LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-SR-822456

Parallel Time Integration for Constrained Optimization

D. King, C. Hills, M. Kielstra, M. Torrence

May 12, 2021

Disclaimer

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Research in Industrial Projects for Students

Sponsor

Lawrence Livermore National Laboratory

Final Report

Parallel Time Integration for Constrained Optimization

Student Members

Dylan King (Project Manager), Wake Forest University
dylank1029@gmail.com

Caroline Hills, University of Notre Dame
chills1@nd.edu

Michael Kielstra, Harvard University
pmkielstra@college.harvard.edu

Matt Torrence, Gettysburg College
torrma01@gettysburg.edu

Academic Mentor

HanQin Cai, hqcai@math.ucla.edu

Sponsoring Mentors

Rob Falgout, falgout2@llnl.gov
Jeff Hittinger, hittinger1@llnl.gov
Cosmin Petra, petra1@llnl.gov

Date: August 19, 2020

This project was jointly supported by Lawrence Livermore National Laboratory and NSF Grant ⟨DMS⟩ 1440415.

Abstract

The number of transistors in an average processor continues to increase, but individual clock speeds have plateaued. Those transistors are instead going into additional cores, increasing the number of different things that a processor can do at once and placing an emphasis on parallel computation. Many problems in scientific computing follow a time-evolution model, and it can be difficult to solve such problems in parallel across the temporal domain. The Multi-Grid Reduction In Time (MGRIT) algorithm, developed at Lawrence Livermore National Laboratory (LLNL), solves differential equations with a method designed specifically to take advantage of extreme numbers of processors by parallelizing across time. The Tri-diagonal MGRIT (TriMGRIT) algorithm, also developed at LLNL, is a generalization of MGRIT which enables parallel-in-time solving of a greater number of problems. Constrained optimization problems, in particular, may be solved in parallel using TriMGRIT. These consist of choosing a control function such that an objective functional is minimized, constrained by a differential-equation. We consider two such problems: applying torque to a pendulum to bring it to a gentle stop and moving a crowd of people from one distribution into another. We also perform some miscellaneous theoretical and practical research, including investigating the use of a line-search subroutine to refine intermediate TriMGRIT results and preliminary work on strategies for choosing operators for TriMGRIT to use.

Acknowledgments

This project was jointly supported by Lawrence Livermore National Laboratory and NSF Grant ⟨DMS⟩ 1440415. We would like to thank everyone at the Institute for Pure and Applied Mathematics who organized the Research in Industrial Projects for Students program and supported us during this project, especially Dr. Susana Serna and Neli Petrosyan, as well as our sponsoring industry mentor Dr. Robert Falgout and academic mentor Dr. Han-Qin Cai.

Contents

Abstract **3**

Acknowledgments **5**

1 Introduction and Background **11**
1.1 Introduction 11
1.2 Background 11

2 A Pendulum Problem **15**
2.1 Problem Definition 15
2.2 Preparing for TriMGRIT 15
2.3 Remarks 17

3 A Crowd Control Problem **19**
3.1 Pedestrian Movement Models 19
3.2 Haber and Horesh's Crowd Model 22

4 Miscellaneous Theoretical Research and Practical Contributions **29**
4.1 Rescaling and Coarse-Grid Operators 29
4.2 Optimizing the Choice of Coarse-Grid Operator 32
4.3 Combinations of TriMGRIT and Line Search 32
4.4 Contributions to XBraid Code 34
4.5 Our Code 34

5 Conclusion and Further Research **35**
5.1 Conclusion 35
5.2 Recommendations for Further Research 35

APPENDIXES

A Abbreviations **37**

REFERENCES

Selected Bibliography Including Works Cited **39**

List of Figures

1.1 Example Fine and Coarse Levels in One Dimension 13

3.1 Non-Preconditioned Simple Solve Visualization 26
3.2 Non-Preconditioned TriMGRIT Visualization 27
3.3 Preconditioned TriMGRIT Visualization 27
3.4 MGRIT Convergence Ratio 28

4.1 3D plot of the transformation from α and β to the α with estimated coarse-grid operator equal to the exact coarse-grid operator of the original α. . . . 31

Chapter 1

Introduction and Background

1.1 Introduction

Lawrence Livermore National Laboratory (LLNL) is a federal research facility with a focus on defense, energy, intelligence, and scientific computing (especially as it pertains to these disciplines) [2]. Within LLNL, the Center for Applied Scientific Computing (CASC) serves as LLNL's window to the broader computer science, computational physics, applied mathematics, and data science research communities [1]. CASC (directed by Dr. Jeff Hittinger) studies extreme-scale computing throughout these fields, and parallelism, or the breakdown of computations across multiple processors, is a keystone design feature of modern supercomputing machines. Designing algorithms which efficiently utilize ever-increasing numbers of processors is an important facet of CASC's work.

This summer we studied TriMGRIT, one such algorithm designed by LLNL to utilize parallelism, and its application to constrained optimization problems. This report will formally introduce the TriMGRIT algorithm and the necessary prerequisites in the remainder of Chapter 1; show how TriMGRIT can be applied to a simple constrained optimization problem concerning pendulum dynamics in Chapter 2; introduce spatial complexity and apply TriMGRIT to pedestrian dynamics in Chapter 3; present analysis and modifications to the TriMGRIT algorithm itself (in some cases with reference to analogous questions for MGRIT) in Chapter 4; and finally summarize our work and list some potential avenues for future progress in Chapter 5.

1.2 Background[1]

This project studies the performance of a new parallel-in-time integration solver for time dependent constrained optimization problems. Unfortunately, the traditional simulation approach of sequential time marching is becoming a bottleneck as computer architectures increasingly rely on higher concurrency to provide greater peak performance (million-way parallelism today, and billion-way expected on the coming exascale platforms). Researchers have developed new parallel-in-time solution approaches to avoid this bottleneck and significantly speed up computer simulations. These approaches solve for the full space-time system all at once in parallel. The multigrid reduction in time (MGRIT) method and open source software XBraid were developed at LLNL to take advantage of existing simulation

[1]This background material is taken and derived from the writing of R. D. Falgout.

code bases and technologies as much as possible, allowing scientists to migrate to a parallel-in-time simulation paradigm more easily than would otherwise be possible. Significant advances have already been made on the development of MGRIT and other parallel-in-time approaches, but many outstanding research questions remain. Fortunately, work in this area has increased dramatically in the past few years.

Many computer simulations of optimization problems are time dependent; for example, optimizing the shape of a car or an airfoil to minimize drag has a time component to it. These applications traditionally involve sequential time marching both forward and backward in time. Recently researchers have used MGRIT and XBraid to parallelize and speed up the time integration component of these optimization algorithms [16]. This project will investigate a new extension of the MGRIT solver, called TriMGRIT, that can be applied to time-dependent constrained optimization algorithms which require the solution of a (potentially nonlinear) system that is fully coupled both forward and backward in time. TriMGRIT provides an approach for solving these systems in parallel.

1.2.1 Parallel Time Integration with MGRIT and XBraid

The MGRIT algorithm (introduced in [13]), implemented in the open-source XBraid library [11], is a parallel-in-time approach for solving time dependent problems that is designed to be as non-intrusive as possible on existing codes.

Consider the basic one-step time discretization

$$u_i = \Phi_i(u_{i-1}) + g_i, \text{ for } i = 1, 2, ..., N \text{ and } u_0 = g_0, \tag{1.1}$$

of some time-dependent process on the interval $[0, T]$. Partition the time domain as in Figure 1.1, where $t_i = i\delta t$ is the fine time grid and $T_i = mi\delta t$ is the coarse time grid for coarsening factor m. Let u_i be an approximation to $u(t_i)$. Then, considering the linear case for simplicity, sequential time stepping is equivalent to a forward solve of the system $A\mathbf{u} = \mathbf{g}$,

$$\begin{pmatrix} I & & & \\ -\Phi_1 & I & & \\ & \ddots & \ddots & \\ & & -\Phi_N & I \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_N \end{pmatrix} = \begin{pmatrix} g_0 \\ g_1 \\ \vdots \\ g_N \end{pmatrix}, \tag{1.2}$$

where the solution $\mathbf{u}$ includes the state at all time points. MGRIT replaces this $\mathcal{O}(N)$ sequential solve of (1.2) with an $\mathcal{O}(N)$ iterative method using parallel multigrid reduction to simultaneously solve at all time points. As such, MGRIT converges to the same solution that serial time stepping produces. Additionally, the user need only define a routine that applies Φ, the operator that advances the solution from one step to the next, making the method non-intrusive and allowing for reuse of serial time-stepping codes.

The concurrency in MGRIT is made possible by coarse time grids that accelerate convergence to the solution on the original fine grid. Each coarse time grid is formed with an approximate block cyclic reduction strategy that eliminates block rows and columns in the system (1.2). If all the block rows corresponding to the points not in the coarse grid (black points in Figure 1.1) are eliminated, then a Schur-complement system, smaller by a factor of $1/m$, would result. This coarse system is approximated by a cheaper system that has the same form as (1.2) and uses a coarse time-stepping operator based on the larger coarse time-step size. Often, this new operator is taken to be just a rediscretization in time.

Relaxation complements the coarse-grid corrections in MGRIT, and alternates between block Jacobi applied to the fine-point rows in (1.2) and block Jacobi applied to the coarse-point rows (for more information on Jacobi methods see [24]). Each relaxation sweep is a

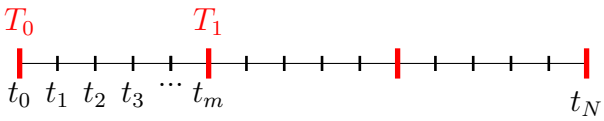

Figure 1.1: Example fine and coarse level for a coarsening factor of $m = 5$. Black points are present on only the fine level, whereas red points are on both the fine and coarse level.

highly parallel process. Interpolation between time grids is done by injection. This process is implemented in XBraid as a full approximation storage (FAS) multigrid cycle, capable of solving nonlinear problems.

For further information on multigrid algorithms, see [26].

1.2.2 TriMGRIT

MGRIT may be viewed as an approximate block cyclic reduction approach for solving block lower bidiagonal systems of the form given in (1.2). The approximate block cyclic reduction idea can generally be applied to block tridiagonal systems where the system matrix A has the form

$$A = \begin{pmatrix} C_1 & -E_1 & & \\ -W_2 & C_2 & -E_2 & \\ & \ddots & \ddots & \ddots \\ & & -W_N & C_N \end{pmatrix}. \tag{1.3}$$

It is often convenient and useful to represent systems like the ones in (1.2) and (1.3) with stencils. The (block) stencil for A in (1.3) is given by

$$A = \begin{bmatrix} -W_i & C_i & -E_i \end{bmatrix}, \tag{1.4}$$

where the index i indicates the i^{th} block row (or i^{th} grid block). For a short introduction to stencils and their relationship to matrices, see [10].

Block cyclic reduction for (1.3) can be viewed as a multigrid method that uses F-relaxation and an exact coarse-grid correction composed of so-called ideal interpolation P, ideal restriction R, and a Petrov-Galerkin coarse-grid operator RAP (for more information on Petrov-Galerkin operators see [14]). The stencils for these components are

$$P = \left] \begin{matrix} C_{i-1}^{-1}E_{i-1} & I & C_{i+1}^{-1}W_{i+1} \end{matrix} \right[, \tag{1.5}$$

$$R = \begin{bmatrix} W_iC_{i-1}^{-1} & I & E_iC_{i+1}^{-1} \end{bmatrix}, \tag{1.6}$$

$$RAP = \begin{bmatrix} -W_iC_{i-1}^{-1}W_{i-1} & (C_i - W_iC_{i-1}^{-1}E_{i-1} - E_iC_{i+1}^{-1}W_{i+1}) & -E_iC_{i+1}^{-1}E_{i+1} \end{bmatrix} \tag{1.7}$$

where we have used the reverse square brackets $]\cdot[$ to denote a column-oriented stencil. With these components, the two-grid method solves the problem in just one cycle. Notice, however, that all of these components involve C_i^{-1} which may not be practical to compute. In addition, the center block for RAP may be difficult to invert even if C_i is easy to invert, making it difficult to recurse to a full multilevel cyclic reduction method. Multigrid reduction circumvents this issue by using various approximations to P, R, and RAP. This leads to a method that is no longer exact, but when designed appropriately, converges with $\mathcal{O}(N)$ work.

Having introduced TriMGRIT and the affiliate background material, in the next Chapter we introduce the first constrained optimization problem we solve using TriMGRIT, concerning the motion of a pendulum.

Chapter 2

A Pendulum Problem

We begin our exploration of TriMGRIT by solving a non-trivial but straightforward problem in Newtonian mechanics. This demonstrates the power of the method, and our notes function as a kind of user's guide for the mathematics behind the algorithm.

2.1 Problem Definition

Say we have a hanging pendulum with angle $\theta(t)$ at time t. We apply a torque $\alpha(t)$. This gives us equations of motion

$$\begin{cases} \ddot{\theta}(t) + \lambda\dot{\theta}(t) + \omega^2\theta(t) = \alpha(t) \\ \theta(0) = \theta^{in}, \dot{\theta}(0) = \dot{\theta}^{in} \end{cases}$$

for constants θ^{in}, $\dot{\theta}^{in}$. These equations contain some second derivatives. Although we can discretize to approximate second derivatives directly, we will use only first derivatives and consequently rewrite these equations as:

$$\begin{cases} \dot{\theta}_1(t) = \theta_2(t) \\ \dot{\theta}_2(t) = \alpha(t) - \lambda\theta_2(t) - \omega^2\theta_1(t) \\ \theta_1(0) = \theta^{in}, \theta_2(0) = \dot{\theta}^{in} \end{cases}$$

where we are using two subscripted functions, θ_1 and θ_2, to denote θ and $\dot{\theta}$ respectively. We would like to choose a torque function to minimize the total amount of "swinging" that the pendulum does over the first second of its lifetime. In particular, we would like to minimize

$$\mathcal{J}(\theta, \alpha) = \int_0^1 \theta_1(t)^2 + \theta_2(t)^2 + \gamma\alpha(t)^2 dt \tag{2.1}$$

for a fixed small γ. The purpose of this final $\gamma\alpha(t)^2$ term is to keep the solver from simply applying infinite force to the pendulum to control its movement exactly.

2.2 Preparing for TriMGRIT

Begin by discretizing in time. Let t^i be timestep $i \in [0, N] \cap \mathbb{Z}$, and let $\theta_j^i \approx \theta_j(t^i)$. Define α^i similarly.

Now discretize the derivatives. From the first constraint, we know that $\frac{\theta_1^{i+1}-\theta_1^i}{\delta t} = \theta_2^i$, or $\theta_1^{i+1} = \theta_1^i + \delta t \theta_2^i$. Similarly, $\frac{\theta_2^{i+1}-\theta_2^i}{\delta t} = -\lambda\theta_2^i - \omega^2\theta_1^i + \alpha^i$, which rearranges to $\theta_2^{i+1} = (1-\delta t\lambda)\theta_2^i - \delta t\omega^2\theta_1^i + \delta t\alpha^i$. We can rewrite this as

$$\begin{pmatrix}\theta_1^{i+1}\\ \theta_2^{i+1}\end{pmatrix} = \Phi\begin{pmatrix}\theta_1^i\\ \theta_2^i\end{pmatrix} + d\alpha^i \tag{2.2}$$

where $\Phi = \begin{pmatrix}1 & \delta t\\ -\omega^2\delta t & 1-\lambda\delta t\end{pmatrix}$ and $d = \begin{pmatrix}0\\ \delta t\end{pmatrix}$. The discrete initial value problem is equivalent to

$$\underbrace{\begin{pmatrix} I & & & \\ -\Phi & I & & \\ & \ddots & \ddots & \\ & & -\Phi & I\end{pmatrix}}_{=:L}\begin{pmatrix}\theta^1\\ \vdots\\ \theta^N\end{pmatrix} - \underbrace{\begin{pmatrix} d & & \\ & \ddots & \\ & & d\end{pmatrix}}_{=:D}\begin{pmatrix}\alpha^0\\ \vdots\\ \alpha^{N-1}\end{pmatrix} = \underbrace{\begin{pmatrix}\Phi\theta^{in}\\ 0\\ \vdots\\ 0\end{pmatrix}}_{=:g}. \tag{2.3}$$

We can approximate the objective function $\mathcal{J}$ given in Equation (2.1) by the Riemann sum $J(\boldsymbol{\theta},\alpha) = \sum_{i=0}^{N}\delta_t{\theta_1^i}^2 + \delta_t{\theta_2^i}^2 + \gamma\delta_t{\alpha^i}^2$.

The Lagrangian of this system is given by $\mathcal{L}(\boldsymbol{\theta},\alpha,w) = J(\boldsymbol{\theta},\alpha) + w^T(L\boldsymbol{\theta} - D\alpha - g)$. Taking gradients with respect to $\boldsymbol{\theta}$, α, and w gives us the following Karush-Kuhn-Tucker (KKT) equations:

$$2\delta t\boldsymbol{\theta} + L^T w = 0 \tag{2.4}$$
$$2\gamma\delta t\alpha - D^t w = 0 \tag{2.5}$$
$$L\boldsymbol{\theta} - D\alpha - g = 0. \tag{2.6}$$

Hence, we want to solve the following KKT system:

$$\begin{pmatrix}2\delta t I_{2N} & 0 & L^T\\ 0 & 2\gamma\delta t I_N & -D^T\\ L & -D & 0\end{pmatrix}\begin{pmatrix}\boldsymbol{\theta}\\ \alpha\\ w\end{pmatrix} = \begin{pmatrix}0\\ 0\\ g\end{pmatrix}. \tag{2.7}$$

To solve this system, we transform using the Schur complement before applying TriMGRIT. For more information on KKT equations, see [3].

2.2.1 The Schur Complement

To simplify this block system, we solve for the vector w using a Schur complement method. If we perform the matrix multiplication in Equation (2.7) we find

$$2\delta t I_{2N}\boldsymbol{\theta} + L^T\boldsymbol{w} = 0 \tag{2.8}$$
$$2\gamma\delta t I_N\boldsymbol{\alpha} - D^T\boldsymbol{w} = 0 \tag{2.9}$$
$$L\boldsymbol{\theta} - D\boldsymbol{\alpha} = g \tag{2.10}$$

and if we solve the rows one and two for $\boldsymbol{\theta}$, $\boldsymbol{\alpha}$ respectively, before substituting into row three, we find that $\boldsymbol{w}$ is given by

$$-\left(\frac{1}{2\delta t}LL^T + \frac{1}{2\gamma\delta t}DD^T\right)\boldsymbol{w} = g. \tag{2.11}$$

This algebraic manipulation, generally, is known as the block Schur-complement method (for more information, see [30]). The stencil for this equation is

$$\left[-\frac{1}{2\delta t}\Phi \quad \left(\frac{1}{2\delta t}\left(I+\Phi\Phi^T\right)+\frac{1}{2\gamma\delta t}dd^T\right) \quad -\frac{1}{2\delta t}\Phi^T\right]. \tag{2.12}$$

For reference, we can expand the center coefficients to find

$$\frac{1}{2\delta t}\left(I+\begin{pmatrix}1 & \delta t\\ -\omega^2\delta t & (1-\lambda\delta t)\end{pmatrix}\begin{pmatrix}1 & -\omega^2\delta t\\ \delta t & (1-\lambda\delta t)\end{pmatrix}\right)+\frac{1}{2\gamma\delta t}\begin{pmatrix}0 & 0\\ 0 & \delta t^2\end{pmatrix}. \tag{2.13}$$

Equation (2.11) is a tridiagonal system, so we can solve it (parallel in time) with TriMGRIT. Once we do this and obtain a value for $\boldsymbol{w}$, we can find return and find $\boldsymbol{\theta}$ and α using the equations

$$\boldsymbol{\theta} = -\frac{1}{2\delta t}L^T w \tag{2.14}$$

$$\boldsymbol{\alpha} = \frac{1}{2\gamma\delta t}D^T w. \tag{2.15}$$

TriMGRIT will take a tridiagonal equation of the form $Ax = b$ and attempt to solve for x. This requires writing two functions: `triResidual()`, which calculates a single component of $Ax' - b$ for some x', and `triSolve()`, which approximately solves for a single component of $Ax = b$. Both of these are passed the values of the components before and after the one being solved for. In this case, that means that `triSolve()` is given values for w_{i-1} and w_{i+1} and must estimate w_i, assuming that those two values are satisfactory. This can in fact be done exactly in this case, since we can invert Φ.

This completes the example of how one might solve a rather simple constrained optimization problem using TriMGRIT. In the next section, we increase the complexity by adding a spatial dimension.

2.3 Remarks

Using this conversion of the problem into a tridiagonal system, we were able to simulate the pendulum system for reasonable parameters, as long as not too many multigrid levels are used. For more information about where to find and run the code, see Section 4.5.

Chapter 3

A Crowd Control Problem

We proceed to a much more difficult and useful class of problems.

3.1 Pedestrian Movement Models

Pedestrian crowds are composed of large numbers of individual agents, each making decisions about their own movement according to their perception of the world around them. Unfortunately mathematical models containing such a level of individual details can be prohibitively difficult to analyze directly. One alternative is mean field analysis, which converts governing equations on motions of individuals, $X(t)$, into mass transport problems, using $\rho(x,t)$ a crowd density function [19, 29]. Intuitively, $\rho(x,t) = 0$ means that the space x and its immediate surroundings are empty at time t and $\rho(x,t) = 1$ means that point x is fully saturated at time t. "Immediate surroundings" here refer to surroundings so immediate that they are several orders of magnitude smaller than the finest grid that our solver uses. We use the phrase only so that ρ can justifiably take values other than 0 and 1.

We similarly define $v(x,t)$ to be the average velocity of the crowd at point x and time t. We can then define the momentum $m = \rho v$, motivated by considering ρ to be proportional to the mass of the crowd near x at t.

This gives the fundamental constraint of crowd modelling problems:

$$\frac{\partial \rho}{\partial t} = -\nabla_x \cdot m. \tag{3.1}$$

This equation encodes the conservation of momentum. We may now set boundaries and objectives with these constraints.

There are two fundamental types of crowd problems, escape and transport, and both are equally amenable to TriMGRIT.

3.1.1 Escape Problems

The first canonical class are escape problems, where a space with initial density distribution ρ_0 is emptied as quickly as possible, is known as an escape problem. We theoretically derived a TriMGRIT system for, but did not have time to practically implement, an escape problem as formulated in [4]. The original paper considers two spatial dimensions; for simplicity, we consider only one. The objective function to clear a crowd as quickly as possible can be defined as

$$I_T(\rho, v) = \frac{1}{2} \int_0^T \int_\Omega F(\rho)|v(x,t)|^2 + E(\rho)\ dx\ dt \tag{3.2}$$

subject to the PDE constraint

$$\partial t\rho + \nabla \cdot (G(\rho)v) = \frac{\sigma^2}{2}\Delta\rho \qquad \rho(x,0) = \rho_0(x) \tag{3.3}$$

with

- $F = \rho$ is the cost of moving through a crowd of density ρ
- $G = \rho$ corresponds to nonlinear mobilities of a person moving through the crows
- $E = \rho^2$ is a cost function which penalizes regions of very high density to actively avoid jams

These functions can be adjusted depending on the specific crowds being studied; for examples from biology or physics see [5, 6, 9, 22, 25]. We would like to formulate the problem in terms of momentum rather than velocity, so we define the flux density by $j(x,t) = G(\rho(x,t))v(x,t)$.

In order to "move" the nonlinearity out of the objective function and into the constraints, we define $w = \sqrt{F(\rho)}v$ and $K = \sqrt{F}$. Then the objective function is

$$J_T(\rho, w) = \frac{1}{2}\int_0^T \int_\Omega |w|^2 + E(\rho)$$

with constraint

$$\partial_t\rho = \frac{\sigma^2}{2}\Delta\rho - \nabla \cdot (K(\rho)w).$$

While we are given initial values $\rho(x,0)$, we are also given boundary conditions representing the maximum amount of people that can leave through the boundary exits at time t. These boundary conditions are

$$-\frac{\sigma^2}{2}\nabla\rho + j = 0$$

which, using our previously defined j value, is

$$-\frac{\sigma^2}{2}\Delta\rho + \frac{G(\rho)}{K(\rho)}w = 0.$$

Let i be the last point in space at which the value of ρ is allowed to vary. We wish to determine the value of ρ at $i+1$, that is, the boundary condition, and do the same for w. To solve for our final ρ and w boundary values, ρ_{i+1} and w_{i+1}, we discretize the boundary condition forward in space and solve for ρ_{i+1}. Then we discretize backwards in space to solve for w_{i+1} with the recently computed ρ_{i+1}.

Similarly, given vectors $\rho_1, \ldots, \rho_n$ and $w_1, \ldots, w_n$, we might want to find boundary conditions w_0 and ρ_0. We omit the algebraic details, but the boundaries are solved similarly with ρ_0 calculated first then used in the computation to get w_0.

After determining the boundaries, the discretized objective function we find by integrating over $0 \le t \le 1$ is

$$J(\rho, w) = \sum_{i=1}^{n}\sum_{j=1}^{m}({w_j^i}^2 + {\rho_j^i}^2)\Delta x\Delta t.$$

As in the pendulum problem, we take the Lagrangian of the above objective function and derive the following KKT equations

$$\begin{aligned} \rho^i \Delta x \Delta t + z^i - S z^{i+1} &= 0 \\ w^i \Delta x \Delta t - R z^{i+1} &= 0 \\ \rho^i - \Phi(\rho^{i-1}, w^{i-1}) &= 0. \end{aligned}$$

Given values for ρ and w at time $i-1$, and for z at time $i+1$, this system could in theory be solved exactly for ρ_i, w_i, and z_i. As stated earlier, we did not have the time to implement this and consequently moved on to other problems.

3.1.2 Transport Problems

Many scenarios arise where crowds of something, whether it be fluids, masses, or even people, need to be transported from distribution μ_0 to position μ_1. In a transport, or Monge-Kantorovich, problem, we are given values for $\rho(x,0)\ \forall x$ and $\rho(x,T)\ \forall x$ for some final time T, representing initial and final configurations for the crowd. There may or may not be initial conditions for m; in the systems which we modeled, there were none. Boundary conditions for m in such problems need not exist.

We focused on the transport problem in [17] with the goals of implementing TriMGRIT and acheiving convergence. The authors, Haber and Horesh, set $T = 1$ with grid domain Ω and define the problem

$$\begin{aligned} \min \qquad & f(m,\rho) = \int_{t=0}^{1} \int_{\Omega} \frac{|m(x,t)|^2}{\rho(x,t)} dx dt \\ \text{s.t.} \qquad & c(m,\rho) = \rho_t + \nabla \cdot m = 0 \end{aligned} \tag{3.4}$$

where $\rho_t = \frac{\partial \rho}{\partial t}$, $\rho(x,0) = \mu_0$, and $\rho(x,1) = \mu_1$. Haber and Horesh use a staggered-grid discretization scheme, staggering ρ in time and m in space. In our case, for a grid system with s space steps and t time steps, ρ has points at $(i, j+1/2)$ for $1 \le i \le s$ and $0 \le j \le t$, while m has points at $(i+1/2, j)$ for $0 \le i \le s$ and $1 \le j \le t$. Crucial to this scheme are the matrices A_s and A_t which average in space and time respectively to give an $s \times t$ grid.

After deriving the discrete form of Equation (3.4), we must (as in our analysis of the forced pendulum) devise a way to solve for the optimal functions m, ρ. TriMGRIT offers two ways to solve this problem. The slightly more ambitious is simply to calculate the Lagrangian in terms of m, ρ, and weights λ. This is a nonlinear function, but we can use Newton's method to approximately calculate solutions in the `triSolve()` function. Ideally, we would plug this into TriMGRIT and solve it in one go.

The second method provided is based on sequential quadratic programming (SQP) [21]. As before, we first calculate the Lagrangian $\mathcal{L}$. We can then start with an estimated ρ, m, and λ and solve a system of equations

$$A \begin{pmatrix} \delta m \\ \delta \rho \\ \delta \lambda \end{pmatrix} = - \begin{pmatrix} \nabla_m \mathcal{L} \\ \nabla_\rho \mathcal{L} \\ \nabla_\lambda \mathcal{L} \end{pmatrix}$$

where the matrix A is a linear approximation of the Karush-Kuhn-Tucker equations for L. We then set $\lambda = \lambda + \alpha \delta\lambda$ and similarly for m and ρ, where the parameter α is determined by a line search algorithm. Haber and Horesh utilized a spatial multigrid algorithm to solve this system and found that this method converges quickly.

In order to investigate the idea of using TriMGRIT as only a component of a larger algorithm, we chose the second approach. We were able to reproduce similar results in a one-dimensional setting.

3.2 Haber and Horesh's Crowd Model

We consider the one-dimensional version of Haber and Horesh's [17] problem, with derivative matrices D_1 in space and D_2 in time, momentum m, and density ρ. The Lagrangian is

$$\mathcal{L} = (A_s m^2)(A_t(1/\rho)) - \lambda^\top (D \left[m^\top, \rho^\top\right] - q) \tag{3.5}$$

where

$$q = \frac{1}{dt} \begin{pmatrix} \mu_0^\top, \underbrace{0, 0, \ldots, 0}_{s(t-1)}, \mu_1^\top \end{pmatrix}^\top \tag{3.6}$$

$D = \begin{bmatrix} D_1 & D_2 \end{bmatrix}$, and the square and inverse operations on m and ρ respectively are performed componentwise. Our goal was apply a SQP method by solving Equation (4.2) from the paper

$$\begin{pmatrix} \hat{A} & D^\top \\ D & 0 \end{pmatrix} \begin{pmatrix} \delta w \\ \delta \lambda \end{pmatrix} = - \begin{pmatrix} \nabla_w \mathcal{L} \\ \nabla_\lambda \mathcal{L} \end{pmatrix} \tag{3.7}$$

and then choosing, using line search, a real value α so that the new solution given by

$$w \leftarrow w + \alpha \delta w \tag{3.8}$$

$$\lambda \leftarrow \lambda + \alpha \delta \lambda \tag{3.9}$$

minimizes the residual in Equation (3.4). Expanding the first row, we can write Equation (3.7) as

$$\begin{pmatrix} 2\operatorname{diag}(A_s^\top A_t(1/\rho)) & 0 & D_1^\top \\ 0 & 2\operatorname{diag}(A_t^\top A_s(m^2))\operatorname{diag}(1/\rho^3) & D_2^\top \\ D_1 & D_2 & 0 \end{pmatrix} \begin{pmatrix} \delta m \\ \delta \rho \\ \delta \lambda \end{pmatrix} = - \begin{pmatrix} \nabla_m \mathcal{L} \\ \nabla_\rho \mathcal{L} \\ \nabla_\lambda \mathcal{L} \end{pmatrix} \tag{3.10}$$

where, again, the inversion and exponentiation operators are applied componentwise.

We have discretized on a grid with s points in space and t in time, with a distance between space points of ds and a distance between time points of dt. Note that in this case ρ has $s(t+1)$ points and m has $(s+1)t$.

3.2.1 Solving the Whole System

Our initial thought was to develop a Schur complement to solve this problem as a simple linear TriMGRIT system; however, there are many cases in which m can be zero. For instance, if a single clump of people in the middle of the crowd splits into two, the total velocity would be zero at the center of the clump leading $2\operatorname{diag}(A_t^\top A_s(m^2))\operatorname{diag}(1/\rho^3)$ to be singular. (Note that the size of $1/\rho^3$ is the same as that of $\dim \rho$ since the operations are componentwise.)

Instead, we aim to solve the system by, rather than finding $\delta\lambda$ first, simply finding δm, $\delta\rho$, and $\delta\lambda$ all at the same time. All the individual blocks in the block matrix that makes up our system of equations are tridiagonal: two are diagonal, three are zero, and the remainder are derivative matrices.

The System to Solve

Define

$$K = \frac{1}{dx}\begin{pmatrix} -1 & 1 & & & & \\ & -1 & 1 & & & \\ & & -1 & 1 & & \\ & & & \ddots & \ddots & \\ & & & & -1 & 1 \end{pmatrix} \in \mathbb{R}^{s\times(s+1)}. \tag{3.11}$$

Define also

$$X = \begin{pmatrix} 1/4 & & & & \\ 1/4 & 1/4 & & & \\ & 1/4 & 1/4 & & \\ & & \ddots & \ddots & \\ & & & 1/4 & 1/4 \end{pmatrix} \in \mathbb{R}^{(s+1)\times s}. \tag{3.12}$$

Then $A_s^\top A_t = \begin{bmatrix} 0 & X & X \end{bmatrix}$ and $A_t^\top A_s = \begin{bmatrix} X^\top & X^\top & 0 \end{bmatrix}$. We may then derive the coefficient matrix P_i of δm_i from the top left block of Equation (3.10), and the coefficient matrix Q_i of $\delta\rho_i$ from the central block:

$$P_i = 2\,\mathrm{diag}(X(1/\rho_i + 1/\rho_{i+1})) \tag{3.13}$$

$$Q_i = 2\,\mathrm{diag}(X^\top(m_i^2 + m_{i-1}^2))\,\mathrm{diag}(1/\rho_i^3) \tag{3.14}$$

where m_0 and m_{n+1} are taken to be 0.

We rewrite Equation (3.10) from above as the following system:

$$P_i\delta m_i + K^\top\delta\lambda_i = -\nabla_{m_i} \qquad 1 \le i \le n \tag{3.15}$$

$$Q_i\delta\rho_{i-1/2} + \frac{1}{dt}\delta\lambda_{i-1} - \frac{1}{dt}\delta\lambda_i = -\nabla_{\rho_{i-1/2}} \qquad 1 \le i \le n+1 \tag{3.16}$$

$$\frac{1}{dt}\delta\rho_{1/2} = -\nabla_{\lambda_0} \tag{3.17}$$

$$-\frac{1}{dt}\delta\rho_{n+1/2} = -\nabla_{\lambda_{n+1}} \tag{3.18}$$

$$K\delta m_i + \frac{1}{dt}\delta\rho_{i+1/2} - \frac{1}{dt}\delta\rho_{i-1/2} = -\nabla_{\lambda_i} \qquad 1 \le i \le n \tag{3.19}$$

Define the $n+2$ "vectors" $v_0, \ldots, v_{n+1}$. Each v_i contains a $\delta\rho_{i-1/2}$, a δm_i, and a $\delta\lambda_i$, except for v_0 which contains only $\delta\lambda_0$ and v_{n+1} which contains only $\delta\rho_{n+1/2}$ and $\delta\lambda_{n+1}$. The vectors v_0 and v_{n+1} correspond to the boundaries of the staggered grid utilized in this problem. Each v_i also has a system of equations associated with it, which will of course draw on values at v_{i-1} and v_{i+1}.

From now on, when we write ρ_i, it will be referring to the ρ that is part of v_i, which is more properly written $\rho_{i-1/2}$.

Vectors v_1 through v_n are governed by the equations

$$P_i\delta m_i + K^\top\delta\lambda_i = -\nabla_{m_i} \tag{3.20}$$

$$Q_i\delta\rho_i + \frac{1}{dt}\delta\lambda_{i-1} - \frac{1}{dt}\delta\lambda_i = -\nabla_{\rho_i} \tag{3.21}$$

$$K\delta m_i + \frac{1}{dt}\delta\rho_{i+1} - \frac{1}{dt}\delta\rho_i = -\nabla_{\lambda_i} \tag{3.22}$$

Vectors v_1 and v_0 are also governed (in the case of v_0, completely governed) by

$$\frac{1}{dt}\delta\rho_1 = -\nabla_{\lambda_0} \tag{3.23}$$

$$Q_1\delta\rho_1 + \frac{1}{dt}\delta\lambda_0 - \frac{1}{dt}\delta\lambda_1 = -\nabla_{\rho_1} \tag{3.24}$$

Vector v_{n+1} is completely governed by

$$-\frac{1}{dt}\delta\rho_{n+1} = -\nabla_{\lambda_{n+1}} \tag{3.25}$$

$$Q_{n+1}\delta\rho_{n+1} + \frac{1}{dt}\delta\lambda_n - \frac{1}{dt}\delta\lambda_{n+1} = -\nabla_{\rho_{n+1}} \tag{3.26}$$

Solving for λ_i in v_i

Recall the equations that govern v_i for $1 \leq i \leq n$:

$$\begin{aligned} P_i\delta m_i + K^\top\delta\lambda_i &= -\nabla_{m_i} \\ Q_i\delta\rho_i + \frac{1}{dt}\delta\lambda_{i-1} - \frac{1}{dt}\delta\lambda_i &= -\nabla_{\rho_i} \\ K\delta m_i + \frac{1}{dt}\delta\rho_{i+1} - \frac{1}{dt}\delta\rho_i &= -\nabla_{\lambda_i} \end{aligned}$$

We may invert P_i, since ρ is never 0. Let us therefore do it, to get

$$\delta m_i + P_i^{-1}K^\top\delta\lambda_i = -P_i^{-1}\nabla_{m_i} \tag{3.27}$$

We substitute this into the final equation, to get

$$\frac{1}{dt}\delta\rho_i = K(-P_i^{-1}K^\top\delta\lambda_i - P_i^{-1}\nabla_{m_i}) + \frac{1}{dt}\delta\rho_{i+1} + \nabla_{\lambda_i} \tag{3.28}$$

We finally substitute this into the second equation, to see

$$dtQ_i(K(-P_i^{-1}K^\top\delta\lambda_i - P_i^{-1}\nabla_{m_i}) + \frac{1}{dt}\delta\rho_{i+1} + \nabla_{\lambda_i}) + \frac{1}{dt}\delta\lambda_{i-1} - \frac{1}{dt}\delta\lambda_i = -\nabla_{\rho_i} \tag{3.29}$$

which simplifies into

$$(-dtQ_iKP_i^{-1}K^\top - \frac{1}{dt}I)\delta\lambda_i = dtQ_iKP_i^{-1}\nabla_{m_i} - Q_i\delta\rho_{i+1} - dtQ_i\nabla_{\lambda_i} - \frac{1}{dt}\delta\lambda_{i-1} - \nabla_{\rho_i} \tag{3.30}$$

Practicalities

In practice, this system was complicated to code and did not converge to a solution. The solutions may have had constant convergence rates, but these were too close to 1 and the algorithm was consistently terminated by hitting the max number of iterations before an acceptable error tolerance was achieved. We thought there may have been a subtle error in our code, and abandoned this scheme as being altogether too conducive to subtle errors in general.

3.2.2 A New Approach: The Regularized Schur Complement

Haber and Horesh solve Equation (3.10) using the Schur complement technique which we introduced previously; the Schur complement here is given by $S = -D\hat{A}^{-1}D^\top$. We would like to invert this using TriMGRIT. To calculate it, first note that

$$S = -\begin{bmatrix} D_1 & D_2 \end{bmatrix} \begin{pmatrix} P^{-1} & \\ & Q^{-1} \end{pmatrix} \begin{bmatrix} D_1^\top \\ D_2^\top \end{bmatrix} = -D_1 P^{-1} D_1^\top - D_2 Q^{-1} D_2^\top. \tag{3.31}$$

We can consider both of these terms separately, getting

$$D_1 P^{-1} D_1^\top = \begin{pmatrix} 0 & & & & \\ 0 & KP_1^{-1}K^\top & & & \\ 0 & 0 & KP_2^{-1}K^\top & & \\ & & & \ddots & \\ & & & & KP_n^{-1}K^\top \quad 0 \\ & & 0 & & \end{pmatrix} \tag{3.32}$$

and

$$D_2 Q^{-1} D_2^\top = \frac{1}{dt^2} \begin{pmatrix} Q_1^{-1} & -Q_1^{-1} & & & & \\ -Q_1^{-1} & Q_1^{-1}+Q_2^{-1} & -Q_2^{-1} & & & \\ & -Q_2^{-1} & Q_2^{-1}+Q_3^{-1} & -Q_3^{-1} & & \\ & & \ddots & & & \\ & & -Q_n^{-1} & Q_n^{-1}+Q_{n+1}^{-1} & -Q_{n+1}^{-1} \\ & & & -Q_{n+1}^{-1} & Q_{n+1}^{-1} \end{pmatrix}. \tag{3.33}$$

This finally gives us our Schur complement. Since P_i and Q_i are diagonal, their inverses can easily be found.

Adding a Regularization Parameter

As described above, Q_i can be ill-conditioned as m_i approaches 0. To counteract this, we add a regularization parameter $\frac{1}{2}\alpha_k\rho^2$ to the objective function. We can now write it as $\int \frac{m^2}{\rho} + \frac{1}{2}\alpha_k\rho^2 \; dx \; dt$, where α_k decreases with each iteration k and approaches 0. (We normally start with $\alpha_1 = 0.0001$ and $\alpha_{k+1} = \alpha_k/2$. We start with this parameter value as an overly-large α_k leaves us with a problem very different to the one we actually want to solve.) We approximate the new objective function by $h^3((A_s m^2)(A_t(1/\rho)) + \frac{1}{2}\alpha_k||\rho^2||_1)$. Immediately, we see that $\nabla_\rho \mathcal{L} = -RA_t^\top A_s(m^2) + D_2^\top \lambda + \alpha_k \rho$.

We must also alter the matrix Q_i, replacing it with the regularized matrix $\tilde{Q}_i = Q_i + \alpha_k I$ with I as the identity matrix. We would like this matrix to be invertible, and, in fact, it is, as long as none of the elements of Q_i are equal to $-\alpha_k$ at iteration k (remember that Q_i is diagonal). Since Q_i is finite, we can always find an $\alpha_k \leq \frac{1}{2}\alpha_{k-1}$ such that this is true. In practice, we found we did not need to worry about the exact value of α_k in this way: the chances of some diagonal element of Q_i taking on any given exact nonzero value are very low compared to the chances of some such element being zero.

With our new $\tilde{Q}_i$ and our new $\nabla_\rho \mathcal{L}$, our system converged even in the case that the solution had $m = 0$ everywhere, previously the worst-case scenario.

3.2.3 Remarks

We ran three different simulations, each of which ran for three distinct solve setups: a normal solve of the system without TriMGRIT or a preconditioner, a TriMGRIT solve with a preconditioner, and a TriMGRIT solve without a preconditioner. Curiously, when comparing the values of the runs with and without a preconditioner, the values of ρ from the two runs differ by almost 0.2 in some cases. This calls into question the accuracy and reliability of the solutions. In real world applications, differences that large in solution values could mean success or failure in projects, or, in our case, an unanticipated (and unwelcomed) density of people. It is possible that increasing the total number of iterations may lead to a more exact result.

We visualized the ρ output of the three runs and the respective contour plots, as shown in Figures 3.1-3.3. While all three have the same general shape, we find that the solve that uses TriMGRIT without the preconditioner (Figure 3.2) introduces some oscillatory errors in the temporal dimension.

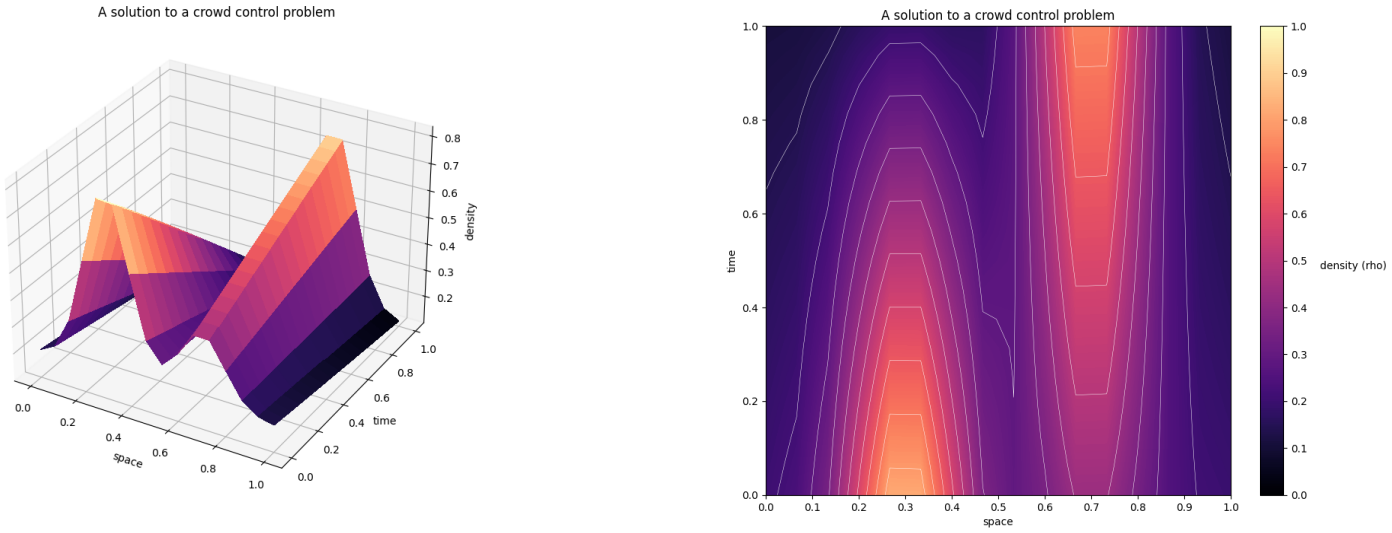

Figure 3.1: Visualization (left) and contour (right) graph of a non-preconditioned simple solve of the Haber and Horesh crowd model

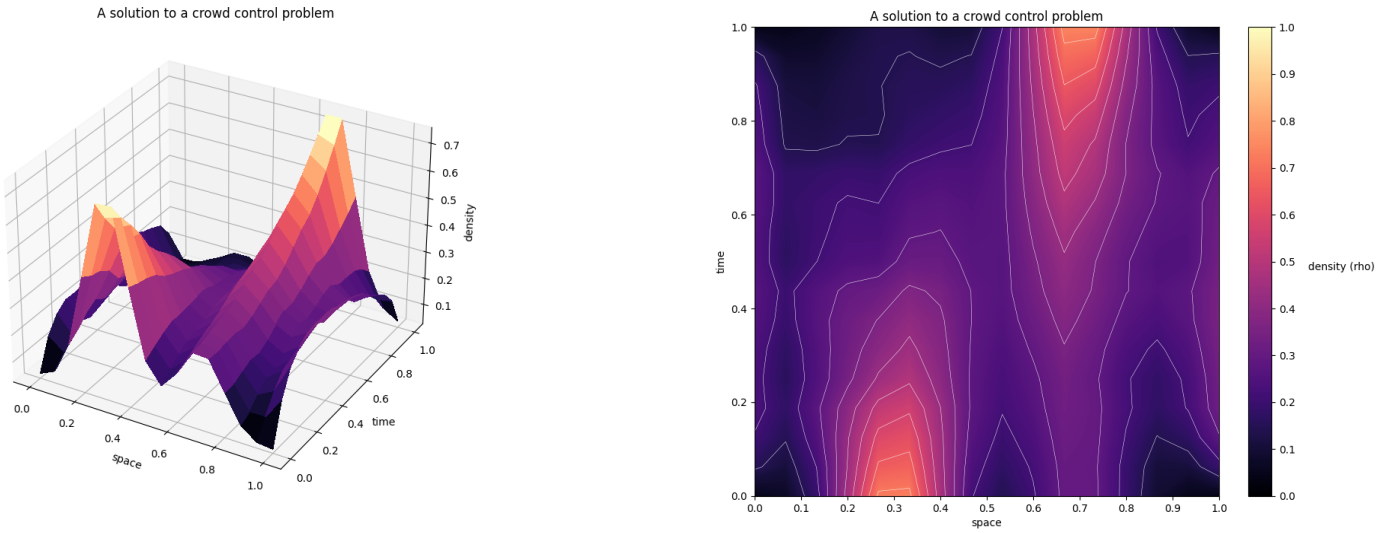

Figure 3.2: Visualization (left) and contour (right) graph of a non-preconditioned TriMGRIT solve of the Haber and Horesh crowd model

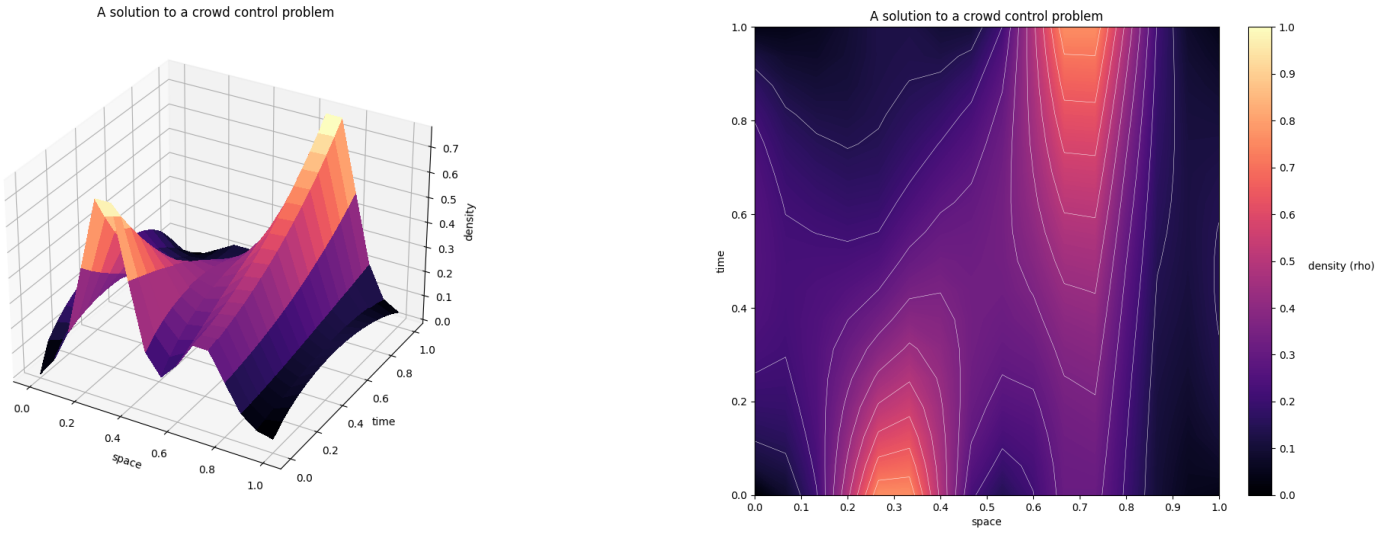

Figure 3.3: Visualization (left) and contour (right) graph of a preconditioned TriMGRIT solve of the Haber and Horesh crowd model

Speed of Convergence

Although most of our tests were done on our own computers, we were able to use LLNL's Quartz cluster to scale up to 36 parallel tasks at once without issues. This was appreciably faster: on one thread of an Intel i7-8550U, capable of Turbo Boost up to 4.0GHz, solving a system with 100 points in space, 100 points in time, 400 fine relaxations, 400 coarse relaxations, and exactly one coarse grid required 37.7 seconds to complete the first five

iterations. Parallelized into 36 processes running on Intel Xeon E5-2695 v4 (36 threads @ 2.1GHz), the same operation took 1.7 seconds.

While using more than one coarse grid could have sped up convergence dramatically, in practice our implementation diverged as soon as we added in a second coarsening. This was a common pattern in the TriMGRIT code we wrote.

Restricting depth to two grids, using the results from one of the supercomputer runs, we calculated the convergence ratios. The i-th convergence ratio c_i is defined as the norm of the residual r_i divided by that of the residual r_{i-1}. (As a matter of convenience, we set $c_0 = 1$.) Plotting $1 - c_i$ for the first 3000 convergence ratios on a log-log chart gives Figure 3.4.

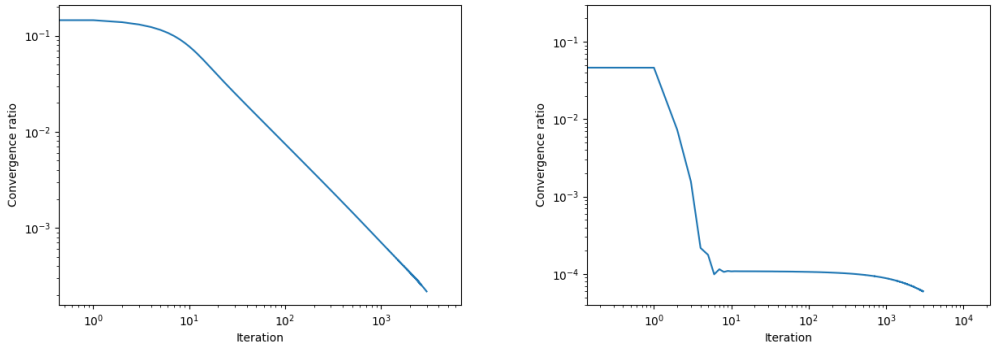

Figure 3.4: Log-log graphs of $1 - c_i$ against i from different runs of TriMGRIT trying to solve similar problems. Larger convergence ratios are better.

On the left, after holding fairly steady for the first few iterations, the convergence ratio slopes down at a fairly constant slope of -1. On a log-log plot, this implies that $c_i \propto 1 - i^{-1}$. This is by no means a universal law: on the right, we see a situation where the convergence ratio has some sharp increases but, outside of those, holds fairly steady.

This concludes the practical work we did on solving problems in TriMGRIT. We now turn to more general contributions to the algorithm itself.

Chapter 4

Miscellaneous Theoretical Research and Practical Contributions

In this chapter we summarize a number of topics which do not fit neatly into other sections of the paper. In Sections 4.1 and 4.2 we examine the way that the actual and ideal coarse-grid operators, both in MGRIT and TriMGRIT. In Section 4.3 we describe a combination of Tri/MGRIT and a line search routine, and in Sections 4.4 and 4.5 we summarize our progress in implementing the methods described throughout this report.

4.1 Rescaling and Coarse-Grid Operators

Given a $(\alpha, \beta) \in \mathbb{R}^2$ pair, we consider the constrained optimization problem

$$\min_{u,v} \frac{1}{2} \int_0^T \beta u^2 + \alpha v^2 dt$$

subject to the constraints $u' = v$ and $u(0) = u_0$. Following all of the algebraic techniques used above, the ideal coarse grid operator for this problem is the same as the rediscretized coarse grid operator for the closely related problem of

$$\min_{u,v} \frac{1}{2} \int_0^T \beta_\Delta u^2 + \alpha_\Delta v^2 dt$$

for $\alpha_\Delta = \frac{4\chi+2\gamma}{4\chi+\gamma}\alpha = \frac{4+2\beta\gamma}{4+\beta\gamma}\alpha$ and $\beta_\Delta = \frac{2\chi+\gamma}{2\chi^2} = \frac{2+\beta\gamma}{2}\beta$. We use the same convention as before, that $\gamma = \frac{\Delta t^2}{\alpha}$, and introduce $\chi = \frac{1}{\beta}$.

Our interest is in a geometric interpretation for these transformations. To solve a problem of this sort (finding a solution to a tridiagonal matrix equation), our relaxation operator A_{RAP} (recall from Chapter 1) is given by a block cyclic reduction process. We start with three equations of the form

$$-W_{i-1}x_{i-2} + C_{i-1}x_{i-1} - E_{i-1}x_i = 0 \tag{4.1}$$

$$-W_i x_{i-1} + C_i x_i - E_i x_{i+1} = 0 \tag{4.2}$$

$$-W_{i+1}x_i + C_{i+1}x_{i+1} - E_{i+1}x_{i+2} = 0. \tag{4.3}$$

We wish to eliminate x_{i-1} and x_{i+1}, which corresponds to an exact coarsening onto a grid which does not contain x_{i-1} or x_{i+1}. Normally, this is done by multiplying the first equation through by $W_i C_{i-1}^{-1}$ and the bottom one through by $E_i C_{i+1}^{-1}$, and adding all three together. This exactly Equation (1.7). We can make two key assumptions here:

1. W, C, and E are in fact independent of i.

2. $W = E$.

We can now rewrite our equations as

$$-Wx_{i-2} + Cx_{i-1} - Wx_i = 0 \tag{4.4}$$
$$-Wx_{i-1} + Cx_i - Wx_{i+1} = 0 \tag{4.5}$$
$$-Wx_i + Cx_{i+1} - Wx_{i+2} = 0. \tag{4.6}$$

We can solve this by multiplying the central equation by CW^{-1} and adding all three equations together to give the exact coarse-grid operator $A_{CW} = \begin{bmatrix} -W & CW^{-1}C - 2W & -W \end{bmatrix}$.

In the context of this particular problem, we set $W = E = \frac{1}{\beta}$ and $C = \frac{2}{\beta} + \gamma$. This gives $A_\Delta = \frac{1}{\Delta t}\begin{bmatrix} -\frac{1}{\beta} & \frac{2}{\beta} + 4\gamma & -\frac{1}{\beta} \end{bmatrix}$ and $A_{CW} = \frac{1}{\Delta t}\begin{bmatrix} -\frac{1}{\beta} & \frac{2}{\beta} + 4\gamma + \gamma^2\beta & -\frac{1}{\beta} \end{bmatrix}$. Since γ^2 is on the order of $(\Delta t)^4$, it is not unreasonable to call A_Δ an approximation to A_{CW}.

We note that A_{CW} looks nothing like A_{RAP}. However, if we take Equation (1.7) and make the substitutions from earlier, we see

$$A_{RAP} = \begin{bmatrix} -WC^{-1}W & C - 2WC^{-1}W & WC^{-1}W \end{bmatrix}$$

which, if we multiply every element on the right by $W^{-1}C$, is A_{CW} exactly. This will be important later.

We now wish to calculate some α_Δ and β_Δ such that, when we calculate A_Δ with α_Δ and β_Δ, we get A_{CW}. By comparing the outer elements of the two stencils, we see that $\beta_\Delta = \beta$. This leaves the inner ones, from which we get

$$\begin{aligned} 4\gamma_\Delta &= 4\gamma + \gamma^2\beta \\ 4\frac{\Delta t^2}{\alpha_\Delta} &= 4\gamma + \gamma^2\beta \\ \alpha_\Delta &= \frac{4\gamma}{4\gamma + \gamma^2\beta}\alpha. \end{aligned}$$

In this form, it becomes much easier to see what α_Δ is actually doing: it is adding a γ^2-order correction, so that, even after the approximation process that generates A_Δ from A_{CW}, such a correction is still there.

This determines $\alpha_\Delta = \frac{4}{4\gamma+\gamma^2\beta}\gamma\alpha$ and $\beta_\Delta = \beta$. These do not look like what we were expecting: we wanted $\alpha_\Delta = \frac{4+2\beta\gamma}{4\gamma+\gamma^2\beta}\gamma\alpha$ and $\beta_\Delta = \frac{2+\beta\gamma}{2}\beta$. However, remember that, to get from A_{RAP} to A_{CW}, we had to multiply through by $W^{-1}C = 2 + \beta\gamma$. If we multiply our new α_Δ and β_Δ by this expression, and then divide through by 2, we get exactly the α_Δ and β_Δ we wanted.

We are allowed to do this sort of multiplication because all it is doing is going from an equation of the form $Ax = 0$ to one of the form $\lambda Ax = 0$ for some scalar λ.

We thereby motivate the study of the transformation $\alpha \mapsto \frac{4}{4+\Delta t^2\beta/\alpha}\alpha$ or, rearranging, $\alpha \mapsto \frac{4\alpha^2}{4\alpha+\Delta t^2\beta}$. As Δt approaches 0, this approaches α as we would expect. Moreover, as Δt grows, the discontinuity at $\alpha = \beta = 0$, which disappears in the limit, becomes more pronounced. Running

```
Manipulate[
Plot3D[4 a^2/(4 a + dt^2*b), {a, -10, 10}, {b, -10, 10}], {dt, 0, 5}]
```

in Mathematica will produce a good visualization, as in Figure 4.1.

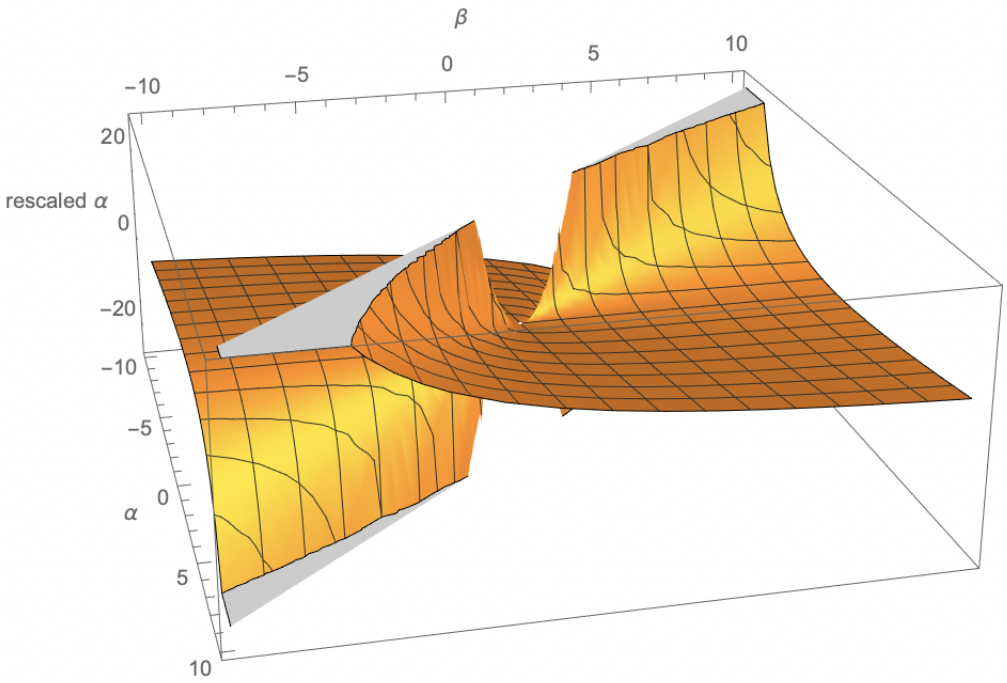

Figure 4.1: Example output from Mathematica code, plotting the form of the transformation from α and β to the α with estimated coarse-grid operator equal to the exact coarse-grid operator of the original α.

4.1.1 A More General Alternative Block Cyclic Reduction

We return to our most general tridiagonal equations, this time assuming only that they do not vary with i:

$$-Wx_{i-2} + Cx_{i-1} - Ex_i = 0 \tag{4.7}$$
$$-Wx_{i-1} + Cx_i - Ex_{i+1} = 0 \tag{4.8}$$
$$-Wx_i + Cx_{i+1} - Ex_{i+2} = 0. \tag{4.9}$$

Once again, we wish to eliminate x_{i-1} and x_{i+1}. However, this time, we begin by multiplying the second one on the left by CW^{-1} and adding the first two, to get

$$-Wx_{i-2} + (CW^{-1}C - E)x_i - CW^{-1}Ex_{i+1} = 0. \tag{4.10}$$

We then multiply the third one on the right by $W^{-1}E$ and add it to our most recent equation to get

$$-Wx_{i-2} + (CW^{-1}C - 2E)x_i - EW^{-1}Ex_{i+2} = 0, \tag{4.11}$$

which gives us an exact coarse-grid operator

$$A_{ECW} = \begin{bmatrix} -W & CW^{-1}C - 2E & EW^{-1}E \end{bmatrix}. \tag{4.12}$$

If C^{-1} is difficult to calculate, we can use W^{-1} (or, with a similar approach, E^{-1}) in our coarse-grid operators instead. This also allows for the possibility of better estimated coarse-grid operators, by approximating W^{-1}, which is often easier than approximating C^{-1}.

Furthermore, this method can be used to allow for exact coarse-grid operators to be generated even in certain nonlinear situations. Consider the differential equation $u_t = u_x + N(u)$, where N is a nonlinear operator on u that contains no derivatives – that is, its discretization in any scheme is just $N(u_j^i)$. We can discretize this into a "matrix" (really a nonlinear operator on the discretized u) where N is only in the central part. Then, following a similar method to that in chapter 5 of the 2019 LLNL/RIPS report [12], we may split that matrix into two, one linear and one nonlinear, and invert only the linear portion.

4.2 Optimizing the Choice of Coarse-Grid Operator

We have been interested in improving the choice A_Δ of coarse grid operator, to one more closely mirroring the action of A_{RAP} the ideal. Of course in the extreme case that $A_\Delta = A_{RAP}$ we perform an exact solve and the algorithm converges in a single step. Unfortunately this is prohibitively costly in computation and we must settle with a close guess, often a rediscretization of the problem onto the coarse grid. We initially began by following the line of thought in [7], which studies this process for MGRIT. That paper begins by citing the result shown in [8] and [18]: the error developed in a single iteration of MGRIT may be estimated using the difference in eigenvalues between the ideal coarse grid operator and the one used in practice. The bound becomes weaker (and therefore a better approximation is desired) as these eigenvalues approach 1 in modulus. Fortunately the matrices involved are often circulant, so their eigenvalues are easily accessible through the Discrete Fourier Transform, and the authors designed a simple weighted optimization problem which allowed numerical discovery of a strong approximation to the ideal coarse operator. We have done a few experiments to find the weight function which gives the absolute best coarse grid operator (in particular running a gradient descent algorithm across a discretized weight function), but have not come across any particular improvements.

We are investigating to what extent these results can be lifted to TriMGRIT, beginning with an analogue of the theory developed in [7] on the error propogation for MGRIT, but for TriMGRIT. Unfortunately, this has proven rather difficult, mainly due to the difficulty in solving (algebraically) a tridiagonal system compared to a subdiagonal system. MGRIT's natural application is to problems of the form

$$u_i = \Phi u_{i-1} + g_i$$

whose solutions have a natural inductive form using the operator Φ^m. On the other hand the tridiagonal systems we are interested in take the form

$$u_i = \Phi(u_{i-1}, u_{i+1}) + g_i$$

and these are much less straightforward to solve explicitly. We believe that developing error analysis for TriMGRIT will depend on understanding solution propagation for these equations. We are hopeful that this may be possible, at least for the coarsening factor $m = 2$ case.

4.3 Combinations of TriMGRIT and Line Search

Like all iterative algorithms, TriMGRIT is prone to starting out in a fairly good place, trying to find a better one, overshooting, and landing further away – possibly by amplifying

certain errors from instabilities that were already present in the starting state. In the worst case, this can cause divergence.

We wondered, if Tri/MGRIT (TriMGRIT or MGRIT) got us to a certain spot after one iteration, could we add a line search to "nudge" the values closer to the solution and avoid possible divergence before the next Tri/MGRIT iteration? Say we want to solve a system $Ax = b$. Tri/MGRIT iteratively generates a sequence $x_1, x_2, x_3 \ldots$ which we would hope converges to x. The algorithm normally looks like this:

1. Start with some guessed solution x.
2. Use Tri/MGRIT on x to get x'.
3. Set $x \leftarrow x'$ and repeat.

Instead, we use the following algorithm:

1. Start with some guessed solution x.
2. Use Tri/MGRIT on x to get x'.
3. Find $\alpha = \underset{\tilde{\alpha}}{\operatorname{argmin}} ||A(\tilde{\alpha}x + (1-\tilde{\alpha})x') - b||$.
4. Set $x \leftarrow \alpha x + (1-\alpha)x'$ and repeat.

This algorithm seems similar in spirit to a simplification of the Generalized Minimal Residual method (GMRES, given in [24]). GMRES, instead of performing a one-dimensional search over the the space spanned by the two most recent iterations, performs a minimization over the space spanned by all candidate solutions found up until the present iteration.

We used a standard golden section search, as described in [23], to find α. This was effective overall and, in some cases, took problems for which TriMGRIT had previously diverged and shepherded them to convergence. However, this took quite a long time in practice, and computational times were often longer than the rest of the algorithm put together. While a line search is inherently a computationally-intensive activity, this slowdown may also have been due to the fact that each calculation of a value for the objective function was in parallel and required communication between CPU cores.

We developed a much cheaper quadratic optimization method which could be used in the case that A is a linear operator. Let $\vec{w}$ be the vector given by TriMGRIT, that is, $x' - x$. Then the problem given in Step 3 above is one of finding the α such that residual at $x + \alpha\vec{w}$ is as minimized. In the linear setting, the function that takes α to the residual squared (the square of the distance between a point on the line given by α and a fixed point off the line) is quadratic in α. This enables the following algorithm:

1. Calculate the residual r_0 at x and r_1 at x'. (TriMGRIT has already calculated these, so in theory we should be able to implement this step as just reading from a few variables. In practice, it proved easier just to calculate them again.)
2. Calculate the residual $r_{1/2}$ at $(x + x')/2$.
3. Calculate $a = 2r_1^2 + 2r_0^2 - 4r_{1/2}^2$ and $b = -r_1^2 - 3r_0^2 + 4r_{1/2}^2$. (These formulae can easily be derived by defining $f(\alpha) = a\alpha^2 + b\alpha + c$, setting $f(0) = r_0^2$, $f(1) = r_1^2$, and $f(1/2) = r_{1/2}^2$, and solving for a, b, and c.)
4. Set $\alpha = -\frac{b}{2a}$. (Since $f(\alpha)$ represents a distance, it is always positive, so it has a global minimum here.)

In tests, this algorithm was appreciably faster than the golden section search and produced similar results. Interestingly, these results were not identical, and the small difference in the values of α in the first iteration led to much larger differences in the next ones. TriMGRIT still eventually converged to the same value.

We reiterate, however, that we can currently only justify the use of the quadratic search in a completely linear setting.

4.4 Contributions to XBraid Code

While working on our project, we often found things we wanted to change with the XBraid code. Some of these changes were small details, such as updating the example scripts for visualizing outputs from the example programs[1], fixing consistency with the output of example 4 from the examples folder[2] [3], and noticing small bugs in the trimgrit branch of the repository[4][5]

In addition to these issues, there were a few larger changes that were suggested for the XBraid code. First, we noticed that a small change to the C++ interface could simplify some of the user written code[6] (this change has yet to be reviewed), with two potential implementations being given to solve this problem. Finally, we also wrote an interface in C++ for the TriMGRIT code in XBraid[7] which was very useful for our own code, which was written in C++.

4.5 Our Code

Our code for the crowd model can be found at the Github repository at `https://github.com/Torrencem/crowd_xbraid`. It uses (as a git submodule) a branch of our fork of the XBraid code[8] which contains the C++ interface to the TriMGRIT code, as well as any of the bug fixes we needed for our code in particular. Our code for the pendulum problem can be found in a branch of our fork of the XBraid repository[9].

The README in our repository gives specific instructions for downloading and running the code. The library we use for sparse matrix manipulation and sparse linear system solving is Eigen, written in C++, which is also included as a submodule to our main project repository. Also included in the repository are the C and C++ interfaces to the line search module described in Section 4.3.

[1] `https://github.com/XBraid/xbraid/pull/44`
[2] `https://github.com/XBraid/xbraid/pull/42`
[3] `https://github.com/XBraid/xbraid/pull/47`
[4] `https://github.com/XBraid/xbraid/issues/58`
[5] `https://github.com/XBraid/xbraid/issues/59`
[6] `https://github.com/XBraid/xbraid/issues/48`
[7] `https://github.com/Torrencem/xbraid/tree/trimgrit_cxx`
[8] `https://github.com/Torrencem/xbraid/tree/ipam_2020`
[9] `https://github.com/Torrencem/xbraid/tree/ipam_pendulum/examples`

Chapter 5

Conclusion and Further Research

5.1 Conclusion

While, once a problem is properly specified, TriMGRIT can solve it very efficiently by taking advantage of massively parallel architecture, specifying problems in the correct form is nontrivial. In particular, without careful strategy, even a technically correctly-specified problem can diverge in some cases or end up amplifying small errors.

Furthermore, TriMGRIT is not well-understood from a theoretical perspective. This is to be expected, given that it is a comparatively new algorithm, but means that it is hard to do things such as choose provably-effective operators. The effectiveness of our line search demonstrates that, in some cases at least, further processing on intermediate TriMGRIT results can make previously divergent schemes converge, but we do not understand why this should happen in those cases in particular.

We have made contributions to all of these areas. Through our studies of the pendulum and crowd problems, we have provided more examples of, and possibly slightly extended existing methods for, translating pure mathematics into TriMGRIT. Through our work on the XBraid code, we have made TriMGRIT more robust, providing a C++ interface and the option to perform either a line or a quadratic search on the intermediate solutions. Finally, although we have not been able to derive any explicit formulae for the behaviour and error of TriMGRIT, we have reviewed several possible approaches which could lead to some useful analysis later on. We have also identified reasons why a few of the most obvious ideas do not in fact work as easily as they could.

In our tests, anecdotal though they are, TriMGRIT was able to take advantage of parallelism to a great degree. In view of this, and the ever-increasing need for parallel-computing algorithms as discussed in the introduction, we believe that developing rigorous theories of and practical guidelines for the TriMGRIT algorithm, encompassing the concepts discussed here as well as, probably, completely new ones, make for important open problems.

5.2 Recommendations for Further Research

This project leaves several open avenues for continued work.

- Our results from Section 4.3 demonstrate that, while TriMGRIT often gives a good direction to move in, it is not always effective at telling us how far in that direction to move. Our line search code is general enough to be added onto any problem that Tri/MGRIT can solve, and anecdotally it seems to work well in many cases, but

rigorous research into its effectiveness or lack thereof, particularly given that it often more than doubles the amount of time required for one TriMGRIT iteration, would be welcome.

This goes double for the three-point quadratic minimization procedure. It seems to produce results similar to, but not equal to, those of the golden section search, but we have no proof that this would be the case in all but the most linear situations. We do not know to what extent it would make a viable replacement to the line search in general.

- TriMGRIT is a natural algebraic extension of MGRIT, but in some cases the gap between theoretical results for the two is very large. Since TriMGRIT is formally a strict generalization of MGRIT one ought to recover all error analysis for MGRIT by taking analysis for TriMGRIT and restricting one of the off-diagonals to be 0. Unfortunately, we have not found a direct analogue of any of the many error analysis techniques available for MGRIT (whether take from the Local Fourier Analyis, algebraic, or other perspective). This makes one of our central goals - improving the convergence of TriMGRIT - difficult to consider outside of experimental observations. We have made an honest attempt at lifting the analysis from MGRIT onto TriMGRIT, but made little progress.

- The very basic analysis, which does not stretch beyond anecdotal at this point, of the changes in conversion rate from Section 3.2.3 seems to imply the possibility of analyzing TriMGRIT's behavior in terms of that of a problem's conversion factors. We do not know what this might lead to, but it could provide a new way to look at error and convergence analysis.

- Section 4.1 looks on paper like it could lead to new ways of calculating and choosing coarse-grid operators for TriMGRIT, although we have not had time to test any of it in the wild.

Appendix A

Abbreviations

GMRES Generalized Minimal Residual method.

IPAM Institute for Pure and Applied Mathematics. An institute of the NSF, located at UCLA.

KKT Karush-Kuhn-Tucker, as in "the KKT equations" or "the Karush-Kuhn-Tucker equations."

LLNL Lawrence Livermore National Laboratory.

MGRIT MultiGrid Reduction In Time.

NSF National Science Foundation.

RIPS Research in Industrial Projects for Students. A regular summer program at IPAM, in which teams of undergraduate (or fresh graduate) students participate in sponsored team research projects.

SQP Sequential Quadratic Programming.

TriMGRIT Tridiagonal MultiGrid Reduction In Time.

Tri/MGRIT Either TriMGRIT or MGRIT.

UCLA The University of California, Los Angeles.

Selected Bibliography Including Works Cited

[1] *Center for applied scientific computing.* `https://computing.llnl.gov/casc`. Accessed: 10 Aug 2020.

[2] *Lawrence livermore national laboratory.* `https://www.llnl.gov`. Accessed: 10 Aug 2020.

[3] M. Avriel, *Nonlinear programming : analysis and methods*, Dover Publications, Mineola, NY, 2003.

[4] M. Burger, M. Francesco, P. Markowich, and M.-T. Wolfram, *Mean field games with nonlinear mobilities in pedestrian dynamics*, Discrete and Continuous Dynamical Systems - Series B, 19 (2013).

[5] M. Burger, M. Francesco, J.-F. Pietschmann, and B. Schlake, *Nonlinear cross-diffusion with size exclusion*, SIAM Journal on Mathematical Analysis, 42 (2010), pp. 2842–2871.

[6] M. Burger, B. Schlake, and M.-T. Wolfram, *Nonlinear poisson-nernst-planck equations for ion flux through confined geometries*, Nonlinearity, 25 (2012), pp. 961–990.

[7] H. De Sterck, R. D. Falgout, S. Friedhoff, O. A. Krzysik, and S. P. MacLachlan, *Optimizing mgrit and parareal coarse-grid operators for linear advection*, arXiv:1910.03726 [cs, math], (2020). arXiv: 1910.03726.

[8] V. A. Dobrev, T. Kolev, N. A. Petersson, and J. B. Schroder, *Two-level convergence theory for multigrid reduction in time (mgrit)*, SIAM Journal on Scientific Computing, 39 (2017), p. S501–S527.

[9] L. Dyson, P. Maini, and R. Baker, *Macroscopic limits of individual-based models for motile cell populations with volume exclusion*, Physical review. E, Statistical, nonlinear, and soft matter physics, 86 (2012), p. 031903.

[10] C. Engwer, R. D. Falgout, and U. M. Yang, *Stencil computations for pde-based applications with examples from dune and hypre*, Concurrency and Computation: Practice and Experience, 29 (2017). LLNL-JRNL-681537.

[11] R. Falgout et al., *Xbraid repository.* `https://github.com/XBraid/xbraid`, 2018.

[12] R. Falgout, I. Meyers, J. Munar, E. Neville, and T. Overman, *A parallel-in-time multigrid approach to constrained optimization*, (2019).

[13] R. D. Falgout, S. Friedhoff, T. V. Kolev, S. P. MacLachlan, and J. B. Schroder, *Parallel time integration with multigrid*, SIAM J. Sci. Comput., 36 (2014), pp. C635–C661. LLNL-JRNL-645325.

[14] C. A. J. Fletcher, *Computational Galerkin Methods*, Springer Berlin Heidelberg, Berlin, Heidelberg, 1984.

[15] G. Grätzer, *More math into LaTeX*, Springer, 4. ed ed., 2007.

[16] S. Günther, N. R. Gauger, and J. B. Schroder, *A non-intrusive parallel-in-time approach for simultaneous optimization with unsteady pdes*, Optimization Methods and Software, 34 (2019), pp. 1306–1321.

[17] E. Haber and R. Horesh, *A multilevel method for the solution of time dependent optimal transport*, Numerical Mathematics: Theory, Methods and Applications, 8 (2015), p. 97–111.

[18] A. Hessenthaler, B. S. Southworth, D. Nordsletten, O. Röhrle, R. D. Falgout, and J. B. Schroder, *Multilevel convergence analysis of multigrid-reduction-in-time*, SIAM Journal on Scientific Computing, 42 (2020), p. A771–A796.

[19] S. Hoogendoorn and P. Bovy, *Pedestrian route-choice and activity scheduling theory and models*, Transportation Research Part B: Methodological, 38 (2004), pp. 169 – 190.

[20] P. M. Kielstra, *Civet repository*. `https://github.com/PMKielstra/Civet`, 2020.

[21] W. Nocedal, *Numerical optimization*, Springer, New York, 2006.

[22] K. Painter and T. Hillen, *Volume-filling and quorum-sensing in models for chemosensitive movement*, Can. Appl. Math. Q., 10 (2002), pp. 501–543.

[23] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Section 10.2: Golden Search in One Dimension*, Cambridge University Press, 3rd ed ed., 2007.

[24] Y. Saad, *Iterative methods for sparse linear systems*, SIAM, Philadelphia, 2003.

[25] M. Simpson, B. Hughes, and K. Landman, *Diffusion populations: Ghosts or folks*, Australasian Journal of Engineering Education, 15 (2009), pp. 59–68.

[26] G. Strang, *Computational science and engineering*, Wellesley-Cambridge Press, 2. print ed., 2012.

[27] M. Torrence, M. Kielstra, C. Hills, and D. King, *Xbraid crowd model*. `https://github.com/Torrencem/crowd_xbraid`, 2020.

[28] ——, *Xbraid fork repository*. `https://github.com/Torrencem/xbraid`, 2020.

[29] J. van den Berg, S. Patil, J. Sewall, D. Manocha, and M. Lin, *Interactive navigation of multiple agents in crowded environments*, 01 2008, pp. 139–147.

[30] F. Zhang, *The Schur complement and its applications*, Springer, New York, 2005.