)';
Z  = repmat(z',1,N+1); %creates a 1xNz+1 matrix with copies of z'
dZ = Z-Z';             %gives zeros along diagonal
m  = (cz*(1./(cz))')./(dZ + I);
m  = m - diag(sum(m'));
M  = m;
D1 = M;                %First derivative matrix
D2 = M*M;              %Second derivative matrix
```

LIST OF REFERENCES

[1] Aitbayev, R., & Bialecki, B. (2000). Orthogonal Spline Collocation for Nonlinear Dirichlet Problems. *SIAM Journal on Numerical Analysis*, 38(5), 1582.

[2] Bialecki, B., & Fernandes, R. I. (2006). An Alternating-direction implicit orthogonal spline collocation scheme for nonlinear parabolic problems on rectangular polygons. *SIAM Journal on Scientific Computing*, 28(3), 1054.

[3] Bialecki, B., & Fernandes, R. I. (2009). An Alternating-direction implicit backward differentiation orthogonal spline collocation method for linear variable coefficient parabolic equations. *SIAM Journal on Numerical Analysis*, 47(5), 3429.

[4] Bialecki, B., & Fairweather, G. (2001). Orthogonal spline collocation methods for partial differential equations. *Journal of Computational and Applied Mathematics*, 128(1-2), 55.

[5] Bialecki, B., & Frutos, J. d. (2010). ADI spectral collocation methods for parabolic problems. *Journal of Computational Physics*, 229(13), 5182.

[6] Bialecki, B., & Dryja, M. (2003). A nonoverlapping domain decomposition method for orthogonal spline collocation problems. *SIAM Journal on Numerical Analysis*, 41(5), 1709.

[7] J. P. Boyd, (1989). *Chebyshev and Fourier Spectral Methods*, Berline: Springer-Verlag.

[8] Bruce, G. H., Peaceman, D. W., Rachford, H. H., & Rice, J. D. (1953). Calculations of unsteady-state gas flow through porous media. *Journal of Petroleum Technology*, 5(3), 79.

[9] Fairweather, G. (1978). *Finite element galerkin methods for differential equations*. New York: M. Dekker.

[10] Fernandes, R. I. & Fairweather, G. (1993). Analysis of alternating direction collocation methods for parabolic and hyperbolic problems in two space variables. *Numerical Methods for Parial Differential Equations*, 9(2), 191.

[11] Greenwell-Yanik, C. E., & Fairweather, G. (1986). Analysis of spline collocation methods for parabolic and hyperbolic problems in two space variables. *SIAM Journal on Numerical Analysis*, 23(2), 282.

[12] Kincaid, D. & Cheney, W. (2002) . *Numerical analysis: mathematics of scientific computing*, Pacific Grove, Calif.: Brooks/Cole.

[13] Lou, Z., Bialecki, B., & Fairweather, G. (1998). Orthogonal spline collocation methods for biharmonic problems. *Numerische Mathematik*, 80(2), 267.

[14] Metzger, P. T., Lane, J. E., Immer, C. D., & Clements, S. (2008). Cratering and blowing soil by rocket engines during lunar landings. *6th International Conference on Case Histories in Geotechnical Engineering*.

[15] Metzger, P. T., Latta, R. C. III, Schuler, J. M., and Immer, C. D. (2008). Craters formed in granular beds by impinging jets of gas. *AIP Conference Proceedings*, 1145(1), 767.

[16] Metzger, P. T., Immer, C. D., Donahue, C. M., Vu, B. T., Latta, R. C., & Deyo-Svendsen, M. (2009). Jet-induced cratering on a granular surface with application to lunar spaceports. *Journal of Aerospace Engineering*, 22(1), 24.

[17] Morton, K. W., & Meyers, D. F. (2006). *Numerical Solution of Partial Differential Equations* (2nd ed.). Cambridge: Cambridge University Press.

[18] Peaceman, D. W., & Rachford, H. H. (1955). The numerical solution of parabolic and elliptic differential equations. *Journal of the Society for Industrial and Applied Mathematics*, 3(1), 28.

[19] Scott, R. F. & Ko, H. Y. (1968). Transient rocket-engine gas flow in soil. *AIAA Journal*, 6(2), 258.

[20] Thomas, J. W. (1995). *Numerical Partial Differential Equations*. New York: Springer.

[21] Trefethen, L. N. (2000). *Spectral Methods in MATLAB*. Phyladelphia, PA: SIAM.

[22] Vázquez, J. L. (2007). *The Porous Medium Equation mathematical theory*. Oxford: Clarendon.

[23] Welfert, B. D. (1997). Generation of pseudospectral differentiation matrics I. *SIAM Journal on Numerical Analysis*, 34(4), 1640.

[24] Brennan, B. (2011). *Numerical Computations for PDE models of rocket exhaust flow in soil* (Unpublished master's thesis). University of Central Florida, Orlando, FL.

[25] Fairweather, G. (1998). *Notes for a lecture on Orthogonal Spline Collocation methods.* Colorado School of Mines, Golden, Co.