Mathematical Modelling and Analysis of Several Diffusive Processes

by

M. Brutz

B.S., The Ohio State University, 2008

M.S., University of Colorado Boulder, 2011

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Applied Mathematics

2014

This thesis entitled:
Mathematical Modelling and Analysis of
Several Diffusive Processes
written by M. Brutz
has been approved for the Department of Applied Mathematics

_____

Prof. Tom Manteuffel

_____

Prof. François Meyer

_____

Prof. Harihar Rajaram

Date _____

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.

Brutz, M. (Ph.D., Applied Mathematics)

Mathematical Modelling and Analysis of

Several Diffusive Processes

Thesis directed by Prof. Prof. Tom Manteuffel

The underlying theme of this research is using numerical methods to develop computationally efficient algorithms for three separate problems driven by diffusive processes. The problems under consideration are: contaminant dispersal through fracture networks, modelling the flow of glacial ice, and community detection on networks.

A common feature of containment facilities for nuclear waste is to use expansive geological formations as an added barrier to contaminant dispersal in the event of a leak. Although these formations are generally comprised of dense rock that is difficult to penetrate, fractures within them provide a potential means for contaminants to rapidly transport across the barrier. The typical width of such fractures is only on the order of millimeters whereas the typical scale of interest for contaminant transport is on the order of kilometers. When particle tracking methods are used to simulate the contaminant dispersal in fracture networks, this disparity of scales severely restricts maximum time step sizes because features at the millimeter scale need to be resolved. Our contribution to this problem is developing a coarse scale particle tracking method that allows for substantially larger time steps when particles are navigating straight fractures.

With global warming comes concerns as to how the changing temperature will impact glacial systems and their contribution to sea level rise. On glacial scales, ice behaves as a very slowly moving non-Newtonian fluid, and the primary problem for numerically simulating the evolution of ice masses comes with Glen's flow law for the effective viscosity. The flow law is empirically based, and its simple form has proven useful for analytical calculations. However, its simple form also allows for the effective viscosity to become unbounded in regions of low strain rate, and has proven to be very problematic for numerical simulations. Our contribution to this problem is re-examining

the datasets the flow law was originally based on to develop an alternative model that fits the data with comparable accuracy, but without the problematic singularity.

When working with networks that represent real world systems, a common feature of interest is to find collections of vertices that form communities. Because the word "community" is an ambiguous term, our interpretation is that it is necessary to quantify what it means to be a community at a minimum of three scales for any given problem. These scales are at the level of: individual nodes, individual communities, and the network as a whole. Although our work focuses on detecting overlapping communities in the context of social networks, our primary contribution is developing a methodology that is highly modular and can easily be adapted to target other problem-specific notions of community.

Dedication

Dedicated to my parents, Rebecca Miller, and Howard Snooks.

Thank you for being my life jacket. I would have drowned without you.

Acknowledgements

I would like to express my special appreciation and thanks to my adviser, Prof. François Meyer. Thank you for your encouragement to explore new paths, and for being the beacon that guided me back whenever I got lost in the woods. Your dedication to your students and your work has been an inspiration. I would also like to thank Prof. Tom Manteuffel and Prof. Harihar Rajaram; this thesis would not have been possible without your patience, understanding, and guidance. To all of you, a heart felt thank you.

Contents

Chapter

Tables

Table

Figures

Figure

Chapter 1

Introduction

Diffusion is the net movement of a quantity from a region of high concentration to a region of low concentration, and diffusive processes play a role in many problems of practical interest. In this thesis, we analyze three such problems: contaminant dispersal, glacial ice flow, and community detection. For each problem, we develop a computationally efficient method to either solve the problem directly, or to augment other existing solution methods. A common thread in how we approach each of these problems is to consider what the process we are trying to capture should look like at multiple scales. Considering this highlights when it is important to capture the fine-grained details of the problem, and when doing so unnecessarily complicates the problem.

1.1 Contaminant Dispersal

Contaminant transport in fractured rock masses is relevant in the context of protecting groundwater resources and safety assessments of nuclear waste repositories, several of which are being sited in subsurface geologic media [57]. Constraining the precise geometry of fracture networks, accounting for flow/transport properties of individual fractures, and capturing fracture-matrix diffusive exchanges are the primary difficulties in modelling contaminant transport in fractured rock.

Approaches to modelling contaminant transport in fractured rock include equivalent porous-medium, dual or multi-continuum and discrete fracture network (DFN) models. Of these approaches, the DFN models are the most computationally demanding since they represent the underlying fracture network at high resolution. To mitigate the computational demand, particle tracking

methods are often employed to determine the time evolution of concentration distributions in such networks, as these methods are meshfree and simple to parallelize.

Various approaches to particle tracking for transport in fractures have been developed [74, 20, 25, 21], some of which account for fracture-matrix interactions [21, 9, 18]. The most efficient of these approaches typically track particles based on the residence time distributions of particles in individual fractures: the time-domain random walk (TDRW) approach of Bodin [9] assumes planar fractures with uniform velocity within each fracture, and the particle residence time in a fracture-matrix unit is calculated from the analytical solution of Maloszewski and Zuber [48] for a linear fracture. The Lagrangian approach of Cvetkovic et al. [21] accounts for spatial variability in velocity field within fractures, and is based on single fracture-scale retention functions whose form was derived from numerical simulations [18].

Although there are many variations of particle tracking methods for solving contaminant dispersal problems, our work is most closely tied to what we call the standard Fokker-Planck method (SFPM), which we describe here. If A and B are taken as constants, and f represents a square integrable probability density function (pdf) evolving in time, the Fokker-Planck Equation can be written as,

$$\frac{\partial f(\mathbf{x},t)}{\partial t} = -\nabla \cdot (A\ f(\mathbf{x},t)) + \nabla \cdot (B\ \nabla f(\mathbf{x},t)). \tag{1.1}$$

Note that aside from the interpretation of what the variables represent, Equation (1.1) is identical to a standard advection-diffusion equation. However, if $\mathbf{X}_t$ represents a random variable associated with f and $\mathbf{W}_t$ a standard Wiener process, then this random variable evolves in time according to the stochastic ordinary differential equation (sde),

$$d\mathbf{X}_t = A\ dt + \sqrt{2B}\ d\mathbf{W}_t. \tag{1.2}$$

This offers an alternative method to calculating the time evolution of the probability density function. Rather than deal directly with Equation (1.1), one can instead generate a very large

number of random variables associated with f, and use Equation (1.2) to evolve their positions in time. Then, at any point where one wishes to recover the approximate time evolution of the pdf itself, one can simply take a histogram of the particle positions.

This approach provides the basis particle tracking methods are built upon. It can be applied to contaminant dispersal problems by simply normalizing the concentration distribution so that it integrates to unity, allowing for the distribution to alternatively be considered as representing a pdf.

The underlying method described thus far is only applicable when the particles are moving in a homogenous medium, but can be generalized to smoothly spatially variable velocities and diffusion coefficients [73]. However, these methods for updating a given particle's position are no longer valid when the particle encounters a sharp interface between two distinct media that represents a discontinuity in diffusivity and/or velocity.

Such interfaces are ubiquitous in the case of transport through fractured rock, wherein there is significant advection within fractures and stagnant water in the surrounding rock matrix. Furthermore, the free molecular diffusivity within the fractures is typically two to three orders of magnitude higher than the diffusivity in the rock matrix. Thus there is a discontinuity in both velocity and diffusivity across each fracture-matrix interface.

To handle these interface conditions, the method is augmented to update particle positions with transition rules to determine where to place a particle when it encounters an interface. Let C_f be the concentration within the fracture, C_m be the concentration within the rock matrix, D be the free molecular diffusion coefficient, D_e be the effective diffusivity of the rock matrix, and ϕ be the porosity of the rock matrix. With this terminology in place, the transition probabilities are given by,

$$P(I \to M) = \frac{\phi\sqrt{D_e}}{\sqrt{D} + \phi\sqrt{D_e}}, \tag{1.3a}$$

$$P(I \to F) = \frac{\sqrt{D}}{\sqrt{D} + \phi\sqrt{D_e}}. \tag{1.3b}$$

Here, $P(I \to M)$ and $P(I \to F)$ indicate the probabilities for the particle to transition from the interface to the matrix and fracture respectively. Although these probabilities are independent of time, they are only valid for particles that encounter only a single interface during a time step. Otherwise, the rules are no longer valid.

We now highlight why this time step restriction is problematic for contaminants traveling through fracture networks in rock. Let a represent the half-width of the fracture. The characteristic time scale for a particle to traverse the width of a fracture from diffusional effects is $O(a^2/D)$. Therefore, the solutions using the transition probabilities given by Equation 1.3 are only valid when the time step sizes are substantially less than a^2/D in order to ensure that particles can only encounter a single interface. For a typical fracture aperture of $\approx$0.1mm, the diffusion time scale is on the order of $\approx$10s. Thus, the time-step restriction becomes cripplingly restrictive, in the context of modelling long-term contaminant transport ($\approx$100 years) in fractured rock.

Though the small time step sizes of the SFPM cause it to be computationally expensive, they also allow it to be highly flexible and remain accurate even when the contaminant is dispersing through a domain with a complicated geometry (i.e. where there is an intersection of several fractures). However, when the geometry of the domain local to a particle does not change from one time step to the next, maintaining this degree of flexibility and accuracy may not be necessary. If the fine-grained details of the solution are superfluous to the intended application of the solution, then there is no need to capture them. This insight provides the motivation for what we call the "coarse-scale" versions of the SFPM that we develop in Chapter 2, and our alternative model for the effective viscosity of glacial ice that we develop in Chapter 3.

1.2 Glacial Ice Flow

Along with the increased interest in climatic variation due to global warming, there has been an accompanying interest in understanding how glacial ice masses evolve in time. Of particular interest is how these ice masses contribute to sea level rise, and the rate at which this occurs.

Although meltwater runoff is one mechanism by which glaciers or ice sheets can become seawater, the more important mechanism to capture is that, at very large scales, ice behaves like a non-Newtonian fluid. As direct experimentation is not physically feasible, researchers analyzing ice flow problems rely primarily on numerical simulations.

The constitutive law capturing the non-Newtonian behavior of ice is an important part of ice flow models, as this is what provides the effective viscosity for flowing ice. To determine a model for this effective viscosity, researchers collect datasets by applying a fixed stress to an ice sample. This allows one to infer the effective stress applied and measure the resulting effective strain rate, which then allows one to obtain a data point of the form $(\dot{\epsilon}_e, \sigma_e)$. An example of such a dataset is provided in Figure 1.1.

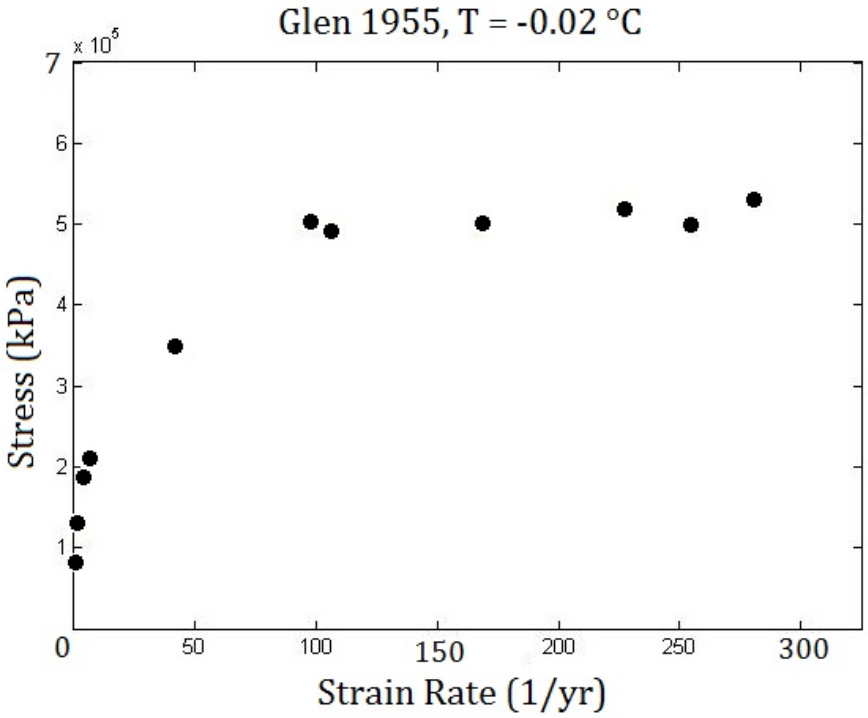


Figure 1.1: An example dataset used to determine the effective viscosity of ice, coming from Glen's original paper for the current flow law model.

The most commonly used form of the constitutive law is Nye's generalization to Glen's flow law [56], which is commonly still called Glen's law for short. It is an empirically based power law

model that gained widespread appeal due to its simple form and ability to fit empirical data with acceptable accuracy.

The stress-strain relationship that leads to Glen's flow law is of the form given by (3.1a) or equivalently (3.1b),

$$\sigma_e = B(T)\dot{\epsilon_e}^{1/n}, \tag{1.4a}$$

$$\dot{\epsilon}_e = B(T)^{-n}\sigma_e^n. \tag{1.4b}$$

It is important to understand that Glen's flow law is based on fitting these data points and the justification for using it is that the law can capture the empirical relation between σ_e and $\dot{\epsilon}_e$ with reasonable accuracy. While simple and acceptably accurate, the drawback to formulating the constitutive relationship with a power law is that it leads to the following form for the flow law relating the stress and strain rate tensors:

$$\sigma_{ij} = \eta(\dot{\epsilon}_e, T)\ \dot{\epsilon}_{ij}, \tag{1.5a}$$

$$\eta(\dot{\epsilon}_e, T) = B(T)\dot{\epsilon_e}^{-(n-1)/n}. \tag{1.5b}$$

where η is the effective viscosity for ice. This form for the effective viscosity is a function which grows unboundedly as the effective strain rate, $\dot{\epsilon}_e$, goes to zero. This is problematic for two reasons. Firstly, it has been demonstrated that in low stress/strain rate regimes the value of the exponent, n, appearing in Glen's flow law should be approximately one [60, 69, 75]. This means the viscosity should go to a finite value in that limit and this isn't reflected in Glen's constitutive relationship. Secondly, although Equation (3.1a) makes analytical calculations for problems like borehole closure rates reasonably straight forward, it is commonly understood to be a great hindrance to numerical calculations [45]. Taking those issues into consideration along with the fact that the Glen's law is empirically based to begin with, we develop an alternative empirical fit for the constitutive relationship and demonstrate its utility in Chapter 3. Our model focuses on accurately capturing

the effective viscosity for glacial ice at high strain-rates (where the glacier is flowing fastest) and using a functional form that omits the artificial singularity of Glen's law at low strain-rates (where the glacier is barely moving).

1.3 Community Detection

Departing from traditional diffusive processes related to the flow of physical fluids considered in the contaminant dispersal and ice flow problems, we also examine community detection through the lens of diffusion. For real world systems represented by graphs, such as social or information networks, a common feature of interest comes from finding groups of vertices that share more or stronger connections with each other than they do with the rest of the vertices in the graph. Community detection is the problem of finding such groups, and is particularly significant to disciplines such as sociology or biology where systems under study are commonly represented as graphs.

The target application for community detection depends on the network under study, and the notion of what makes one algorithm's grouping "better" than another depends on the concept used to define a community for a given network. For protein-protein interaction networks, the goal of community detection may be to find groups of protein interactions involved in the same biological function [17]. If it is a network comprised of items that are purchased together, the goal of community detection may be to make purchase recommendations to customers. When a network comprised of words people associate with a target word, the goal of community detection may be to find the different meanings of that target word [59]. Or in the case of a network comprised of social interactions, the goal of community detection may be to find communities in the literal sense. It is a problem that arises in a myriad of applications.

Although community detection is commonly referred to as though it is a single problem, our view is that it is a body of related problems. Our reason for this lies in the fact that the definition of "community" means different things in different contexts; the organizational structure for a "community" of synonymous words in a WordNet graph has little if not nothing to do with the organizational structure for a "community" of football teams in a conference or the "communities"

of taxonomic distinctions in a food web. Different problems inherently have different notions implied by the word "community", and it is baseless to claim that different qualitative descriptions for communities all have the same underlying mathematical structure.

While it is generally accepted that the basic problem of community detection is to find groups of vertices that are more interconnected with each other than they are to vertices in other groups, there are numerous questions to consider when approaching this problem. What specific properties should a set of nodes possess in order for it to be considered a good community? Should one allow for a vertex to belong to multiple communities, or is it desirable for each to have a unique community assignment? Are the target communities roughly the same size, or does this vary? Should one use a divisive approach where all the vertices belonging to one community and successively divide that community into some final partitioning, or should one take an agglomerative approach with each vertex comprising its own community and successively clustering these communities together? Is a hierarchical community structure expected within the network? There are different ways to answer each of these questions, which in turn leads to different general techniques for approaching the problem.

Our approach to the problem is to detect communities by collections of nodes that share a similar local perspective on a particular community structure in the network, where we represent each perspective by a set of edges. We then form communities by starting with one such perspective as a community base, and allow it to diffuse to encompass similar perspectives. We discuss the details of our approach in Chapter 4.

1.4 Chapter Organization and Original Contributions

The organization of the remainder of this thesis is as follows.

Chapter 2 addresses the contaminant dispersal problem. In this chapter, we make use of locally valid analytical solutions to construct two augmentations to the SFPM that allow for particles to take substantially larger time steps when the assumptions of the analytical solutions are satisfied. Our contribution is developing two complementary coarse-scale particle tracking methods, either

of which can be used as a stand alone augmentation to standard particle tracking methods. This work has been submitted to *Applied Mathematical Modelling* [14].

In Chapter 3, we address glacial ice flow. Our contribution lies in developing an alternative empirically based model for the constitutive relationship. The proposed flow law model is analytically invertible, like Glen's, which allows computational researchers to use standard velocity-pressure formulations common to many computational fluid dynamics (CFD) problems. The proposed law fits empirical data points with accuracy on par with the traditional model, but avoids the viscosity blow up by using a logarithmic functional form for the constitutive relationship as opposed to a simple power function. To demonstrate the computational properties of our flow law, we solve the ice flow problem with a First Order System Least Squares finite element method (FOSLS for short) using both flow laws for comparison. This work has been presented at the 12th Copper Mountain Conference on Iterative Methods [12].

Chapter 4 addresses our approach to the community detection problem. Our approach to the problem begins with modelling the perspectives of community structure on the network with respect to individual nodes. We then form communities on the network by a diffusive process, where we agglomerate nodes into communities based on the nodes possessing similar perspectives of local community structure. Our contribution to this problem lies in developing a computationally efficient algorithm to detect overlapping communities in social networks. Our algorithm is currently one of the very few overlapping community detection algorithms that scales linearly with the number of nodes in the network. This work has been submitted to *Social Network Analysis and Mining* [13].

Chapter 2

Particle Tracking Methods for Contaminant Dispersal

2.1 Introduction

Contaminant transport in fractured rock masses is relevant in the context of protecting groundwater resources and safety assessments of nuclear waste repositories. From a mathematical standpoint, examining this problem involves solving the transport equation for the contaminant's dispersal. Although meshed based numerical approaches to solve Eulerian transport models are often plagued by artificial numerical dispersion or oscillations, the random walk particle tracking (RWPT) method offers an alternative approach which avoids these difficulties. The idea underlying the method is to treat the governing partial differential equation (pde) as describing the time evolution of a probability density function (pdf), thereby making the problem amenable to methods for solving stochastic differential equations (sde's). Specifically, let D represent the diffusion coefficient, v represent the velocity field, and c represent the concentration of contaminant so that Equation (2.1) represents the convection-diffusion equation of interest to a given contaminant dispersal problem.

$$\frac{\partial c}{\partial t} = \nabla \cdot (D \nabla c) - \nabla \cdot (\mathbf{v} c) \tag{2.1}$$

If f is the probability density function evolving in time, μ is the drift vector, and D is the diffusion tensor, the Fokker-Planck equation given by Equation (2.4) below.

$$\frac{\partial f(\mathbf{x},t)}{\partial t} = \nabla \cdot (D \nabla f(\mathbf{x},t)) - \nabla \cdot (\mu f(\mathbf{x},t)) \tag{2.2}$$

Note that save for interpretation of what the variables represent, the convection-diffusion equation is identical to the Fokker-Planck equation. All that needs to be done to map Equation (2.1) to Equation (2.4) is to ensure that the concentration distribution integrates to one, which can be accomplished simply by normalizing c. The significance of this is that a common way to solve the latter is to generate a set of random variables from $f(\mathbf{x}, 0)$ and evolve them in time in order to generate $f(\mathbf{x}, t)$. The time evolution of particle positions is carried out according to the corresponding stochastic ordinary differential equations given by Equation (2.3),

$$dX_t = \mu dt + \sqrt{2D}\ d\mathbf{W}_t \tag{2.3}$$

where $\mathbf{W}_t$ is a standard Wiener process. The appealing characteristics of this method from a computational standpoint is that it grid independent and simple to parallelize. Moreover, particle tracking methods do not suffer from the numerical dispersion effects of grid based methods when dealing with advection dominated problems [1, 73].

Various approaches to particle tracking for transport in fractures have been developed [74, 20, 25, 21], some of which account for fracture-matrix interactions [21, 9, 18]. The most efficient of these approaches typically track particles based on the residence time distributions of particles in individual fractures: The time-domain random walk (TDRW) approach of Bodin [9] assumes planar fractures with uniform velocity within each fracture, and the particle residence time in a fracture-matrix unit is calculated from the analytical solution of Maloszewski and Zuber [48] for a linear fracture. The Lagrangian approach of Cvetkovic et al. [21] accounts for spatial variability in velocity field within fractures, and is based on single fracture-scale retention functions whose form was derived from numerical simulations [18].

2.2 Particle Tracking Notation

D: free molecular diffusion coefficient

D_e: effective diffusivity in rock matrix

ϕ: porosity of rock matrix

u: velocity within fracture

$\bar{u}$: velocity within fracture averaged over fracture width

t: time

y: distance perpendicular to the center line of the fracture

C_f: concentration within fracture

C_m: concentration within matrix

k_A: adsorption constant

ρ_b: bulk density

k_d: distribution coefficient

R: retardation factor

T: non-dimensionalized time

a: fracture half width

b: distance to end of matrix from fracture center line

Δt: time step size

Δt_m: portion of time step spent within matrix

Δt_f: portion of time step spent within fracture

Δt_{FPT}: time step calculated from a first passage time distribution

r_N: random variable $\sim\ N(0,1)$

r_u: random variable $\sim\ unif(0,1)$

2.3 Overview of RWPM / High Resolution Method

Although there are many variations of particle tracking methods for solving contaminant dispersal problems, our work is most closely tied to what we call the standard Fokker-Planck method (SFPM) or "high-resolution method" that we describe in this section. If A and B are taken as constants, and f represents a square integrable probability density function (pdf) evolving in time, the Fokker-Planck Equation can be written as,

$$\frac{\partial f(\mathbf{x},t)}{\partial t} = -\nabla \cdot (A\ f(\mathbf{x},t)) + \nabla \cdot (B\ \nabla f(\mathbf{x},t)). \tag{2.4}$$

Note that aside from the interpretation of what the variables represent, Equation (2.4) is identical to a standard advection-diffusion equation. However, if $\mathbf{X}_t$ represents a random variable associated with f and $\mathbf{W}_t$ a standard Wiener process, then this random variable evolves in time according to the stochastic ordinary differential equation,

$$d\mathbf{X}_t = A\ dt + \sqrt{2B}\ d\mathbf{W}_t. \tag{2.5}$$

This offers an alternative method for calculating the time evolution of the probability density function. Rather than deal directly with Equation (2.4), one can instead generate a very large number of random variables associated with f, and use Equation (2.5) to evolve their positions in time. Then, at any point where one wishes to recover the approximate time evolution of the pdf itself, one can simply take a histogram of the particle positions.

This approach provides the basis particle tracking methods are built upon. It can be applied to contaminant dispersal problems by simply normalizing the concentration distribution so that it integrates to unity, allowing for the distribution to alternatively be considered as representing a pdf.

The underlying method described thus far is only applicable when the particles are moving in a homogenous medium, but can be generalized to smoothly spatially variable velocities and diffusion coefficients [73]. However, these methods for updating a given particle's position are

no longer valid when the particle encounters a sharp interface between two distinct media that represents a discontinuity in diffusivity and/or velocity.

Such interfaces are ubiquitous in the case of transport through fractured rock, wherein there is significant advection within fractures and stagnant water in the surrounding rock matrix. Furthermore, the free molecular diffusivity within the fractures is typically two to three orders of magnitude higher than the diffusivity in the rock matrix. Thus there is a discontinuity in both velocity and diffusivity across each fracture-matrix interface.

To handle these interface conditions, the SFPM is augmented to update particle positions with transition rules to determine where to place a particle when it encounters an interface. The transition rules were originally derived by using the method of images [41], but equivalent rules can be obtained by considering the time evolution of a Dirac delta concentration distribution located at the interface between the two media. Let C_f be the concentration within the fracture, C_m be the concentration within the rock matrix, D be the free molecular diffusion coefficient, D_e be the effective diffusivity of the rock matrix, ϕ be the porosity of the rock matrix, δ be a Dirac delta function, and s be a spatial coordinate perpendicular to the fracture wall such that positive s lies within the surrounding rock matrix. With this terminology in place, the time evolution of the concentration distribution is given by the following system of equations,

$$\partial C_f/\partial t = D\ \partial^2 C_f/\partial s^2, \qquad s < 0, \tag{2.6a}$$

$$\partial C_m/\partial t = D_e\ \partial^2 C_m/\partial s^2, \qquad s > 0, \tag{2.6b}$$

$$C_f = C_m, \qquad s = 0, \tag{2.6c}$$

$$D\ \partial C_f/\partial s = \phi\ D_e\ \partial C_m/\partial s, \qquad s = 0, \tag{2.6d}$$

$$C_f + \phi\ C_m = \delta(s), \qquad t = 0. \tag{2.6e}$$

The probability for the particle to transition to either the matrix or the fracture is taken to be equal to the fraction of the concentration distribution falling within that medium. These

probabilities are given by,

$$P(I \to M) = \frac{\phi\sqrt{D_e}}{\sqrt{D} + \phi\sqrt{D_e}}, \tag{2.7a}$$

$$P(I \to F) = \frac{\sqrt{D}}{\sqrt{D} + \phi\sqrt{D_e}}. \tag{2.7b}$$

Here, $P(I \to M)$ and $P(I \to F)$ indicate the probabilities for the particle to transition from the interface to the matrix and fracture respectively. Note that although these probabilities are independent of time, the implicit semi-infinite domain requires that particles can only encounter a single interface during a time step for the probabilities to be valid.

If a represents the half-width of the fracture, then characteristic time scale for a particle to traverse the width of a fracture from diffusional effects is $O(a^2/D)$. Therefore, the SFPM solutions are only valid when the time step sizes are substantially less than a^2/D in order to ensure that particles can only encounter a single interface. For a typical fracture aperture of ≈0.1mm [37], the diffusion time scale is on the order of ≈10s. Thus, the time-step restriction becomes cripplingly restrictive, in the context of modeling long-term contaminant transport (≈100 years) in fractured rock.

The small time step sizes of the SFPM cause it to be computationally expensive, but they also allow it to be highly flexible; the SFPM will remain accurate even when the contaminant is dispersing through a domain with a complicated geometry (i.e. where there is an intersection of several fractures or tortuous fractures). However, when the characteristics of the domain local to a given particle does not change significantly from one time step to the next, maintaining this degree of flexibility may not be necessary. This insight provides the motivation for what we call the "coarse-scale" versions of the SFPM that we develop in the following sections.

2.4 Proposed Approach: Coarse-scale Particle Tracking Methods

Our work focuses on developing two complementary methods that allow for particles to take substantially larger time steps. One method handles particles starting within the fracture, and the other handles particles starting within the rock matrix surrounding an isolated fracture. Although we present results for when these two methods are used in conjunction with each other, either can be replaced by the SFPM (or other methods) in sub-domains (e.g. vicinity of fracture intersections) where higher resolution is needed. These methods address two-way transfer of particles across fracture-matrix interfaces, and may be combined with the SFPM for handling transport within the fracture plane (advection + multi-dimensional diffusion/dispersion), and diffusion in the matrix in directions parallel to the fracture plane (although this is a very slow transport mechanism, and becomes important only over very long time scales [53]). The advantage here is that because the proposed methods can take time steps that are substantially greater than the diffusive timescale (a^2/D), the overall time-step is controlled by the time-step required to resolve advective transport within complex velocity fields within the fracture plane, and does not need to be smaller than (a^2/D). In cases where aperture (and thus aperture-averaged velocity) variations within individual fractures is explicitly represented, the time-step required for resolving these variations accurately is typically based on requiring that the advective displacement during a time-step remains a fraction of the grid-block dimensions in the flow model. When aperture variability in individual fractures is not explicitly represented, the time-step needed for resolving velocity variations is controlled by the advective travel time along a fracture.

For both cases noted above, we can imagine a locally planar fracture, with a local y coordinate aligned with the s coordinate in Equation (2.6) and local x, z coordinates in the fracture plane. We also assume that there are no other fractures nearby, so that the rock matrix can be assumed to be semi-infinite in extent for practical purposes [21] at least in the context of particles starting in the fracture. Since the focus on taking time steps that are substantially greater than the diffusive timescale, a^2/D, a reasonable assumption to make is that the particles position between the fracture

walls can be modeled as being uniformly distributed whenever the particle is within the fracture. A depiction of this scenario is given in Figure 2.1.

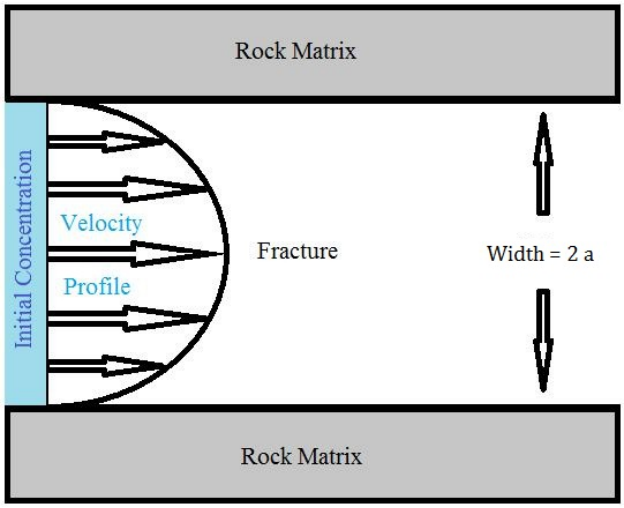


Figure 2.1: Situational set up for planar fracture problem.

2.4.1 Coarse-Scale Method for Particles Starting in Rock Matrix

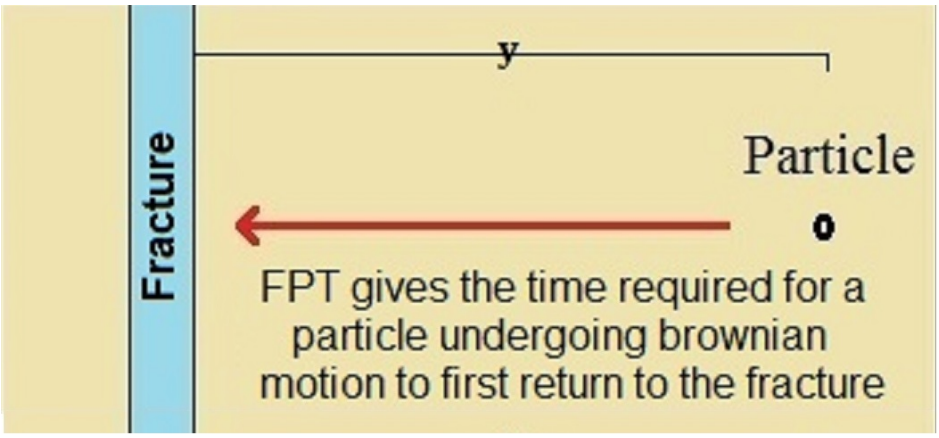


Figure 2.2: When a particle undergoing Brownian motion starts at a distance y from the fracture, the first passage time, Δt_{FPT}, provides a stochastic estimate of how long it takes the particle to first reach the fracture.

If a particle is in the rock matrix near a planar fracture, with no other fractures nearby, then

the quantities of interest are how long it takes for the particle to return to the fracture and where in the fracture the particle returns to.

While in the rock matrix, the particle's diffusion in the x, y and z directions are independent, so calculating the return time only needs to consider the y-diffusion. The problem of determining the time it takes for a particle to return to the fracture is identical to a first passage time (FPT) problem for the one-dimensional diffusion equation [7], which implies that the time taken can be modeled as being Lévy distributed, and can be generated through Equation (2.8) [16],

$$\Delta t_{FPT} = (y/r_N)^2/(2D_e), \tag{2.8}$$

where $r_N \sim N(0,1)$. Once the total time within the matrix has been calculated, the diffusional effects in the x and z direction can easily be added back in as,

$$\Delta x_{FPT} = \sqrt{2D_e\Delta t_{FPT}}\ r_N\ , \tag{2.9a}$$

$$\Delta z_{FPT} = \sqrt{2D_e\Delta t_{FPT}}\ r_N\ . \tag{2.9b}$$

Note that Δt_{FPT} is unbounded, meaning that this approach allows for particles to effectively take time steps with no upper bound at the computational cost of generating a few random variables. The role this technique plays in our algorithm is to provide an accurate estimate for how much time passes for particles that have wandered into the rock matrix before returning to the fracture. However, it is important to note that this technique only requires that the particle's distance to the nearest fracture is much smaller than the distance to other fractures. It can easily be modified to be applicable in the neighborhood of fractures which are not planar or by fracture intersections by creating a straight line that bounds the fracture(s) away from the particle, as in Figure 2.3. The FPT approach can then be used to advance the particle up to the straight line before continuing to track it with the SFPM. The importance of distinction is that this first passage

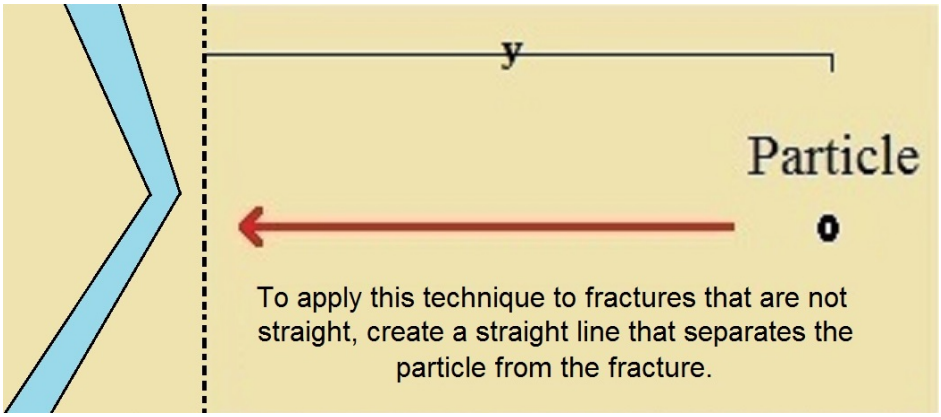


Figure 2.3: When a particle undergoing Brownian motion near a fracture which is not straight, the FPT approach can still be applied to a straight line bounding the fracture away from the particle.

time approach is widely applicable, and can be used as a stand alone method to augment the SFPM in a computationally efficient way.

2.4.2 Coarse-scale Method for Particles Starting in a Planar Fracture

Our coarse-scale method for particles starting within the fracture is based on leveraging the fact that, in this special case, a particle's long term behavior can accurately be determined based on how much time it spends within each medium. While the particle is in the rock matrix, it undergoes Brownian motion. While it is in the fracture, its displacement is primarily determined by the velocity field. Note that during a time step $\Delta t >> a^2/D$, the mean velocity of a particle originating anywhere across the aperture is equivalent to the mean velocity of field [2], so the particle's rate of displacement is given by the mean of the velocity field in this direction. Accurately modeling how much time the particle spends in each medium over a large time step thereby allows one to use these two simpler models to determine the particle's overall displacement.

2.4.2.1 Task 1: Does the Particle Transition to the Surrounding Matrix?

When a particle is in the fracture, our first task is to determine if the particle transitions to the rock matrix during a time step. Determining the probability of transition for a particle whose

initial position is uniformly distributed across the aperture is equivalent to calculating how much of an initially uniform concentration distribution across the fracture width diffuses into the rock matrix over a time step. Note that this is only governed by the diffusional effects perpendicular to the fracture length, and that the domain is symmetric about the center of the fracture. With these simplifications, we can model the initial concentration distribution by an indicator function, $\frac{1}{a}\chi_{(0,a)}(y)$, and the evolution of the particle's initially uniform concentration distribution by the following pde,

$$\partial C_f/\partial t = D\ \partial^2 C_f/\partial y^2, \qquad 0 < |y| < a, \tag{2.10a}$$

$$\phi\ \partial C_m/\partial t = \phi\ D_e\ \partial^2 C_m/\partial y^2, \qquad |y| > a, \tag{2.10b}$$

$$\partial C_f/\partial y = 0, \qquad |y| = 0, \tag{2.10c}$$

$$C_f = C_m, \qquad |y| = a, \tag{2.10d}$$

$$D\ \partial C_f/\partial y = \phi\ D_e\ \partial C_m/\partial y, \qquad |y| = a, \tag{2.10e}$$

$$C_f = \frac{1}{a}\chi_{(0,a)}(y), \qquad t = 0 \tag{2.10f}$$

The solution to this equation at the end of a time step of size Δt is given by [58],

$$C_f(y,\Delta t) = 1 - \frac{1-\gamma}{2}\sum_{n=0}^{\infty}\gamma^n[erfc(\frac{(2n+1)a-y}{2\sqrt{D\ \Delta t}}) + erfc(\frac{(2\ n+1)a+y}{2\sqrt{D\ \Delta t}})], \tag{2.11a}$$

$$C_m(y,\Delta t) = \frac{1+\gamma}{2}\sum_{n=0}^{\infty}\gamma^n[erfc(\frac{2\ n\ a+\mu(y-a)}{2\sqrt{D\ \Delta t}}) + erfc(\frac{2(n+1)a+\mu(y-a)}{2\sqrt{D\ \Delta t}})], \tag{2.11b}$$

where μ and γ are defined as

$$\mu = \sqrt{\frac{D}{D_e}}, \tag{2.12a}$$

$$\beta = \frac{1}{\phi}\sqrt{\frac{D}{D_e}}, \tag{2.12b}$$

$$\gamma = \frac{\beta - 1}{\beta + 1}. \tag{2.12c}$$

Since there is a fixed amount of solute in the system, we know that $\int_0^a C_f(y)\,\mathrm{d}y + \phi \int_a^\infty C_m(y)\,\mathrm{d}y = a = \text{constant}$. From this, we get the transition probability to go from the fracture to the matrix during a time step of size Δt:

$$P(F \to M) = \frac{\phi}{a}\int_a^\infty C_m(\tilde{y}, \Delta t)\,\mathrm{d}\tilde{y}. \tag{2.13}$$

Rather than calculate this integral at every time step (which would be a prohibitively expensive process), we represent $P(F \to M)$ as a function of the parameters D, D_e, ϕ, a, and, Δt. We then calculate $P(F \to M)$ for a set of parameter values in order to construct a set of approximating interpolation polynomials for the function. In building this approximation, the values of Δt are set to be of the form $\Delta t = Ta^2/D$ for a fixed set of values of T, which provide the time step sizes available to the method. The values of the remaining parameters are used as interpolation points to construct approximating polynomials, one for each desired T value, capable of providing transition probabilities for arbitrary values of D, D_e, ϕ, and a. Using the methods outlined in Section 2.5, these polynomials can be evaluated in only $O(10)$ flops.

Once $P(F \to M)$ has been calculated, we generate a uniformly distributed random variable, r_u. If $r_u \leq P(F \to M)$, then the particle does not transition and we simply advance its position along the fracture length by $\overline{u}\ \Delta t$. If $r_u > P(F \to M)$, then there are several additional quantities that must be calculated. Specifically, we need to determine how deep within the rock matrix the particle penetrated, and how much time it took for the particle to reach that displacement. These will be discussed in the following sections.

2.4.2.2 Task 2: How Deep Does the Particle Penetrate?

When a particle transitions from the fracture to the rock matrix, we first determine how deep within the rock matrix the particle has penetrated during the time step, as this will later be used to estimate the amount of time it took to reach that displacement. We treat this displacement as a random variable whose pdf is given by portion of the concentration distribution lying within the rock matrix, normalized so that it integrates to unity. This allows us to determine the displacement by again generating a uniformly distributed random variable, and determining the value for y such that,

$$r_u = \frac{1}{\int_a^\infty C_m(\tilde{y})\,\mathrm{d}\tilde{y}} \int_a^y C_m(\tilde{y})\,\mathrm{d}\tilde{y}. \tag{2.14}$$

To make this calculation tractable, we use the methods explained in 2.5.1 to determine a set of y_j offline such that,

$$\frac{1}{J} = \frac{1}{\int_a^\infty C_m(\tilde{y})\,\mathrm{d}\tilde{y}} \int_{y_j}^{y_{j+1}} C_m(\tilde{y})\,\mathrm{d}\tilde{y}. \tag{2.15}$$

where $J > 1$ is an integer and $j \in \{0, 1, ..., J\}$. The $y_j's$ then partition the matrix domain into sections with equal probability of $1/J$ for the particle to be in it at the end of a time step. While running a particle tracking simulation, this reduces the problem of calculating the y value such that Equation (2.14) holds to performing a binary search for the y_j values bracketing r_u and calculating the root of a simple polynomial interpolation representing $\int_a^y C_m(\tilde{y})\,\mathrm{d}\tilde{y}$ over that region. For the results presented in this paper, we used $J = 20$, with linear approximations to C_m for all but one of the sub-intervals. The only sub-interval where we did not use a linear approximation for C_m is the final one containing the long tail of the pdf that extends out to infinity. For this case, we use a finite value for y_{J+1} where $C_m(y_{J+1}) \approx 0$, and artificially fix $C_m(y_{J+1}) = 0$ at that point; a simple heuristic that we have found works well is to set $y_{J+1} = 2\ y_J$. With these two points in place, we then use a monotonic quartic polynomial for interpolation over this interval.

All of these choices for the approximation process were primarily made for the sake of ef-

ficiency, and provide sufficient accuracy as will be demonstrated in Section 2.6. However, the accuracy of the approach can be further improved if needed by increasing J or using higher order approximations to $C_m(y)$ for $y \in [y_j, y_{j+1})$ to determine the appropriate y value.

2.4.2.3 Task 3: How Much Time is Spent Within Each Medium

Once we have determined the particle's displacement into the rock matrix, we use this information to determine how much time the particle spent within the fracture, Δt_f, and matrix, Δt_m. Because the time step for the particle, Δt, is fixed at the beginning and $\Delta t = \Delta t_f + \Delta t_m$, we need only determine one of these times to be able to obtain the value of the other.

The motivation for this separation is because the particle's motion is very simple to model in each medium. While it is in the fracture, its displacement is primarily determined by the advection it experiences. If one knows Δt_f, then a simple but accurate estimate of the particle's displacement along the fracture is $\bar{u}\ \Delta t_f + \sqrt{2D\Delta t_f}\ r_N$. If one knows Δt_m, then the particle's displacement along the fracture due to Brownian motion within the matrix can be taken as $\sqrt{2D_e\Delta t_m}\ r_N$. As the displacement perpendicular to the fracture has already been determined from the previous section, the displacements in the x and z directions are all the remaining information needed to determine the particle's overall displacement during a time step.

To begin this process, we note that a somewhat crude approximation to Δt_m would be to treat the time it took the particle to reach a distance y into the matrix as a first passage time problem, as can be inferred from Figure 2.2. In this case, an initial estimate for Δt_m is $\Delta t_m \approx \Delta t_{FPT}$, which can be generated through Equation (2.8).

However, the time taken to reach this y value is not accurately reflected by a first passage time problem of the type depicted in Figure 2.2. One additional consideration is that a first passage time is potentially unbounded, whereas the time taken to reach the displacement cannot exceed the Δt value set at the beginning. This limitation can be enforced by repeatedly generating Δt_{FPT} until the constraint is satisfied or a threshold number of generations has been reached. In this latter case, simply set $\Delta t_{FPT} = \Delta t$. Another additional consideration is that the particle can pass its ending

position and double back during a time step, meaning it literally is not a first passage problem. Because particles generally pass their ending displacement multiple times during a time step, this implies using the constrained Δt_{FPT} by itself to estimate Δt_m will consistently underestimate the time spent within the matrix. To adjust for this, we first set

$$\Delta t_f = (\Delta t - \Delta t_{FPT})\ r_u, \tag{2.16}$$

where the extra $r_u \sim unif(0,1)$ is to account for the additional time spent in doubling back through the matrix, and it is understood that the constraint $\Delta t_{FPT} \leq \Delta t$ is enforced. This is an estimate that numerical experimentation has shown can break down for time steps significantly greater than $\Delta t = 500\ a^2/D$, but is reliable otherwise. Although this further restricts the upper bound for the time step sizes one can take using our algorithm, it is worth mentioning that such time steps are more than 3 orders of magnitude larger than those required by the SFPM methods, which are on the order of $\Delta t = 0.1\ a^2/D$.

Once we have calculated Δt_f using the method above, we determine Δt_m by

$$\Delta t_m = \Delta t - \Delta t_f\ . \tag{2.17}$$

The final step of the process for determining the particle's position after a time step is to set the particle's displacement along the x and z directions to

$$\Delta x = \overline{u}\Delta t_f + \sqrt{2D\Delta t_f}\ r_N + \sqrt{2D_e\Delta t_m}\ r_N\ , \tag{2.18a}$$

$$\Delta z = \sqrt{2D_e\Delta t_m}\ r_N. \tag{2.18b}$$

The general form of Equation (2.18) shows how our approach for handling fracture-matrix transfer can be supplemented with the standard SFPM for transport within fracture planes. In Equation (2.18a), the advection term is the dominant term, and the diffusion-dispersion terms

within the fracture may be neglected in the interest of computational efficiency, as also in some of the TDRW methods discussed above [21, 9]. However, over long time scales, the diffusion-dispersion terms are also important and enable the incorporation of the mechanisms noted by [53].

2.5 Cost Efficiency for In-fracture Coarse-scale Method

A significant hurdle in making our coarse-scale method for particles within fractures more cost effective than just using the SFPM comes from making the equations governing transition probabilities and particle placement within the rock matrix cost effective to evaluate. These are given by Equation (2.13) and Equation (2.14) respectively. In the following subsections, we explain the subtleties of constructing cost effective interpolation polynomials to determine the quantities of interest in both of those equations.

2.5.1 Cost Efficiency for Calculating Transition Probabilities

The integral required to determine the probability for a particle to transfer from the fracture to the matrix, given by Equation (2.13), has the following analytical representation [58]

$$\begin{aligned}\int_a^\infty C_m(\tilde{y},\Delta t)\,\mathrm{d}\tilde{y} = \frac{\sqrt{D_e}(1+\gamma)a}{\sqrt{D}}[\sqrt{\frac{T}{\pi}}\sum_{k=0}^{\infty}(\gamma^k(e^{-\frac{k^2}{T}}-e^{-\frac{(k+1)^2}{T}})+\sum_{k=0}^{\infty}\gamma^k \\ +\sum_{k=0}^{\infty}\gamma^k(k\ \mathrm{erf}(\frac{k}{\sqrt{T}})-(k+1)\ \mathrm{erf}(\frac{k+1}{\sqrt{T}})]\end{aligned} \tag{2.19}$$

where $\Delta t = Ta^2/D$ and $\gamma = \frac{\frac{1}{\phi}\sqrt{\frac{D}{D_e}}-1}{\frac{1}{\phi}\sqrt{\frac{D}{D_e}}+1}$. To make evaluation of this function computationally tractable, we approximate it by a Lagrange interpolation polynomial written in a cost effective form.

The full space of parameters we would need to interpolate over is D_e, D, ϕ, a, and T. However, if we try to interpolate over all of those dimensions then even the interpolation polynomial will be too computationally expensive to be of much use. Fortunately, this function can be written in the form,

$$\frac{\sqrt{D_e}(1+\gamma)a}{\sqrt{D}}\ f(\gamma,T), \tag{2.20}$$

with $f(\gamma,T)$ representing the three infinite sums. Because the function can be split up in this way, we need only concern ourselves with creating an interpolation polynomial for $f(\gamma,T)$ because we can then simply multiply that polynomial by $\frac{\sqrt{D_e}(1+\gamma)a}{\sqrt{D}}$ to approximate the full function. Because $f(\gamma,T)$ depends heavily on T, it is impractical to interpolate over this dimension as it would require many interpolation points and drive up the cost of evaluating the function. This can be easily sidestepped by simply having T take on values from a fixed set (e.g. $T \in \{1, 10, 100\}$) and restrict ourselves to only taking time steps such that $\Delta t = Ta^2/D$. This leaves us with only needing to interpolate over the γ dimension. As $0.5 \leq \gamma < 1$ for practical values of the parameters involved and the function $f(\gamma,T)$ varies smoothly with respect to γ for any given T value, interpolation over this dimension can be done accurately with only a handful of interpolation points. Once one constructs Lagrange interpolation polynomials over the γ dimension for each T value, the final preprocessing step is to collect the coefficients for each power of γ so that one can use Horner's scheme to inexpensively evaluate the polynomial in order to determine $P(F \to M)$ for any values of $D_e,\ D,\ \phi,$ and a while running a particle tracking simulation. For the results presented in this paper, we only used 10 interpolation points, evenly spaced between 0.5 and 0.99999.

2.5.2 Cost Efficiency for Calculating Particle Placement

The problem of determining where to place a particle within the rock matrix at the end of a time step is slightly more involved than determining transition probabilities. It entails calculating the integral given by the right hand side of Equation (2.14), which has the form

$$\begin{aligned}&\mathrm{cdf}(y,D_e,D,\phi,a,T) = \\ &\quad 1 - \frac{\sqrt{D_e}(1+\gamma)}{2\ P(F\to M)}\frac{a}{\sqrt{D}}\sum_{k=0}^{\infty}\gamma^k[\tau(\frac{y}{a},T,\frac{D}{D_e},k) - \tau(\frac{y}{a},T,\frac{D}{D_e},k+1)]\end{aligned} \tag{2.21}$$

$$\tau(\frac{y}{a}, T, \frac{D}{D_e}, k) = (\sqrt{\frac{D}{D_e}}(1-\frac{y}{a}) - 2k) + \sqrt{\frac{4T}{\pi}}\exp[\frac{(\sqrt{\frac{D}{D_e}}(1-\frac{y}{a}) - 2k)^2}{4T}] - (\sqrt{\frac{D}{D_e}}(1-\frac{y}{a}) - 2k)\text{erf}[\frac{\text{abs}[(\sqrt{\frac{D}{D_e}}(1-\frac{y}{a}) - 2k)]}{2\sqrt{T}}] \tag{2.22}$$

This function can be approximated more easily by writing it in the form,

$$\text{cdf} = 1 - \frac{\sqrt{D_e}(1+\gamma)}{2\ P(F \to M)}\frac{a}{\sqrt{D}}\ F(\gamma, \frac{y}{a}, T, \sqrt{\frac{D}{D_e}}), \tag{2.23}$$

as we need only create an interpolation polynomial for F as opposed to the entire function.

For this problem, F can be written to only involve the variables γ and $\zeta = \sqrt{\frac{D}{D_e}}(1-\frac{y}{a})$. However, the purpose of the cdf is for finding the $y_j's$ which satisfy Equation (2.15), and we do not need to be concerned with creating a cost effective approximation to the cdf itself. In light of this, we instead create interpolation polynomials that can be used to construct the $y_j's$ by the following steps.

Step 1) Set the value of J to be used in Equation (2.15). For the sake of discussion, let $p_j = \frac{j}{J}$ for $j = 0, 1, ..., J-1$.

Step 2) Set the interpolation points to be used in the γ dimension. We refer to a point in this set as γ_i. The dependence of F on this dimension is not as great as the f if the transition probability and the results presented in this paper used only six equally spaced interpolation points for this process.

Step 3) For each of the p_j's and for each of the γ_i points, use a root finding method to calculate the ζ_{ij} value such that $cdf(T, \gamma_i, \zeta_{ij}) = p_j$.

Step 4) Treat ζ_{ij} as a function of γ, and use the points found in step 3 to create a Lagrange interpolation polynomial, $\zeta_j(\gamma)$ such that $\zeta_j(\gamma_i) = \zeta_{ij}$.

Step 5) Once the Lagrange interpolation polynomial has been constructed, calculate the coefficients necessary to implement Horner's scheme to evaluate it.

Once these steps have been completed, one can obtain the necessary y_j values from the $\zeta_j(\gamma)$ function by using the formula $\frac{y_j}{a} = \frac{\sqrt{\frac{D}{D_e}} - \zeta_j(\gamma)}{\sqrt{\frac{D}{D_e}}}$. This allows one to inexpensively obtain the y_j values satisfying Equation (2.15). These can then be used to construct a piecewise polynomial interpolation for $\int_a^y C_m(\tilde{y})\,\mathrm{d}\tilde{y}$, using a linear polynomial for interpolation purposes on all pieces save for the last segment that comes from satisfying Equation (2.24).

$$\frac{1}{J} = \frac{1}{\int_a^\infty C_m(\tilde{y})\,\mathrm{d}\tilde{y}} \int_{y_{J-1}}^\infty C_m(\tilde{y})\,\mathrm{d}\tilde{y} \tag{2.24}$$

For this special case, instead of using $y_J = \infty$, we set $y_J = 2\ y_{J-1}$, and use a quartic polynomial that is monotonically increasing over this region to interpolate the cdf between the points y_{J-1} and y_J. Making this switch in choice of interpolating polynomials helps to account for the long tail of the cdf in this region.

2.5.3 Summary of Algorithm

Assembling all of the steps explained in the preceding sections, an outline of the method as applied to a single fracture is as follows.

Offline Calculations: Construct and store efficient polynomial approximations to the right hand sides of Equation (2.13) and Equation (2.14), as explained in Section 2.5.

Input: the length of the fracture, L, the half-aperture, a, the mean velocity along the fracture length, $\overline{u}$, the particle's starting time, t_0, and the particle's position along the fracture x.

Preprocessing: Set $\Delta t = T\ \frac{a^2}{D}$, where T is taken as the largest T value used to construct the approximating polynomials that satisfies:

- $x + T\ \frac{a^2}{D}\overline{u} < L$
- $T\ \frac{a^2}{D} < \frac{L}{N_{step}\ \overline{u}}$

The latter condition is necessary as our algorithm only recovers true particle behavior on average over a series of time steps. Setting $N_{step} >> 1$ provides a means to ensure multiple applications for a typical particle. Even with this restriction on available time step sizes, the coarse-scale algorithm can reduce computational costs by two to three orders of magnitude over the SFPM.

Step 1) If a particle is in the rock matrix surrounding the fracture, use the method described in Section 2.4.1 to determine how much time, Δt_{FPT}, passes before the particle enters the fracture. Advance the particle's time by this amount. Adjust its position along the fracture by $\Delta x_{FPT} = \sqrt{2D_e\Delta t_{FPT}}\ r_N$, and its position in the vertical direction by $\Delta z_{FPT} = \sqrt{2D_e\Delta t_{FPT}}\ r_N$.

Step 2) For a particle in the fracture, generate a uniform random variable, r_u. Then use Equation (2.13) to determine if the particle transitions from the fracture, and advance its time to $t_0 + \Delta t$.

Step 3) If a particle does not transition, advance its position along the fracture by $\Delta x = \overline{u}\Delta t + \sqrt{2D\Delta t}\ r_N$.

Step 4.1) If a particle does transition into the matrix, determine how far into the matrix the particle has gone using Equation (2.14).

Step 4.2) Using Equation (2.16) subject to the constraint $\Delta t_f \leq \Delta t$, determine how much time the particle spent within the fracture, Δt_f, and the rock matrix Δt_m. Advance the particle's position along the fracture by $\Delta x = \overline{u}\Delta t_f + \sqrt{2D\Delta t_f}\ r_N + \sqrt{2D_e\Delta t_m}\ r_N$, and its position in the vertical direction by $\Delta z = \sqrt{2D_e\Delta t_m}\ r_N$

Repeat from Step 1 until no T value is small enough to satisfy the constraints for Δt. If no such T value exists, revert to using the SFPM.

2.6 Verification of Coarse-scale Methods

We use two sets of model problems to verify our coarse-scale methods. The first set compares the in-fracture concentration distributions generated by the coarse-scale methods against those generated by the SFPM for a fixed set of parameter values. The second set examines the breakthrough curves produced using the coarse-scale methods, and compares them against the analytical solution for transport parameters corresponding to the Oak Ridge [39] and Shoal [66] sites as given in Table

2.2.

2.6.1 Experiment Set 1: In Fracture Concentration Distribution Recovery

For practical purposes, the most important portion of the contaminant concentration distribution is that which lies within the fractures, as this is the most mobile portion. In the first series of experiments, we compared the in-fracture concentration distribution results given by the coarse-scale methods against those of the SFPM for a long straight fracture with $\bar{u} = 1\ m/d$, $a = 0.5\ mm$, $D = 10^{-9}(m^2/s)$, $D_e = 10^{-10}(m^2/s)$, and $\phi = 0.1$. Each test consists of determining the in-fracture distribution at specified ending times of $t_{end} = T_{end} * a^2/D$ where $T_{end} \in \{1000, 10000, 34560\}$ when starting with a uniform concentration at the upstream end of the fracture at time $t = 0$. For this series of experiments, we set the value of N_{step} to 50.

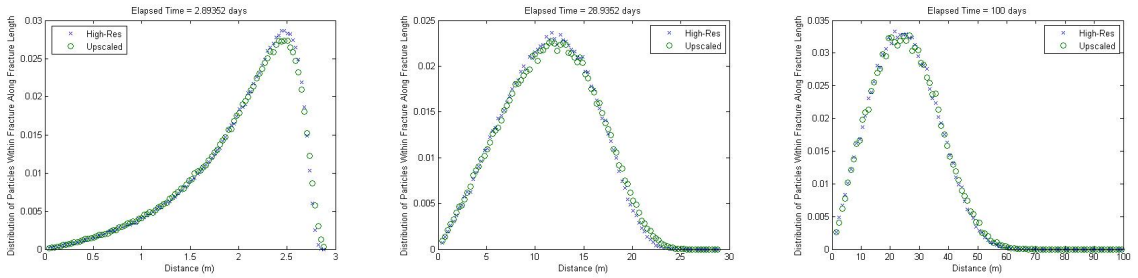
These experiments verify that we are recovering the same concentration distribution regardless of which method we use to obtain the results, and the solutions are plotted in Figure 2.4. The computational costs of the methods are presented in Table 2.1 as well as the relative errors in the L^1 norm of using the coarse-scale methods compared against the SFPM. The concentration distributions have been normalized to integrate to unity so that these L^1 errors reflect the overall absolute difference between the results as a percentage of the total area under the curve. As we can see, the coarse-scale methods essentially recover the same concentration distribution as the SFPM with a computational cost that is two to three orders of magnitude lower.

2.6.2 Experiment Set 2: Breakthrough Curve Recovery

A common feature of interest to contaminant dispersal problems are breakthrough curves, which measure the contaminant flux that passes a fixed downstream point along a fracture as a function of time. To this end, our second series of experiments focuses on applying our coarse-scale methods to accurately recover breakthrough curves obtained by the analytical solutions for breakthrough curves in single fractures derived by Maloszewski and Zuber [48], using parameter values for two real world sites presented in Table 2.2 [39, 66]. Whereas the first series of experiments

Test	FLOPS: High-Res	FLOPS: Upscaled	RV: High-Res	RV: Upscaled	L^1 Error
No-Adsorption $T = 1000$	240,000	2580	20,000	135	0.0342
No-Adsorption $T = 10,000$	2,400,000	1810	200,000	120	0.0211
No-Adsorption $T = 34560$	8,294,400	3510	691,200	215	0.0365

Table 2.1: The computational costs and errors for each of the tests. The RV columns in this table refer to the number of random variables (either uniformly or normally distributed) that need to be generated in running the algorithm. As the coarse-scale methods only recovers the high-resolution on average, we limited the time step size for these tests so that the coarse-scale method had to take at least 50 time steps in order to demonstrate that one can recover the in fracture concentration distribution.



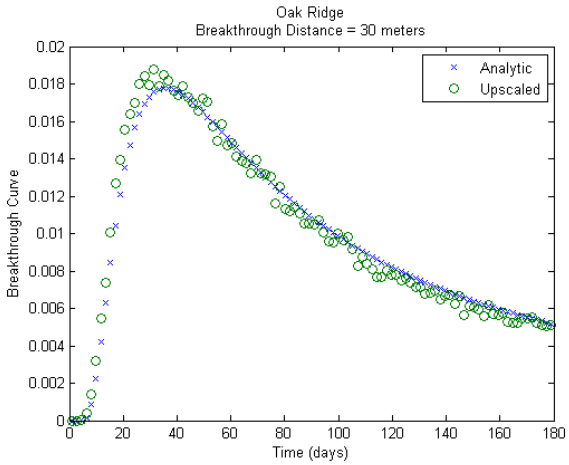
(a) T = 1000 (b) T = 10000 (c) T = 34560

Figure 2.4: Comparing in fracture concentration distribution results of the high-resolution and coarse-scale methods for the non-adsorbing case. For these tests, time steps sizes for the coarse-scale method were restricted so that at least 50 applications of the coarse-scale method were used for each particle. The motivation for this restriction is to demonstrate that the coarse-scale method is capable of yielding the same solution as the high-resolution method through a sufficient number of applications.

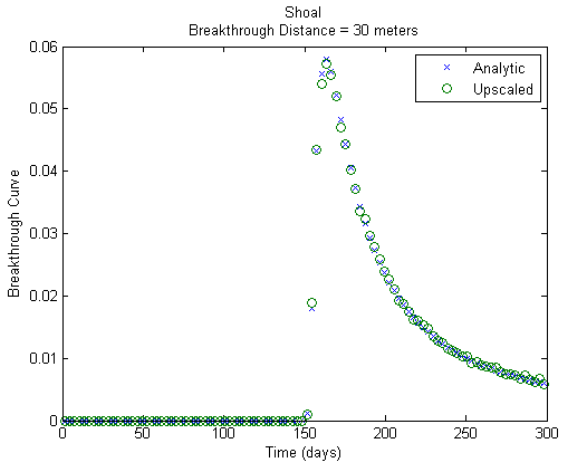
Test	$\bar{u}$ (m/day)	a (mm)	ϕ	D (m^2/s)	D_e (m^2/s)	Duration (days)	Distance (m)
Oak Ridge	100	0.04	0.20	$2 * 10^{-9}$	$1.2 * 10^{-9}$	180	30
Shoal	0.2	0.5	0.01	$2 * 10^{-9}$	10^{-10}	300	30

Table 2.2: Real world test site parameter values.

focus on generating the final concentration distribution at a particular ending time, breakthrough curves concern the time evolution of the concentration distribution at a fixed location. Because of this, we increased the value of N_{step} to 1000 to allow for higher resolution in the time domain. Even with this large value for N_{step}, the maximum coarse-scale time step sizes are $\approx 30\frac{a^2}{D}$ for the Oak Ridge test and $\approx 100\frac{a^2}{D}$ for the Shoal test. The breakthrough curves coming from the analytical solution as well as the coarse-scale particle tracking method are presented in Figure 2.5, where the error using the Oak Ridge parameter values is 2.75% and the error using the Shoal parameter values is 0.15%.



(a) Oak Ridge: 2.75% Error



(b) Shoal: 0.15% Error

Figure 2.5: Comparing breakthrough curves generated by an analytical solution to those generated by the coarse-scale methods for parameter values corresponding to the Oak Ridge and Shoal sites. The data points corresponding to the analytical solution are denoted by x's and those of the coarse-scale methods are denoted by o's.

2.7 Extending These Methods to Cases with Adsorption

Another case of interest to contaminant dispersal is where contaminants can adsorp onto the fracture walls with adsorption constant k_A. In this section, we attempt to extend the results of the previous sections to handle this case when particles begin a time step within such a fracture.

The coarse-scale method can be handled in a manner nearly identical to what was done before. The only difference is that there is no analytical solution to work with, so one instead needs to generate a numerical solution to the one dimensional diffusion problem perpendicular to the fracture length as a preprocessing step. A drawback of this is that the methods outlined in Section 2.5 no longer apply, and one must generate a numerical solution for each set of constants that appear in the fracture network.

The interface transition probability rules for the high-resolution method are derived in a similar manner as for the case without adsorption. We start off again by modelling the situation of a particle hitting the fracture/matrix interface by the following pde,

$$\partial C_f(y,t)/\partial t = D\ \partial^2 C_f(y,t)/\partial y^2, \qquad |y| < 0, \tag{2.25a}$$

$$\phi\ \partial C_m(y,t)/\partial t = \phi\ D_m\ \partial^2 C_m(y,t)/\partial y^2, \qquad |y| > 0, \tag{2.25b}$$

$$C_f = C_m, \qquad |y| = 0, \tag{2.25c}$$

$$-D\ \partial C_f(y,t)/\partial y = k_A \partial C_f(y,t)/\partial t - \phi\ D_e\ \partial C_m(y,t)/\partial y, \qquad |y| = 0, \tag{2.25d}$$

$$C(t=0) = \delta(y+y_0), \tag{2.25e}$$

and taking the limit as $y_0 \to 0$. Here, $D_m = \frac{\phi\ D_e}{\phi + k_A}$. To derive a solution to this pde, we make use of Laplace transforms to map it to the set of ode's given by Equation (2.26).

$$-\overline{C_f}(y,0) + s\overline{C_f}(y,s) = D\ \partial^2 \overline{C_f}(y,s)/\partial y^2, \qquad |y| < 0, \tag{2.26a}$$

$$s\overline{C_m}(y,s) = D_m\ \partial^2 \overline{C_m}(y,s)/\partial y^2, \qquad |y| > 0, \tag{2.26b}$$

$$\overline{C_f} = \overline{C_m}, \qquad |y| = 0, \tag{2.26c}$$

$$-D\ \partial \overline{C_f}(y,s)/\partial y = k_A(-\overline{C_f}(y,0) + s\ \overline{C_f}(y,s)) - \phi\ D_e\ \partial \overline{C_m}(y,s)/\partial y, \qquad |y| = 0, \tag{2.26d}$$

$$\overline{C}(s=0) = \delta(y+y_0), \tag{2.26e}$$

Let region I be the set of y values such that $y < -y_0$, region II be the set of y values such that $-y_0 < y < 0$, and region III be the set of y values such that $y > 0$. The solution to Equation(2.26) in each region is given by:

- Region I: $\overline{C_f} = c_1\ e^{\sqrt{s/D}y}$
- Region II: $\overline{C_f} = c_2\ e^{\sqrt{s/D}y} + c_3\ e^{-\sqrt{s/D}y}$
- Region III: $\overline{C_m} = c_4\ e^{-\sqrt{s/D_m}y}$

These solutions are subject to the constraints that they must have the same values at the region interfaces and that they must satisfy the jump condition, given by Equation (2.27) at $y = y_0$.

$$\int_{-y_0-\epsilon}^{-y_0+\epsilon} [s\ \overline{C_f}(y,s) - D\frac{\partial^2}{\partial y^2}\overline{C_f}(y,s)]dy = \int_{-y_0-\epsilon}^{-y_0+\epsilon} \delta(y+y_0)dy \tag{2.27}$$

Solving for c_1-c_4 using these constraints and letting $\epsilon \to 0$, we obtain

$$c_1 = \frac{\sqrt{D\ s}\ \cosh(\sqrt{\frac{s}{D}}y_0) + (k_A\ s + D_e\sqrt{\frac{s}{D_m}}\phi)\sinh(\sqrt{\frac{s}{D}}y_0)}{s(D + k_A\sqrt{D\ s} + \sqrt{\frac{D}{D_m}}D_e\ \phi)}, \tag{2.28a}$$

$$c_2 = -\frac{D_m e^{\sqrt{\frac{s}{D}}y_0}(k_A\ s - \sqrt{D\ s} + D_e\sqrt{\frac{s}{D_m}}\phi)}{2s(D\ D_m + D_m\ k_A\sqrt{D\ s} + D_e\sqrt{D\ D_m}\phi)}, \tag{2.28b}$$

$$c_3 = \frac{e^{-\sqrt{\frac{s}{D}}y_0}}{2\sqrt{D\ s}}, \tag{2.28c}$$

$$c_4 = \frac{e^{-\sqrt{\frac{s}{D}}y_0}}{\sqrt{s}(\sqrt{D} + k_A\sqrt{s} + \frac{D_e\ \phi}{\sqrt{D_m}})}. \tag{2.28d}$$

Now we let $y_0 \to 0$, so that region II vanishes and c_1 and c_4 simplify to the following,

$$c_1 = \frac{1}{\sqrt{s}(\sqrt{D} + k_A\sqrt{s} + \frac{D_e\ \phi}{\sqrt{D_m}})}, \tag{2.29a}$$

$$c_4 = \frac{1}{\sqrt{s}(\sqrt{D} + k_A\sqrt{s} + \frac{D_e\ \phi}{\sqrt{D_m}})}. \tag{2.29b}$$

The final step is to calculate the inverse Laplace transforms for $\int_{-\infty}^{0} c_1 e^{\sqrt{s/D}y}dy$ and $(\phi + k_A)\ \int_0^\infty c_4 e^{-\sqrt{s/D_m}y}dy$ which give $P(I \to F)$ and $P(I \to M)$ respectively. $P(I \to I)$ can then be obtained by $1 - [P(I \to F) + P(I \to M)]$. The resulting transition probability rules for a time step Δt are given in Equation (2.30)

$$P(I \to F) = \frac{1}{q}\sqrt{D}\ [1 - \exp(\frac{q^2\Delta t}{k_A^2})\ \mathrm{erfc}(\frac{q\sqrt{\Delta t}}{k_A})], \tag{2.30a}$$

$$P(I \to M) = \frac{1}{q}\sqrt{\phi D_e(\phi + \rho_q k_d)}\ [1 - \exp(\frac{q^2\Delta t}{k_A^2})\ \mathrm{erfc}(\frac{q\sqrt{\Delta t}}{k_A})], \tag{2.30b}$$

$$P(I \to I) = \exp(\frac{q^2\Delta t}{k_A^2})\ \mathrm{erfc}(\frac{q\sqrt{\Delta t}}{k_A}), \tag{2.30c}$$

$$q = \sqrt{D} + \sqrt{\phi D_e(\phi + \rho_b k_d)}. \tag{2.30d}$$

Unfortunately, when we compare the results of the two methods we do not see the agreement we were seeing in the case without adsorption as illustrated in Figure 2.7. For this case, we found that an error of approximately 21% remains regardless of how many applications of the coarse-scale method are made. This is severely problematic as there is no analytical solution to compare against in this case, meaning we have no reliable way to see if the problem lies with the transition probabilities or the coarse-scale method (or both).

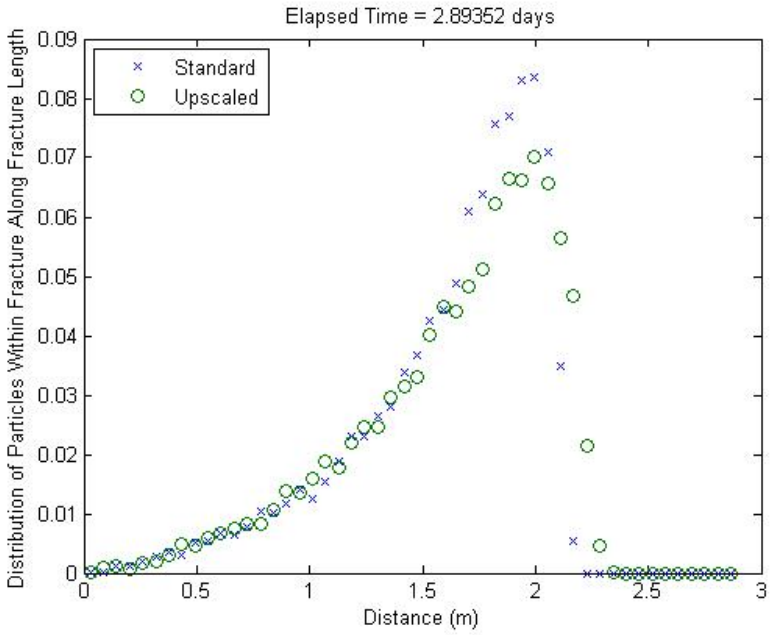


Figure 2.6: Comparing the in fracture concentration distributions obtained by the coarse-scale and high-resolution methods for a case with adsorption.

2.8 Conclusions and Directions for Future Work

The coarse-scale particle tracking methods outlined in this chapter demonstrate a significant computational savings over the high-resolution method when tracking particles through long straight cracks. The computational efficiency of the proposed methods specifically derives from (i) considering time steps significantly greater than the characteristic diffusion time scale across the fracture aperture for particles starting in the fracture (Section 2.4.2) and (ii) exploiting first passage time theory for particles starting in the matrix (Section 2.4.1). Together, these methods can be combined to efficiently model transport in fractured rock including the influence of fracture-matrix interactions. However, they can also be used separately when the assumptions of one method fail to be satisfied but not the other. For example, in the close vicinity of fracture-fracture intersections, reverting to standard particle tracking is more rigorous in the rock matrix, but our in-fracture coarse-scale method is still applicable. Similarly, in the vicinity of a tortuous fracture, using the SFPM may be required to track the particle while it is within the fracture, but our in-matrix coarse-scale method can still be used when the particle works its way into the rock matrix.

An important component in cutting the computational costs is making use of the analytical solution to the one-dimensional sub-problem of finding the particles displacement perpendicular to the fracture length, and then approximating that solution with simple polynomial functions. Although constructing the polynomials is a non-trivial process, using them allows one to estimate the time spent in the rock matrix which in turn allows one to estimate how long the particle was advected along the fracture at a cost of only $O(10)$ flops. We attempted to utilize the same procedure to develop analogous methods for cases where the fracture walls can adsorp particles, but these attempts were less successful for reasons we were unable to pinpoint.

A limitation of our method is that it requires a number of time steps to be taken before the solution is recovered accurately, owing to the fact that the coarse-scale methods can only recover the analytical solution on average. However, the limitation is not a drastic one and through numerical experimentation we have found that only approximately 50 time steps is sufficient to reduce the

error of solutions represented in a spacial domain to only a few percent. Additionally, numerical experimentation has shown that some of our approximations break down for time steps significantly greater than 500 a^2/D, but this is not a serious limitation in light of the fact that the high-resolution method requires time steps on the order of 0.1 a^2/D.

Future work in this area will focus on utilizing the coarse-scaled methods in network simulations and also examining its utility in cases with variable fracture aperture. We will also look into incorporating the analytical breakthrough curves given by Maloszewski and Zuber [48] to directly advance particle positions and times when they apply. Lastly, we hope to sort out the issues appearing in the adsorption case so that we are able to extend these methods to be applicable to such fracture networks.

Chapter 3

Modelling Ice Viscosity for Glacier Flow Simulations

3.1 Introduction

With increased interest in climatic variation due to global warming, there has been an accompanying interest in examining how large-scale ice masses evolve in time. As direct experimentation is not physically feasible, researchers analyzing ice flow problems rely primarily on numerical simulations, treating ice as a highly viscous non-Newtonian fluid. This has come to highlight a major numerical drawback of the traditional power law model for the effective viscosity of ice. The form of this power law model, known as Glen's law, causes the effective viscosity tend to infinity in regions of low strain rate, and this greatly impedes the convergence of numerical solvers [46].

In this chapter, we explore an alternative empirically based model for the constitutive relationship. The proposed flow law model is analytically invertible, like Glen's, which allows computational researchers to use standard velocity-pressure formulations common to many CFD (computational fluid dynamics) problems. By analytic invertibility, we refer to the ability to analytically represent the constitutive relationship in the form $\dot{\epsilon}_e = f(\sigma'_e)$ or $\sigma'_e = f^{-1}(\dot{\epsilon}_e)$, where σ'_e and $\dot{\epsilon}_e$ represent the effective deviatoric stress and effective strain rate respectively. These are defined as $(\sigma'_e)^2 = \frac{1}{2}\sigma'_{ij}\sigma'_{ij}$ and $(\dot{\epsilon}_e)^2 = \frac{1}{2}\dot{\epsilon}_{ij}\dot{\epsilon}_{ij}$, where σ' denotes the deviatoric stress tensor and $\dot{\epsilon}$ denotes the strain rate tensor. For this chapter, repeated indicies in tensor relationships indicate using the summation convention. The proposed law fits empirical data points with accuracy comparable to the traditional model, but avoids the viscosity blow up.

3.2 Overview of Modelling the Stress Strain Rate Relationship for Ice

The constitutive law capturing the non-Newtonian behavior of ice is an important part of ice flow models. The most commonly used form of the constitutive law is Nye's generalization to Glen's flow law [56], which is commonly still called Glen's law for short. It is an empirically based power law model that gained widespread appeal for its simple form and ability to fit empirical data with acceptable accuracy.

Unfortunately, a major drawback of this law has become evident in more recent years with the development of numerical models of ice flow. The problem stems from that the functional form Glen's law causes the effective viscosity to be unbounded, and this significantly impacts the computational efficiency with which the ice flow problem can be solved. This is currently considered one the main difficulties to overcome in computational simulations of ice flow [45].

However, this viscosity blow up is an artifact of using a simple power law model to fit the empirical data points that capture the relationship between stress and strain rate for ice. Although more accurate constitutive laws have been developed since Nye's 1955 paper [47, 70, 32], they have not attained widespread use in the computational community. A primary reason for this is because they cannot be analytically inverted to give the strain rate in terms of stress, which makes them impractical for solvers requiring pressure-velocity problem formulations. While the proposed law does not suffer from this issue, its primary appeal is that it naturally avoids the viscosity blow up inherent in an unmodified Glen's law. This is a feature also possessed by the more accurate, but not analytically invertible, polynomial flow laws [47, 70] so it should not be taken as physically unrealistic.

3.3 Avoiding Numerical Difficulties from Glen's Flow Law

Nye's generalization of Glen's flow law (commonly called "Glen's law" for short) is a widely used, empirically based power law model for the constitutive relationship that relates the stress to the strain rate in ice. The stress-strain relationship that leads to Glen's flow law is of the form

given by (3.1a) or equivalently (3.1b),

$$\sigma_e = B(T)\dot{\epsilon_e}^{1/n}, \tag{3.1a}$$

$$\dot{\epsilon}_e = B(T)^{-n}\sigma_e^n. \tag{3.1b}$$

By applying a fixed stress to an ice sample, one can calculate the effective deviatoric stress and measure the resulting effective strain rate. This allows one to obtain a data point of the form $(\dot{\epsilon}_e, \sigma_e)$. It is important to realize that Glen's flow law is based on fitting these data points and the justification for using it is that the law can capture the empirical relation between σ_e and $\dot{\epsilon}_e$ with reasonable accuracy.

While simple and acceptably accurate, the drawback to formulating the constitutive relationship with a power law is that it leads to the following form for the flow law relating the stress and strain rate tensors:

$$\sigma_{ij} = \eta(\dot{\epsilon}_e, T)\ \dot{\epsilon}_{ij}, \tag{3.2a}$$

$$\eta(\dot{\epsilon}_e, T) = B(T)\dot{\epsilon_e}^{-(n-1)/n}. \tag{3.2b}$$

where η is the effective viscosity for ice. This form for the effective viscosity is a function which grows unboundedly as the effective strain rate, $\dot{\epsilon}_e$, goes to zero. This is problematic for two reasons. Firstly, it has been demonstrated that in low stress/strain rate regimes the value of the exponent, n, appearing in Glen's flow law should be approximately one [60, 69, 75]. This means the viscosity should go to a finite value in that limit and this is not reflected in Glen's constitutive relationship. Secondly, although Equation (3.1a) makes analytical calculations for problems like borehole closure rates reasonably straight forward, it is commonly understood to be a great hindrance to numerical calculations [45].

Taking those issues into consideration along with the fact that the Glen's law is empirically based to begin with, we instead propose the following alternative empirical fit for the constitutive relationship,

$$\sigma_e = s_\sigma(\boldsymbol{\xi}) \ln(\dot{\epsilon}_e / s_\epsilon(\boldsymbol{\xi}) + 1). \tag{3.3}$$

For the present discussion $s_\sigma(\boldsymbol{\xi})$ and $s_\epsilon(\boldsymbol{\xi})$ can just be thought of as functions of temperature akin to $B(T)$ in Glen's law. In general, however, $\boldsymbol{\xi}$ denotes the collection of variables that influence the constitutive relationship apart from stress and strain-rate, including but not limited to temperature. The relationship given by Equation (3.3) leads to the following alternative flow law,

$$\sigma_{ij} = s_\sigma(\boldsymbol{\xi}) \frac{\ln(\dot{\epsilon}_e / s_\epsilon(\boldsymbol{\xi}) + 1)}{\dot{\epsilon}_e} \ \dot{\epsilon}_{ij}, \tag{3.4a}$$

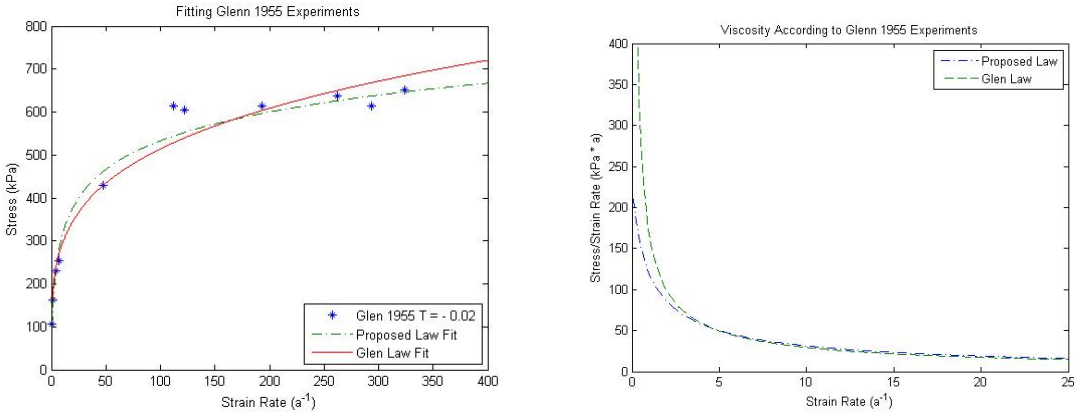
$$\sigma_{ij} = \eta(\dot{\epsilon}_e, \boldsymbol{\xi}) \ \dot{\epsilon}_{ij}. \tag{3.4b}$$

3.4 Similarity of Flow Laws

It may seem as though the different functional form of Equation (3.3) would result in a flow law that is qualitatively distinct from the traditional constitutive relationship used in Glen's flow law, but that is not the case. As an example, Figure 3.1(a) shows the results of using a traditional power law formulation to fit the $T = -0.02$ °C data from Glen's 1955 paper as well as the alternative logarithmic law formulation. The errors in interpolating the raw data in the least squares sense are comparable between the two fits with neither showing a distinct advantage over the other. The same held true overall for the twelve other data sets we considered which resulted from experiments carried out at fixed temperatures [71, 60, 49, 38, 50]. A representative sample of the resulting fits from those data sets is presented in Figure 3.2. The average percent error per data point when using the traditional Glen's law is 9.34% and 9.57% for the log law. Although one law may fit a given set of data significantly better than the other, the overall difference between the two is negligible.

For computational purposes, we are not concerned with the direct relationships between stress and strain-rate, but the effective viscosities of ice derived from them. The effective viscosities corresponding to the fits displayed in Figure 3.1(a) are shown in Figure 3.1(b). While the fits to the

original data are comparable, the resulting effective viscosities from each law are nearly identical save for near the origin. This holds true for all of the other data sets considered. Although the logarithmic law only differs from the standard power law over a very narrow region, the difference is significant when these laws are used in doing numerics because the former does not introduce any unbounded terms.



(a) Glen Data Set with Fitting Functions

(b) Resulting Viscosity Functions

Figure 3.1: The constitutive relationships and resulting viscosity functions given by Glen's power law and our alternative logarithmic formulation. Note that the domain in the viscosity plot has been restricted to the narrow region where the functions visibly differ.

3.5 A Fundamental Issue Concerning Glen's Flow Law

During the course of developing and testing the log law formulation, we noticed an inconsistency with the standard Glen's flow law. If equation (3.1a) is correct, then sets of data collected at different temperatures can be mapped onto the curve given by the function $f(\dot{\epsilon}_e) = \dot{\epsilon}_e{}^{1/n}$ by using (3.1a) and dividing both sides by $B(T)$ to rescale the σ_e ordinate. If one knows the values for $B(T)$ in the experiments under consideration, this will remain true even if the value of n is unknown.

To see if this holds for Glen's flow law, we first recast it in the form suitable to do least squares fits to data sets given by (3.5). This is equivalent to (3.1a) where the fitting constants relate to the parameters in the flow law in that $c_1 = B(T)$ and $c_2 = 1/n$.

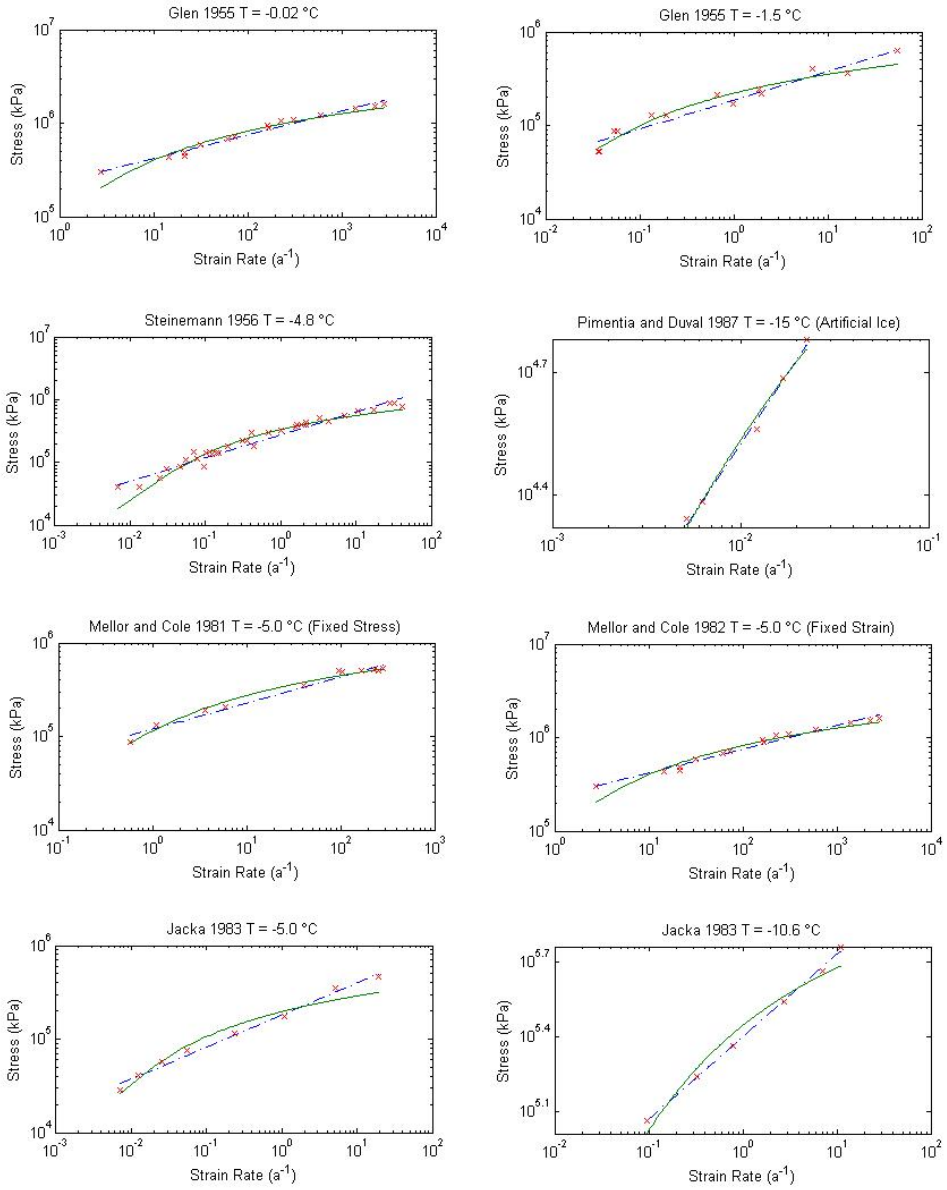


Figure 3.2: This is a representative sample of the data sets considered along with the fits given by Glen's law as well as those of the proposed logarithmic law. The plots are on a log-log scale so fits given by Glen's law are represented by the straight lines and the fits given by the log law are given by the slightly curved lines. While one model may fit an individual data set more accurately than the other, neither shows a distinct advantage when all of the data sets are taken into account.

$$\sigma_e = c_1 \dot{\epsilon}_e^{\,c_2} \tag{3.5}$$

For each of the data sets [30, 71, 60, 49, 38, 50], we fit c_1 and c_2 via a weighted least squares method to determine what their values should be to get the best fit to each individual data set. The weights were chosen so that the least squares method minimizes the square of the relative error for each data point as opposed to the absolute error. This prevents data points at high stress/strain-rate scales from influencing the error to be minimized more than the data points at the lower scales, and treats all points equally in determining the fitted function.

As the tests under consideration were all done at constant temperatures, we are effectively fitting constants to both $B(T)$ and n from (3.1a). Once these numerical values have been determined, we take each data point $(\dot{\epsilon}_e, \sigma_e)$, and rescale the σ_e ordinate by the calculated c_1 from the data set containing that point. This yields a set of data points of the form $(\dot{\epsilon}_e, \frac{\sigma_e}{B(T)})$ or simply $(\dot{\epsilon}_e, f(\dot{\epsilon}_e))$. This eliminates any error we would incur by an imprecise knowledge of the form of $B(T)$ or the exact value for n. The result should be that they all approximately lie along a single curve given by $f(\dot{\epsilon}_e) = \dot{\epsilon}_e^{\,1/n}$, for whatever n happens to be. On a log-log scale plot, this means the points should all be mapped to a straight line. However, as Figure 3.3(a) shows, there seems to be a distinct curve to the scaled data points. The line corresponding to a Glen's law with $n = 3$ has been added to illustrate that it is not necessarily appropriate to use in the low or high strain-rate regimes.

To see what may be going wrong, let's go back to the basic assumption (3.1a) is based on, namely that the relationship between $\dot{\epsilon}_e$ and σ_e is of the following form,

$$\sigma_e = a(\boldsymbol{\xi}) f(\dot{\epsilon}_e). \tag{3.6}$$

where $\boldsymbol{\xi}$ represents all relevant material properties of the ice being deformed (temperature, grain size, water content, etc.). In general, there is no reason to assume the relationship can necessarily take a form where the dependence on $\boldsymbol{\xi}$ and $\dot{\epsilon}_e$ are separable as in (3.6), but the power law functional

form implicitly assumes this to be true. Alternatively, the relationship between effective strain rate and effective stress could be of the slightly more general form given by

$$\sigma_e = s_\sigma(\boldsymbol{\xi}) f(\dot{\epsilon}_e / s_\epsilon(\boldsymbol{\xi})). \tag{3.7}$$

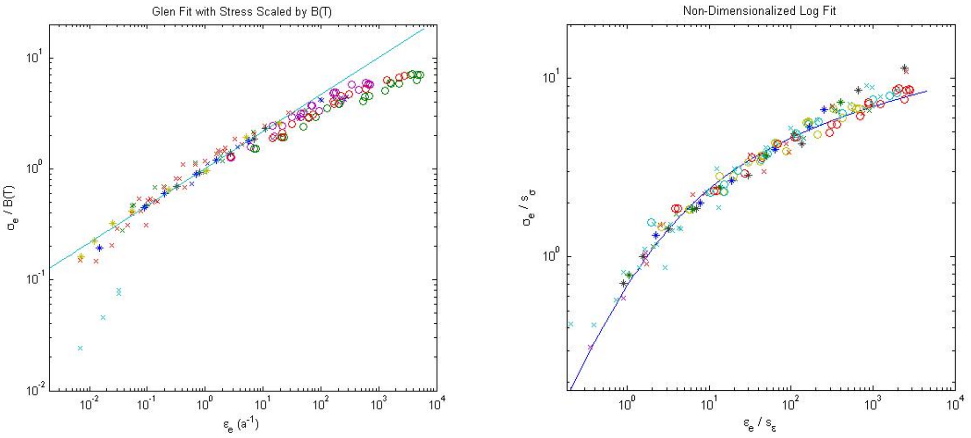
The important difference between (3.6) and (3.7) is that the latter explicitly recognizes the possibility that the dependence of the stress/strain-rate relationship may not be of the form where the functional dependence of σ_e on $\boldsymbol{\xi}$ is separable from its functional dependence on $\dot{\epsilon}_e$. The motivation behind this particular modification is that $s_\epsilon(\boldsymbol{\xi})$ corresponds to capturing the physical dependence of what governs the scale for the strain-rate where $s_\sigma(\boldsymbol{\xi})$ does the same for stress, and there is no reason why these should be combined together as (3.1a) or more generally (3.6) assumes.

As great care was taken to ensure uniform properties in the ice samples within each set of the aforementioned papers, it is reasonable to assume $\boldsymbol{\xi}$ can be taken as constant throughout each individual set of experiments carried out at a fixed temperature. Being that the forms of $s_\epsilon(\boldsymbol{\xi})$ and $s_\sigma(\boldsymbol{\xi})$ are unknown at this point, we set them as the constants c_ϵ and c_σ, which are then estimated using least squares fits to each of the data sets for the following function,

$$\sigma_e = c_\sigma \ln(\dot{\epsilon}_e / c_\epsilon + 1). \tag{3.8}$$

If (3.3) accurately gives the stress strain relationship, then if we take each data point $(\dot{\epsilon}_e, \sigma_e)$, and rescale the σ_e ordinate by the calculated c_σ calculated from the data set containing the given point, all of the points should approximately lie on a single curve given by $f(\dot{\epsilon}_e) = \ln(\dot{\epsilon}_e / c_\epsilon + 1)$. As Figure 3.3(b) shows, this is the case.

There could be other functional forms that pass this test, and perhaps a variable exponent version of Glen's law could also be appropriate. It is simply something we noticed during our research and felt worth pointing out as it may explain why there is so much ambiguity surrounding the rate factor in the traditional Glen's law. Specifically, the ambiguity may stem from that the rate factor inherently combines s_σ and s_ϵ into a single function when there is no a priori reason to



(a) Glen's Law Formulation Check (b) Log Law Formulation Check

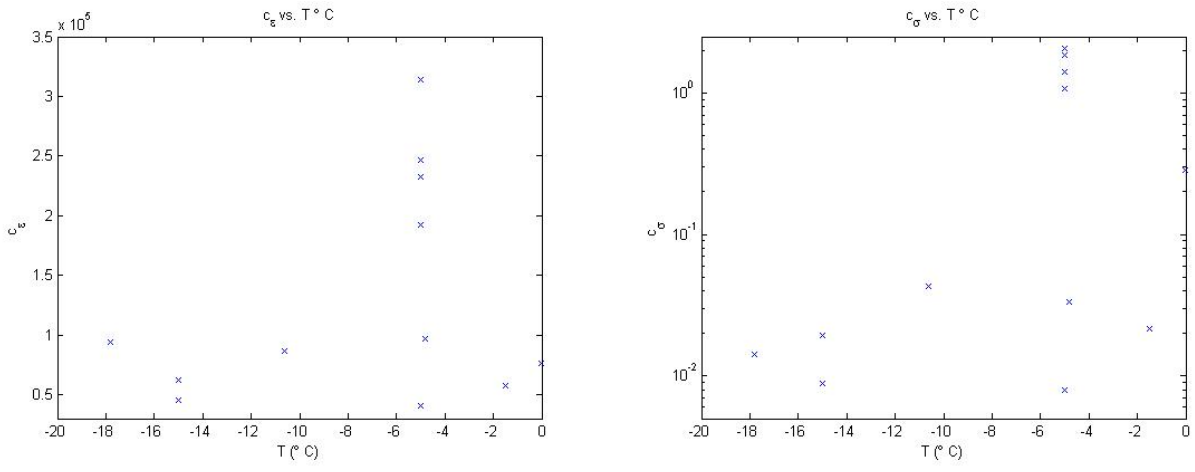
Figure 3.3: a) If the general form of Glen's law is correct, then all of the points shown should be mapped to a single straight line of slope $1/n$ on the log-log scale plot. This is not the case. For demonstrative purposes the curve the points should be mapped to for a Glen's flow law with $n = 3$ has been added as well. b) The equivalent procedure is carried out on the log law and the points are all approximately mapped to a single curve. This curve is given by $\hat{\sigma}_e = ln(\hat{\epsilon}_e + 1)$ where $\hat{\sigma}_e$ is the effective deviatoric stress non-dimensionalized by s_σ and $\hat{\epsilon}_e$ is the effective strain rate non-dimensinalized by s_ϵ. This is what we should expect if the general functional form of the law is correct.

do so.

3.6 Temperature Dependence

The main drawback for the proposed law is that the functions $s_\epsilon(\boldsymbol{\xi})$ and $s_\sigma(\boldsymbol{\xi})$ appearing in the flow law are not yet determined. If temperature is the only material property that plays a role in these functions as is assumed the case for Glen's law's $B(T)$, then these functions reduce to $s_\epsilon(T)$ and $s_\sigma(T)$. We should be able to get some insight on their functional forms by taking sets of experimental data obtained at different temperatures, fitting c_ϵ and c_σ to them, and plotting the temperature dependence of these c's. The results of carrying out this process on the previously mentioned data sets are displayed in Figure 3.4.

The significant scatter in these data points seems to suggest that $s_\epsilon(\boldsymbol{\xi})$ and $s_\sigma(\boldsymbol{\xi})$ do not depend systematically on temperature. This is an absurd conclusion which would suggest that the



(a) c_ϵ vs T (C)

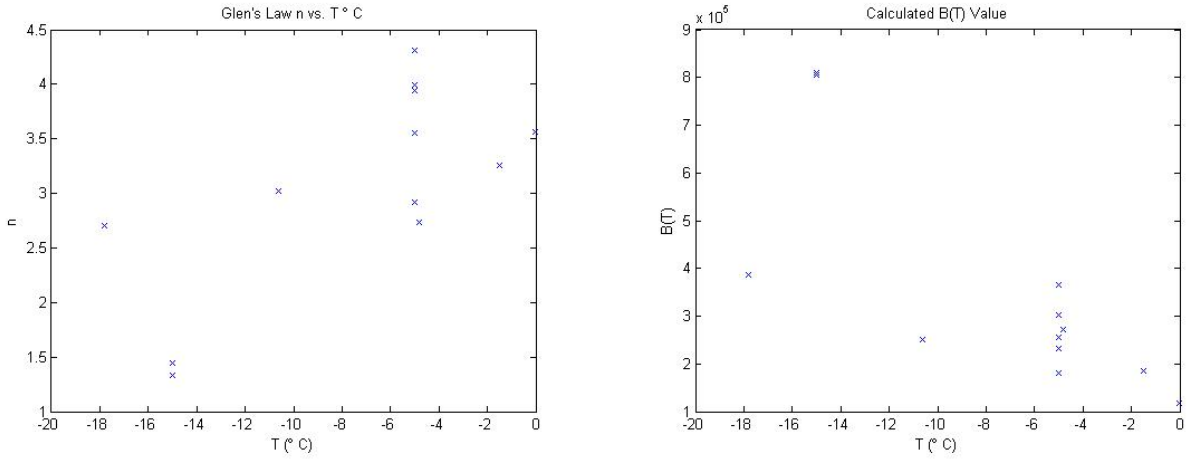
(b) c_σ vs T (C)

Figure 3.4: These plots show c_ϵ and c_σ lack a discernible systematic temperature dependence. As these are data points to determine the functional forms of $s_\epsilon(\boldsymbol{\xi})$ and $s_\sigma(\boldsymbol{\xi})$, this is unexpected. Note that (b) uses a logarithmic scale for the c_σ axis to accommodate the spread of values

logarithmic formulation is fundamentally flawed. However, it may come as a surprise but in all actuality the same can be said for the fits to Glen's law's $B(T)$ as displayed by Figure 3.5. This figure also shows the fairly severe scatter in calculated values for n in Glen's law, which one might have anticipated due to the curve of Figure 3.3(a). While there is no reason to suspect the n in Glen's flow law is temperature dependent, it is plotted against temperature as a means to visually identify the corresponding $B(T)$ value. The severity of the scatter seen in all of these plots may be a result of that there are other significant factors which are not being taken into account. In any case, hopefully they impress upon the reader that there is no obvious empirical reason to choose one law over the other.

3.7 A Pragmatic Compromise

Until studies which analyze the impact of the various material properties of ice on the scales of its stress and strain rate individually or more consistent data in general become available, a rudimentary fix to make the proposed flow law usable for immediate numerical implementations



(a) n vs T (C)

(b) $B(T)$ vs T (C)

Figure 3.5: A similar problem arises with the calculated values of $B(T)$ used in Glen's law. Also, there is a fairly severe scatter in calculated n values.

can be achieved by fitting the logarithmic form of the law directly to Nye's generalization of Glen's flow law. That is, assume Nye's law is correct and use it to generate a large number of data points to which we can fit the logarithmic form of the law to. In doing this, one can determine the functional forms for $s_\epsilon(T)$ and $s_\sigma(T)$ that will mimic a Glen's law. This allows one to still effectively use the traditional Nye's generalization of Glen's flow law, but avoid the numerical difficulties coming from an unbounded effective viscosity. This process has been carried out for a variety of n values, using an Arrhenius model for the associated rate enhancement factor in generating these fits. The necessary information to include temperature dependence in the log model is included in Table 3.1 in conjunction with Equation (3.9). The values of σ_f in Table 3.1 represent the fixed stress values where the power law and the logarithmic law are equivalent irrespective of temperature for each given value of n.

$$s_\epsilon(T) = (B_0 \; exp\left(\frac{T_0}{T} - \frac{C}{(T_r - T)^k}\right))^{-n} \frac{\sigma_f^n}{exp(\sigma_f/s_\sigma) - 1} \tag{3.9}$$

The reason s_σ is constant with respect to temperature for a given n and all of the temperature dependence is captured by s_ϵ is because we are fixing the stress range to fit over as running from

n	s_σ (kPa)	σ_f (kPa)
1.5	492.426	735.595
2	306.249	765.255
2.5	241.492	796.275
3	207.088	823.445
3.5	184.610	845.963
4	168.023	864.361
4.5	154.847	879.391

Table 3.1: This data in conjunction with Equation (3.9) can be used to generate the equivalent of the associated rate enhancement factor for the alternative flow law presented in this chapter.

0-1000 kPa. If we instead perform the fits to be over a fixed strain-rate range, s_ϵ would be a constant and s_σ would be temperature-dependent.

3.8 A Comparison of Numerical Performance

As mentioned earlier, it has been previously demonstrated that in low stress regimes the value of the exponent, n, appearing in Glen's flow law should be approximately one [60, 69, 75]. Having $n \approx 1$ implies that the viscosity is approximately constant in these regions, so instead of the effective viscosity blowing up as $\dot{\epsilon}_e \to 0$, it should go to a finite number. Figure 3.1(b) shows that the proposed flow law does this naturally. In addition to being more physically realistic in this sense, it is also more computationally friendly as the viscosity term is now bounded.

To illustrate the features of the proposed model for the constitutive relationship, we used (3.3) to model the effective viscosity and solved the system of equations given by (3.10) for an idealized test problem. The notation used in the equations is as follows: Ω is the interior domain of the glacier, Γ_B is the basal boundary, Γ_S is the surface, Γ_D is the boundary at the ice divide, $\boldsymbol{u}$ is the velocity vector with components u_1 and u_2, $\hat{\boldsymbol{n}}$ is the unit normal to the given boundary, and p is pressure. This system represents a 2D vertical cross section of an isothermal glacier undergoing steady gravity induced ice flow over a region with mild bumps, no slip basal boundary conditions, stress free conditions on the surface of the glacier, and symmetry boundary conditions at the ice divide:

$$\nabla \cdot \boldsymbol{\sigma} = -\rho \boldsymbol{g}, \quad \text{in } \Omega, \tag{3.10a}$$

$$\nabla \cdot \boldsymbol{u} = 0, \qquad \text{in } \Omega, \tag{3.10b}$$

$$\boldsymbol{\sigma} + p\boldsymbol{I} - \eta(\dot{\boldsymbol{\epsilon}})\dot{\boldsymbol{\epsilon}}(\boldsymbol{u}) = \boldsymbol{0}, \qquad \text{in } \Omega, \tag{3.10c}$$

$$\boldsymbol{u} = \boldsymbol{0}, \qquad \text{on } \Gamma_B, \tag{3.10d}$$

$$\hat{\boldsymbol{n}} \cdot \boldsymbol{\sigma} = \boldsymbol{0}, \qquad \text{on } \Gamma_S, \tag{3.10e}$$

$$u_1 = 0, \qquad \text{on } \Gamma_D, \tag{3.10f}$$

$$\hat{\boldsymbol{n}} \cdot \nabla u_2 = 0, \qquad \text{on } \Gamma_D. \tag{3.10g}$$

The method we used to solve this set of equations was to use the First Order System Least Squares finite element method (FOSLS for short). In this method, the solution is sought by minimizing the following FOSLS functional over an appropriately chosen set of finite element spaces:

$$\mathscr{F}(\boldsymbol{u}, \boldsymbol{\sigma}) = \quad ||\nabla \cdot \boldsymbol{\sigma} + \rho \boldsymbol{g}||^2_{\Omega} + ||\nabla \cdot \boldsymbol{u}||^2_{\Omega} + ||\boldsymbol{\sigma} + p\boldsymbol{I} - \eta(\dot{\boldsymbol{\epsilon}})\dot{\boldsymbol{\epsilon}}(\boldsymbol{u})||^2_{\Omega} \tag{3.11a}$$

$$+ ||\hat{\boldsymbol{n}} \cdot \boldsymbol{\sigma}||^2_{\Gamma_S} + ||\hat{\boldsymbol{n}} \cdot \nabla u_2||^2_{\Gamma_D} \tag{3.11b}$$

Equations (3.10d) and (3.10f) are enforced strongly (meaning they are forced to be satisfied exactly) and so are not represented in the functional. The spaces used for our tests are as follows: $P1$ elements for p, $P2$ elements for the η and $\boldsymbol{u}$, and first degree Raviart-Thomas elements for $\boldsymbol{\sigma}$. The norms we used are all L^2 over the domains where each relationship is to hold.

Because the model glacier in our test is taken to be isothermal and traditionally the associated rate factor, $B(T)$ is taken to be only temperature dependent, we set $s_\epsilon(\boldsymbol{\xi})$ and $s_\sigma(\boldsymbol{\xi})$ to be constants as described in Section 3.7. A simple Picard iteration was implemented to handle the nonlinear η in (3.10) where we initially set η to have a constant value and updated it based on the strain-rate field calculated during the previous iteration. The initial constant was chosen to correspond to the $\eta(\dot{\epsilon}_e)$ resulting from the logarithmic formulation in the limit $\dot{\epsilon}_e \to 0$, and the iterations were carried out until the relative change in the FOSLS functional from one iteration to the next was less than 0.1%.

The numerical advantages of the proposed flow law are mildly evident in that this simple approach to handling the non-linearity does not work when using the traditional Glen's law (regardless of what the initial constant is set to) and the solver will fail to converge unless extra measures are taken. A common measure is to tweak Glen's flow law by adding a small constant, ϵ_0, to the denominator in the following manner,

$$\sigma_{ij} = B(T)(\dot{\epsilon}_e + \epsilon_0)^{-(n-1)/n}\ \dot{\epsilon}_{ij}. \tag{3.12}$$

However, the benefits are much more apparent from the results of Table 3.2. This table shows how the logarithm based flow law compares to the modified Glen's flow law given by Equation (3.12) in terms of number of Picard iterations to reach convergence and the norm of the converged FOSLS functional. Although the number of required iterations and the values of the converged functionals are marginally better for the logarithmic formulation, the important feature to note is that it always converged whereas the convergence when using the modified Glen's law was highly dependent on the value of ϵ_0 in conjunction with the mesh size.

	Log Law	Glen's Law $\epsilon_0 = 10^{-3}$	Glen's Law $\epsilon_0 = 10^{-4}$	Glen's Law $\epsilon_0 = 10^{-7}$
$h = 1/2$ Iterations	18	19	X	X
$h = 1/4$ Iterations	18	19	35	X
$h = 1/8$ Iterations	17	18	20	21
$h = 1/2$ Functional Norm	0.319767	0.421258	X	X
$h = 1/4$ Functional Norm	0.127189	0.169377	0.28677	X
$h = 1/8$ Functional Norm	0.0616705	0.0815971	0.11208	0.150462

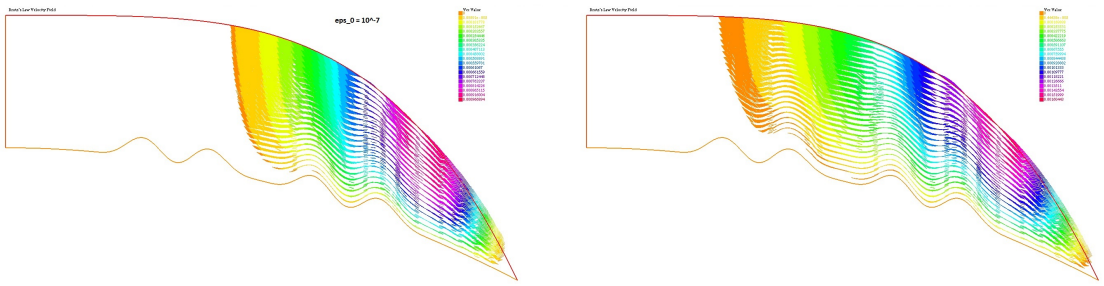
Table 3.2: Performance of the log law as well as modified Glen's laws with several values of ϵ_0. Here, h is the mesh size in km and X's signify that the solver does not converge for the given model. The log law performs slightly better than any of the modified Glen's laws regardless of mesh size in that it requires fewer iterations and attains a lower value of the FOSLS functional. More importantly, the convergence when using Glen's law is highly sensitive to the value chosen for ϵ_0, whereas the logarithmic law does not suffer from this drawback.

Although the focus of this work is on developing an acceptable alternative to Glen's law, there is a feature of the FOSLS methodology worth pointing out. The way this method satisfies the constitutive relationship is by minimizing the difference between the deviatoric stress tensor, $\boldsymbol{\sigma}' \equiv \boldsymbol{\sigma} + p\boldsymbol{I}$, and the strain-rate tensor multiplied by the effective viscosity, $\eta\ \dot{\boldsymbol{\epsilon}}$. This means it

is indifferent as to whether the constitutive relationship is given in the form $\boldsymbol{\sigma}' = g(\boldsymbol{\xi}, \dot{\epsilon}_e)\dot{\boldsymbol{\epsilon}}$, where $\boldsymbol{\sigma}'$ is given in terms of $\dot{\boldsymbol{\epsilon}}$ as pressure/velocity formulations require, or in the form $\dot{\boldsymbol{\epsilon}} = f(\boldsymbol{\xi}, \sigma_e)\boldsymbol{\sigma}$. Although the flow law presented in this chapter is invertible and we could have just as easily used a pressure/velocity formulation, it is worth mentioning this feature because there has been a wealth of research developing more accurate constitutive models for ice, which are of the form $\dot{\boldsymbol{\epsilon}} = f(\boldsymbol{\xi}, \sigma_e)\boldsymbol{\sigma}$ [32, 47, 70]. Unfortunately, these generally lead to a function f that is not analytically invertible and thus impractical to use in formulations which do not directly treat stress as a dependent variable. Since FOSLS methods can be set up to only focus on minimizing the error in the constitutive relationship, it does not matter which form one casts it in; that the improved models are not analytically invertible is not a stumbling block for this type of solver.

3.9 A Method to Obtain Empirical Viscosity Values

The flow fields generated via the FOSLS method when using the logarithmic law as well as that for the modified Glen's law with $\epsilon_0 = 10^{-7}$ are presented in Figure 3.6. The qualitative features of the two laws are very similar overall. However, there is a notable difference in the magnitudes of the velocities where the peak velocities seen when using the logarithmic law are approximately 60% higher than those seen when using the modified Glen's law.



(a) Glen's Law Flow Field (b) Log Law Flow Field

Figure 3.6: These are the computed velocity fields when using the logarithmic law and Glen's flow law. Note, the plots are not to scale; the glacier is 1 km thick at the divide and runs a length of 50 km. The plots are qualitatively very similar but the peak velocities seen when using the logarithmic law are approximately 60% higher than those when using Glen's law.

As shown previously in Figure 3.4, the effective viscosities are identical for practical purposes save for extremely low strain-rates, so this discrepancy was unexpected. What we did not foresee is that the empirical measurements made to determine the stress strain-rate relationship were routinely taken at deviatoric pressure scales that are 100 to 1,000 times larger than the scales of interest to the problem, and that the power law models largely rely on extrapolation to infer the relationship between the quantities of interest. The reason data points were routinely taken at such impractical scales largely stems from that it is less costly to carry out high stress experiments as they do not need to be conducted over as long of a period of time as low stress experiments. The result is that although the models only differ significantly over a very narrow region of the available data, that narrow region turns out to be the only region that is actually relevant for glacier modelling.

That being the case, we sought a means to see which law is more physically accurate. Our previous model comparisons examined the similarities and differences in how the log law and Glen's law fit the empirical data points measuring the stress strain-rate relationship, but, as stated previously, there is no distinct winner between the two in that area. However, what we are interested in is not this relationship, but the effective viscosity this relationship implies. To this end, we developed a method to obtain empirical viscosity values from each of the stress strain-rate data points, allowing us to directly compare how the two models fit the measured values of the quantity pertinent to modelling ice flow rather than simply inferring it as was done in previous work [56, 47, 70, 71, 60, 49, 38, 50].

To begin explaining our method, recall that the general stress strain relationship is given by Equation (3.13), and the problem is to find η,

$$\sigma_{ij} = \eta(\dot{\epsilon}_e, \boldsymbol{\xi})\ \dot{\epsilon}_{ij}. \tag{3.13}$$

Now recall that the effective deviatoric stress and strain rate are defined as $(\sigma'_e)^2 = \frac{1}{2}\sigma'_{ij}\sigma'_{ij}$ and $(\dot{\epsilon}_e)^2 = \frac{1}{2}\dot{\epsilon}_{ij}\dot{\epsilon}_{ij}$; where the repeated indicies again indicate using the summation convention.

As Equation (3.13) holds for each of the components of the tensors involved, we can multiply the respective sides of the equation for one component by the equation for another component to obtain,

$$\sigma_{kl}\sigma_{ij} = \eta^2(\dot{\epsilon}_e, \boldsymbol{\xi})\ \dot{\epsilon}_{ij}\dot{\epsilon}_{kl}. \tag{3.14}$$

In summing the respective sides of Equation (3.14) over all possible values of i, j, k, and l, we obtain,

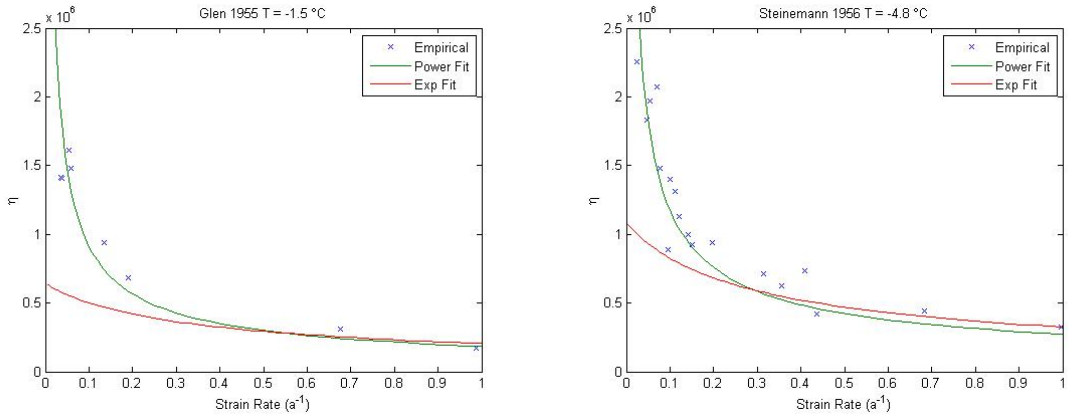
$$2\ (\sigma_e)^2 = \eta^2(\dot{\epsilon}_e, \boldsymbol{\xi})\ 2\ (\epsilon_e)^2. \tag{3.15}$$

From this, we can cancel the factors of 2 and take the square roots of both sides to get the following relation between effective strain rate and effective deviatoric stress,

$$\sigma_e = \eta(\dot{\epsilon}_e, \boldsymbol{\xi})\ \epsilon_e. \tag{3.16}$$

Traditionally the methods for deriving a viscosity relationship have been to fit curves through data points of the form (ϵ_e, σ_e) to infer what the viscosity should be. However, Equation (3.16) allows us to directly obtain empirical values for the viscosity. Specifically, if one divides the value of the σ_e ordinate by the value of the ϵ_e ordinate, then the resulting coordinate pair is of the form $(\epsilon_e, \eta(\epsilon_e))$. This gives us a method to directly observe what the empirical viscosity is for each of the data points. As the flow laws only differ significantly in regions of low strain rate and these are the prevalent values in real world glaciers, we examined how the two flow laws compare in regards to how well they fit empirical values of the viscosity in those regions. Figure 3.7 shows characteristic results for the data sets considered.

Although the infinite viscosity of Glen's law is nonsensical and Figure 3.3(a) demonstrates that it is not a "law" in the sense that it does not fit multiple data sets, we found that it nonetheless matches the empirical values of viscosity at low strain rates more successfully than does the log law when using the values for $c_1,\ c_2,\ c_\sigma$, and c_ϵ obtained for each data set under consideration.



(a) Glen's Law Flow Field

(b) Log Law Flow Field

Figure 3.7: The power law formulations do a better job of fitting the empirical values of viscosity than does the log law (recast as an exponential in the plot as the independent variable has switched).

What makes this especially surprising is that there is no underlying physical basis Glen's law was derived from [31], and the law actually fits the empirical values for viscosity better than it does the stress-strain rate data points it was originally derived from in each case. Still, the fact of the matter is that it matches effective viscosities at low strain-rates extremely well, and we found it difficult to argue in favor of using the log law over Glen's. Because of these findings, we ceased pursuing this avenue to make the equations governing glacier dynamics more amenable to multigrid solvers.

3.10 Conclusions and Directions for Future Work

In examining the relationship between stress and strain rate in ice, we have found that one can fit empirical data equally well when using a power law model or a logarithm as the basic functional form for the constitutive law. The favorable features of our proposed log law model are that it avoids the viscosity blow up inherent in Glen's law, and it fits non-dimensionalized data points more accurately as demonstrated in Figure 3.3. The favorable features of Glen's law is that it provides a more accurate fit to empirical viscosity data points in the very low strain rate regime, and this is where most glaciers are active.

Although this scale issue makes the logarithm model for viscosity questionable for use in

most glacier dynamics, it is still applicable to quickly moving glaciers where the two laws overlap. The benefit of using the logarithm based law over the power law in such a case is that the latter's viscosity blow up in regions of low activity can cause errors to propagate and pollute the solution in the more quickly moving regions that are of practical interest. Also, there are a number of other shear thinning fluids for which the model may be appropriate, and the technique of fitting a logarithm based model to a power law to avoid viscosity blow up would still be applicable in any case.

A direction for future work dealing with the ice viscosity blow up is to try to exploit the fact that while the viscosity blows up as the strain rate goes to zero, the product of strain rate and viscosity together goes to zero. The blow up is the result of splitting up this single term that goes to zero, given by $\dot{\epsilon}_e$, into two terms. One of these terms blows up to infinity, given by the effective viscosity, and the other rapidly goes to zero, given by the strain rate component. As the combination of these terms is well behaved, it appears that the difficulties arise primarily by trying to force the problem to fit standard flow solvers.

One unresolved issue we came across in our studies is the scatter in data points from which temperature dependence should be derivable, as highlighted in Figures 3.4 and 3.5. These data points show a lack of systematic dependence on temperature, which suggests that there may be unaccounted for parameters which these factors strongly depend on. As this term is traditionally taken to be solely dependent on temperature, this merits further attention.

Chapter 4

Community Finding

4.1 Introduction

For real world systems represented by graphs, such as social or information networks, a common feature of interest comes from finding groups of vertices that share more and/or stronger connections with each other than they do with the rest of the vertices in the graph. Community detection is the problem of finding such groups, and is particularly significant to disciplines such as sociology or biology where systems under study are commonly represented as graphs.

The target application for community detection depends on the network under study, and the notion of what makes one grouping "better" than another depends on the concept used to define a community for a given network. For protein-protein interaction networks, the goal of community detection may be to find groups of protein interactions involved in the same biological function [17]. If it is a network comprised of items that are purchased together, the goal of community detection may be to make purchase recommendations to customers. When a network comprised of words people associate with a target word, the goal of community detection may be to find the different meanings of that target word [59]. Or in the case of a network comprised of social interactions, the goal of community detection may be to find communities in the literal sense. Community detection is a problem that arises in a myriad of applications.

4.2 Overview of Community Detection

Although community detection is often referred to as though it is a single problem, in actuality it is a body of related problems. The reason it is erroneous to view it as a single problem lies in that the definition of "community" means different things in different contexts; the organizational structure for a "community" of synonymous words in a WordNet graph has little if not nothing to do with the organizational structure for a "community" of football teams in a conference or the "communities" of taxonomic distinctions in a food web. Different problems inherently have different notions of community, and it is baseless to claim that these different qualitative descriptions for communities all have the same underlying mathematical structure. All any particular algorithm can do is try find the types of communities it was built to find, and if the network an algorithm is being applied to does not define communities in the same way that particular algorithm does, then applying it to the network will not produce the desired results.

While it is generally accepted that the basic problem of community detection is to find groups of vertices that are more interconnected with each other than they are to vertices in other groups, there are numerous questions to consider when approaching this problem. What specific properties should a set of nodes possess in order for it to be considered a good community? Should one allow for a vertex to belong to multiple communities, or is it desirable for each to have a unique community assignment? Are the target communities roughly the same size, or does this vary? Should one use an divisive approach where all the vertices belonging to one community and successively divide that community into some final partitioning, or should one take an agglomerative approach with each vertex comprising its own community and successively clustering these communities together? Is a hierarchical community structure expected within the network? There are different ways to answer each of these questions, which in turn leads to different general techniques for approaching the problem.

4.2.1 Graph Partitioning

Graph partitioning is the problem of dividing the vertices of a graph into a predefined number of groups of predefined sizes such that the number of edges lying between groups, known as the cut size, is minimal [28]. Although an undesirable feature in many applications, specifying the number of clusters is necessary to avoid the trivial solution where all the vertices belong to one group. Although the trivial solution has the minimal possible cut size of zero, it also does not provide any useful information. Similarly, specifying the desired group size is also necessary, as it avoids simply picking off the least connected vertices as their own groups. Still, these are only limitations when they are not known, and there are several scenarios of practical interest where they are. For instance, such techniques are applicable to scientific computing problems where the goal is to minimize communication costs as one knows the number of cores they have to work with [61]. Similarly, they are also applicable to partitioning circuits onto circuit boards such that the wiring between boards is minimized [26]. Still, such a limitation is crippling to applications where the number and sizes of communities is the intended goal of community detection.

4.2.2 Embedding in a Metric Space

Partitional clustering techniques are similar to graph partitioning, but instead of optimizing cut size they embed all the nodes into a metric space and optimize a cost function relying on that metric. Typical cost functions [28] include: minimum k-clustering, k-clustering sum, k-center, k-median, and the popular k-means. The goal of minimum k-clustering is to partition the vertices into k clusters such that the largest diameter of the clusters is minimized, where the diameter of a set of vertices is defined as the largest distance between any two points in the set. K-clustering sum is the same as minimum k-clustering, but the cost function to minimize is based on the average distance between vertices in a cluster as opposed to just the largest such distance. For the k-center method, one defines k centroids representing cluster centers and seeks to minimize the largest distance between any clustered point and its corresponding cluster centroid. The k-median method is the

same as k-center, but seeks to minimize the largest average distance from the centroids as opposed to the maximum. Lastly, the k-means method seeks to minimize the sum of the distances between each node in the graph and its corresponding cluster centroid. For each of these methods, although one need not necessarily specify the number of members each group should have as is required by graph partitioning techniques, one still needs to prespecify the number of desired communities as well as formulate a meaningful metric space for the embedding.

Spectral clustering [3] offers one means to embed vertices into a metric space in a way that helps reveal clusters. Spectral methods take k of the eigenvectors of an adjacency matrix (or a matrix that is some function of S), and use the values of the entries of these eigenvectors to embed each node into a k-dimensional space. For example, if v_j represents the j^{th} eigenvector being used for the embedding, then a node i would be represented by the coordinate $(v_1(i), v_2(i), ..., v_k(i))$. The advantage of doing so is that this naturally tends to spread out vertices into distinct groups, with the distinction generally improving with the number of eigenvectors used. Once such an embedding is accomplished, clustering can be performed using standard partitional clustering methods.

4.2.3 Quality Function Optimization

Another broad family of techniques focus on optimizing specific functions meant to capture what it means to be a "good" community. An example of a quality function from the previously mentioned techniques is to minimize the cut size of the graph. A variation of this is to minimize what is known as known as the conductance. Let C be a subgraph of a graph G, k_C be the total degree of C, $k_{G\backslash C}$ be the total degree of the rest of the graph $G\backslash C$, and $c(C, G\backslash C)$ be the cut size of C. With this terminology in place, the conductance, $\Phi(C)$ is as defined in Equation (4.1) below.

$$\Phi(C) = \frac{c(C, G\backslash C)}{min(k_C, k_{G\backslash C})} \tag{4.1}$$

However, this quality function inherits the one of the main issues encountered in using simple cut size to form partitions, namely that one must specify the number of partitions in the end result. To avoid this, it is preferable for the number of communities to be incorporated as part

of the quality function. One of the most commonly used quality function that accomplishes this is modularity [54]. Modularity is based on comparing the edge distribution of a network against one for which there is no community structure. Let A be the adjacency matrix for a graph, m be the total number of edges, P_{ij} be the expected number of edges between vertices i and j in the null model graph (a copy of the graph with the community structure removed), δ be the Kronecker delta function, and C_k be the community to which node k belongs, then the modularity function for a matrix is given by Equation (4.2) below.

$$Q = \frac{1}{2m}\sum_{ij}(A_{ij} - P_{ij})\delta(C_i, C_j) \tag{4.2}$$

Ideally, Q should be zero whenever the adjacency matrix had edges distributed in a manner that is consistent with a model where there is no community structure present, and should be positive when the number of edges within communities is greater than what is expected from the null model. Aside from incorporating the number of communities as a factor in the function to be optimized (through the $\delta(C_i, C_j)$ term), a benefit of using modularity as the quality function is that it forces one to address the questions pertinent to community finding. What does it mean to be a member of a community, $\delta(C_i, C_j)$? What does it mean for there to be no community structure, P_{ij}? The drawback is that these are not easy questions to answer, especially so regarding P_{ij}. Although the choice of null model is somewhat arbitrary and generally not known for a given real world network, a common choice is to take the null model as a graph that has the same number of edges as the original network but distributed so that each node has the same probability to be connected to any other node. This choice is based on the intuition that one should not expect a community structure in a random graph. However, it has been shown that random graphs can nonetheless have large modularity values with an appropriate partitioning of the nodes [65], despite the fact that Q should be zero under these conditions. This highlights how the question of what P "should" be for a real world network is a non-trivial one to address, and even having the appropriate P in hand does not guarantee a meaningful partitioning. That aside, there exists a host of successful

algorithms that operate based on directly maximizing modularity [54, 8, 19, 76].

4.2.4 Hierarchical Clustering

For networks for which there is reason to suspect a hierarchy of community structures, one may apply hierarchical clustering algorithms. Examples of such networks would be those corresponding to interpersonal communications in military organizations, or the network of cells comprising a tissue in a network of tissues that in turn comprises an organ. Hierarchical clustering does not refer to a stand alone method, but a technique employed by many community detection algorithms. This technique focuses on identifying clusters of vertices with high similarity (under some given similarity measure), and are either agglomerative or divisive in how they handle these clusters. Agglomerative hierarchical clustering techniques start with many small clusters which are then merged if their similarity measure is sufficiently high; they work from the bottom up. Divisive hierarchical clustering techniques start with large clusters (e.g. the "cluster" comprised of all the nodes in the network) which are then divided into smaller clusters by removing edges with low similarity; they work from the top down.

4.2.5 Novel Approaches

The approaches discussed up to this point involve techniques that fall into fairly broad categories, many of them originally developed in other fields and later adapted for community detection. However, there are a number of novel approaches developed specifically for community detection that do not neatly fit into any of these divisions.

4.2.5.1 Clique Percolation

Palla et al. [59] took a unique approach to community detection. Instead of looking at a community in terms of link density between vertices, they look at it as a problem of reachability. They define the notion of a k-clique, which is a fully connected subgraph of k vertices. One k-clique can reach another clique if they share k-1 vertices, a process they call clique percolation. A

k-clique community is then defined as groups of nodes that can be reached via k-clique percolation. This process naturally allows communities to overlap as there is nothing preventing a node from belonging to multiple distinct k-clique communities. Also, it has a natural hierarchical structure as any $(k+1)$-clique community necessarily lies within a k-clique community. Unfortunately, implementing this method requires computing all of the cliques in the network, an exorbitantly expensive process that scales like $O(e^n)$ where n is the number of nodes in the network.

A related piece of work is Rees and Gallagher's collective friendship group inference method [63]. The idea underlying their method is to use the local community structure of the network from each individual node's perspective as a base to build community structure of the network. The main tool by which they form local perspectives of community structure is by using egonets. An egonet is a sub-graph of the full graph representing a network that consists of a given node, the nodes it shares an edge with, and all the edges between those nodes. The given node which the egonet is constructed around is called the egocentric node. Clusters of nodes that remain connected in an egonet when the egocentric node is removed are taken as local perspectives of the overall community structure. These are used to form "friendship groups" by adding the egocentric node back into each of the found clusters. These friendship groups play the same role as the cliques in the clique percolation algorithm, and are agglomerated in the same fashion, namely by starting with one as a community base and then expanding the community by including new friendship groups that satisfy some set similarity criterion. This results in an algorithm that is highly parallelizable and scales well with respect to the number of nodes present in the network.

4.2.5.2 Flow Analysis

Rosval and Bergstrom [67] depart from conventional methods by looking at community detection as the problem of building the most concise two level description for the path of a random walker on the network. This entails decomposing the network into modules that compress the description of probability flow through the network. Specifically, given a module partition M of n nodes into m modules, they seek to minimize the average description length of a single step which

is given by Equation (4.3).

$$L(M) = q\ H(Q) + \sum_{i=1}^{m} p_i\ H(P^i) \tag{4.3}$$

In this equation, q is the probability to transition from one module to another, p_i is the probability for a walker starting in module i to end in module i, $H(Q)$ is the entropy of the module names, and $H(P^i)$ is the entropy of within-module movements. Entropy in this case is defined as $H(X) = -\sum_1^n p_i \log(p_i)$ where p_i is based on the visit frequency to the i^{th} node contained in the set X, which contains n nodes total. The benefit of making use of this quality function as opposed to modularity is that it naturally applies to both weighted and directed graphs, whereas standard modularity does not.

4.2.5.3 Label Passing

The label passing method [62] is a diffusion based approach which gives each node a label which it then passes to its neighbors. Once this step is completed, each node adopts the label that is shared by the majority of its neighbors and the process repeats. Eventually, a consensus is reached where all the nodes in a single community have a distinct label differing from those in other communities. Although this produces a very computationally efficient algorithm, the primary issue with this technique is that it creates "the rich get richer" scenarios where very large communities can develop that do not correspond to anything meaningful.

4.2.6 Overlapping Communities

The approaches discussed thus far primarily focus on notions of community where nodes can belong to only one community; the only exception to this is the clique percolation method. However, it is commonly understood that there is nothing preventing one from being a member of multiple communities, and there has been a wealth of research in recent years focusing on the problem of overlapping community detection. A thorough overview of the state of the art for

overlapping community detection is given in Xie et al. [78], and we go over some of the highlights here.

The key insight of the clique percolation idea that allowed for overlapping community detection was switching from viewing communities as collections of nodes to viewing communities as collections of edges. As nodes are generally connected by multiple edges to the rest of the network, one can assign the node to multiple communities depending on how the edges are clustered. Moreover, clustering edges provides a more natural setting for community detection; it is meaningless to claim that a set of nodes form a community if the edges connecting them do not support that claim, and techniques like modularity maximization are essentially ways of clustering edges indirectly. The first researchers to explicitly acknowledge basing the notion of communities on edges was Evans and Lambiotte [27], where they make the observation that one can approach the problem of detecting overlapping communities by running standard node clustering algorithms on the line graph for a given network.

This is not to say that edge clustering is the only method to approach overlapping community detection. Some alternative methods come from generalizing label propagation to multiple communities [64, 35]. However, one of the most common alternatives to edge clustering is to use stochastic models capable of modelling a network with community structure as a means to detect communities in real world networks [55, 34, 5]. Although stochastic models in general do not necessarily require community overlaps, any such model for artificially generating a network with overlapping community structure inherently involves parameters governing the number and sizes of the communities generated. The basic idea behind how these methods work is that if one believes a given real world network follows a given stochastic framework, then a potential method for community detection is to fit the parameters governing community generation for the model to the observed network and infer the community membership of the nodes.

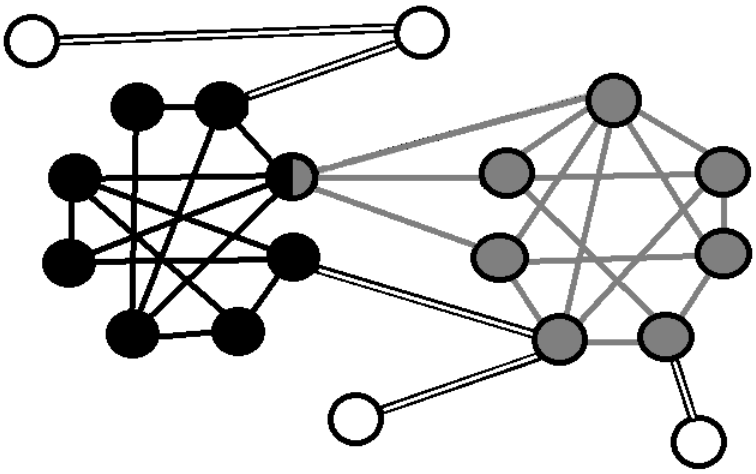


Figure 4.1: Example network with community structure.

4.3 Proposed Approach

Our work focuses on community detection with respect to social networks representable by undirected/unweighted graphs. We assume communities are fundamentally defined in terms of the properties of the edges comprising it, so we approach community detection as an edge clustering problem. As clustering edges naturally allows a node to belong to multiple communities based on how the edges it is connected to are clustered, this conforms well with a common feature of social networks where communities can overlap with one another. Our interpretation of the community detection problem is that it is an inherently multiscale problem, where there are three primary viewpoints on what it means to be a community that need to be taken into account.

Using the graph depicted in Figure 4.1 as a model network, several questions naturally arise on how to describe the community structure present. Firstly, what does it mean for a node to view itself as a member of a community; what differentiates connections with fellow community members from connections with members of other communities? Without addressing this question, there is not a principled understanding of why a node belongs to the community it does. Secondly, what does it mean for a collection of nodes to comprise a community; what property differentiates

the sets of black and gray edges from the white ones? Without addressing this question, there is not a principled understanding as to why the collection is a "community" in the context of the problem and not just a simple collection. Lastly, what are the desired properties the collection of communities should possess at the full network level? Specifically, what should be done with nodes that do not decisively belong to any particular community; how should one handle the white nodes? Although not as fundamental as the previous viewpoints, this is an important question to address because it is the level at which the researcher considering the community detection problem will have to interpret the data.

Uncovering answers to these questions involves addressing how the community structure should appear at three distinct scales. These scales are at the level of: individual nodes, individual communities, and the network as a whole. Each of these scales involves quantitative features of community structure that are not accurately represented at the other scales, but are important for defining the notion of community.

Our approach is to answer the question of what it means for a node to view itself as a member of a community is to find a way of describing individual nodes in a way that captures what their perspective would be of communities it belongs to. Being that the only means available to describe any particular node is the network itself, it seems sensible that a good way to describe how a node fits into the surrounding network makes use of its local view of the network, the node's egonet as defined in Section 4.2.5.1. Keeping in mind that communities in many settings are comprised of the edges linking nodes, the way we describe individual nodes should be amenable to the overall goal of collecting them together based on the properties of their edges. This implies that one should use sets of edges local to a given node to describe the relation of that given node to the rest of the network, where the nature of the sets reflect the specific community detection problem under consideration. We will call such sets of edges *edge descriptor sets*.

Although the specific nature of what makes an edge set appropriate to use is dependent on the nature of a given community detection problem, we adopt the general heuristic that if a node belongs to a community, then it should be a member of a clique lying fully within that community.

The motivation for doing so comes from reflecting on the nature of socially based communities. Being a member of a certain community often coincides with developing a circle of friends in that community. In an egonet, such structures would be represented by a set of nodes that either forms a clique or at least does so approximately. Larger cliques are presumably more important for describing the communities a node would see itself as belonging to, so they are the most important ones to determine accurately. In the situation where a node's egonet contains several sets of edges that approximately form cliques that are largely disjoint from one another, this is an indication that the node belongs to multiple distinct communities. This chain of reasoning suggests using approximate cliques that are largely disjoint from one another as the edge descriptor sets for a node, as it provides a means to not only accurately gauge not only how many distinct communities, but how strongly that node is connected to the community. To find these specific types of clique like structures, we developed a spectral clustering method tailor made for the purpose; the details of which are discussed in Sections 4.3.1.1 through 4.3.3.3.

The answer to the question of what it means for a collection of nodes to view itself as a community hinges upon the answer to the previous one, but is also dependent on the specific community detection problem being addressed. With edge descriptor sets for each node describing what it locally views as the communities it belongs to, our goal is to aggregate these sets into what would be perceived as a community at a coarser grained level. Since this coarser grained view entails qualitative features we believe a community should have, it seems that the natural approach to take is to use quality function optimization.

Though simple, a sensible quality function to use is the density of connections between nodes involved in a community. The optimization portion is to include as many nodes within the community, while still remaining above a user supplied edge density threshold. The motivation for this choice is that a reasonably general heuristic to expect communities to adhere to is that they have an edge density substantially higher than the average density of the network. The reason that the edge density threshold is taken as an input is because it is something we see as dependent on the scale of the structures one is trying to extract from the network; lower threshold densities

correspond to looser definitions of what it means to be a community and higher threshold densities correspond to tighter ones. It should be up to the user to decide what they are looking for.

Lastly, the answer to the question of what properties of a collection of communities should have at the level of the entire network depends on the expectations of the researcher applying the algorithm. As a general heuristic, we require that there are no unclustered nodes in the network, where any node remaining unclustered will be binned with the community it shares the most edges with; a process carried out iteratively if needs be. We also prune out smaller communities detected, subject to the constraint that all nodes are represented in at least one community. The motivation for this requirement simply comes from that as a human being, we do not want to have to sort through an unwieldy number of communities

Although we developed this train of thought independently, during the time of writing we discovered the collective friendship group inference method [63, 64] which shares some similarities. The collective friendship group inference method is motivated by using edge descriptor sets based on the egonets of individual nodes, which are aggregated to form communities. The main differences between our method and theirs is that we consider multiple levels of community viewpoints, whereas their method solely relies on the viewpoint at the level of the individual node. The reason we see this as problematic is that certain features that characterize communities at a larger scale, such as edge density, do not meaningfully carry over to a purely local scale. Moreover, there generally is no reason to assume that the nature of community structure is scale invariant as their approach implicitly does.

4.3.1 Node Scale Features of Community Structure: Edge Descriptor Sets

Because we are interested in identifying communities from the arrangement of edges, we introduce the notion of edge descriptor sets as a general term for using sets of edges to describe each node. The cliques (or friendship circles) a node belongs to, would be examples such edge descriptor sets.

In our approach, we assume that a node is more likely to belong to a community if it has

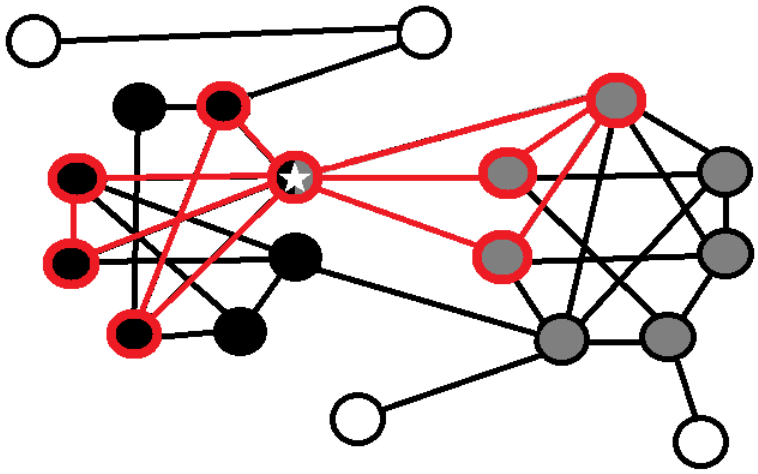


Figure 4.2: A depiction of the egonet for the starred node in the network; all of the red nodes/edges are included in the egonet.

many mutual friends within that community. This suggests that edges linking members of the same communities should be densely inter-connected. The prototypical example of such a cluster of edges if formed by a clique. Furthermore, we note that if a node belongs to several relatively large cliques that are mutually disjoint, then this node may be at the intersection of (mostly) disjoint communities. Guided by these simple heuristic principles, we propose to extract sets of edges that are both densely connected and largely disjoint from each other from each node's egonet to serve as the edge descriptor sets for that node. Each set can be thought of as encoding the fine-scale features of community structures present in the network.

We approach the task of extracting edge descriptor sets by first using a simple spectral clustering method to sparsify the local egonet defined around each node. The goal of the sparsification is to reveal the largest disjoint cliques that are present in the egonet. Finally, a more advanced spectral clustering method is used to construct the edge descriptor sets associated with the node in question. We begin our discussion with the latter of these two processes, because it is more involved and provides a natural introduction to the former.

4.3.1.1 ICM-Matrices

In this section, we examine the spectral properties of idealized egonets formed by cliques that are only connected through a single (egocentric) vertex. While this situation may appear unrealistic, we explain in the next section how to extract such subgraphs from the original graph. Our present goal is simply to identify and extract the corresponding cliques; we propose a spectral approach to solve this problem.

Let us consider the sub-matrix of the network adjacency matrix that describes the local egonet. A trivial re-indexing of the nodes allows us to represent this submatrix as a block-diagonal matrix, where each block is a clique. The blocks do not overlap, but there is a row (and corresponding column) of ones to describe the connection of the egocentric node to all the cliques. We can assume that the row and corresponding column associated with the egocentric node are the first row and column, respectively. Instead of working directly with this matrix, we propose to make some slight modifications that boost the spectral approach. The modifications are as follows: we add self connections to all the nodes if not already present, and we scale all connections to the egocentric node by a small parameter, δ. We call these modified matrices *ideal community member matrices* for the sake of discussion, or *ICM-matrices* for short.

Let A be an ICM-matrix with m blocks (each corresponding to a clique) along its diagonal, where the size of the i^{th} block is $k_i \times k_i$. Without loss of generality, we assume the indices are arranged such that $k_j \leq k_i$ whenever $j > i$ so that larger indices correspond to smaller blocks. Also assume that the first index corresponds to the egocentric node. We represent the set of indices for the k^{th} block by V_k. With this notation in place, A is defined by Equation (4.4), and an example of the structure of such a matrix is given in Figure 4.3.

$$A(i,j) = \begin{cases} \delta, & \text{if } i = 1 \text{ or } j = 1, \\ 1, & \forall i, j \in V_k, \quad k = 1, ..., m, \\ 0, & \text{otherwise.} \end{cases} \tag{4.4}$$

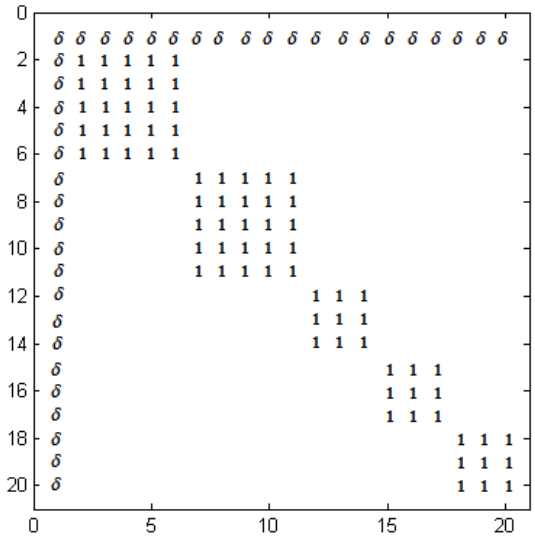


Figure 4.3: The non-zero values of an ICM-matrix for a node belonging to two cliques with 6 members, and three cliques of four members.

In order to identify the blocks associated with the cliques, we compute the eigenvectors of A that are associated with positive eigenvalues. Our focus is on the eigenvectors of these matrices with corresponding positive eigenvalues, as they will be the ones we will use for spectral clustering. Let $P = \{i|\lambda_i > 0\}$ be the positive eigenvalues, and let $\{\mathbf{x}_p|p \in P\}$ be the associated eigenvectors. Assume that the eigenvalues are ordered so that $\lambda_j \leq \lambda_i$ if $i > j$ (e.g. $\mathbf{x}_1$ is the dominant eigenvector). Let $\mathbf{x}_p(1)$ be the value of the first entry in such an eigenvector. Using a simple symmetry argument, one can show that each $\mathbf{x_p}$ is constant over each clique i; $\mathbf{x_p}(l \in V_i) = r_i$, for each vertex, l, in the set of vertices, V_i, in block i.

We now examine the properties of the entries of $\mathbf{x_p}$ by explicitly looking at the system of equations resulting from the constraint $\lambda_p\ \mathbf{x}_p = A\ \mathbf{x}_p$. For any $j \in V_i$, this constraint takes the following form,

$$\lambda_p\ \mathbf{x}_p(j \in V_i) = \delta\ \mathbf{x}_p(1) + \sum_{l\ \in\ V_i} \mathbf{x}_p(l), \tag{4.5}$$

and since $\mathbf{x_p}$ is constant and equal to r_i over each clique, we obtain

$$\lambda_p \; r_i = \delta \; \mathbf{x}_p(1) + k_i \; r_i, \tag{4.6}$$

or

$$(\lambda_p - k_i) r_i = \delta \; \mathbf{x}_p(1). \tag{4.7}$$

As Equation (4.7) holds for any row indices, i and j, this leads us to Equation (4.8),

$$r_i = \frac{\lambda_p - k_j}{\lambda_p - k_i} r_j. \tag{4.8}$$

The importance of Equation (4.8) is that it shows that $|r_i| > |r_j|$ if $|\lambda_p - k_i| < |\lambda_p - k_j|$. In other words, the closer the number of members in the block corresponding to the i^{th} clique is to λ_p, the larger the magnitude of r_i in $\mathbf{x}_p$ and the easier it will be to pull out members of distinct cliques via spectral clustering. Now note that by setting δ to a small value, the ICM-matrix is a small perturbation of a block diagonal matrix. As the positive eigenvalues of a block diagonal matrix correspond directly to the sizes of the blocks present within the matrix, this causes the eigenvalues of the ICM-matrix to be small perturbations of k_i. This implies that for each block of size k_i there exists an eigenvalue, λ, such that $\lambda \approx k_i$. For the results presented in this chapter, we use $\delta = 1/N_{ego}$, where N_{ego} is the number of nodes in the egonet.

For illustrative purposes, the first three dominant eigenvectors of the matrix depicted in Figure 4.3 are shown in Figure 4.4. Note that the components of the eigenvectors clearly reveal each clique. However, this property does not extend to eigenvectors without positive eigenvalues. Indeed, the nullspace vectors are quite noisy and are detrimental to use for spectral clustering of the egonet.

For the case of λ_1, we can also prove that $r_i \geq r_j$ when $j > i$. This follows from the fact that all of the entries of A are non-zero and the power method converges to the dominant eigenvector. As the power method converges regardless of the initial starting vector, we can take a vector with all non-negative entries as our initial guess and the power method iteration preserves this property.

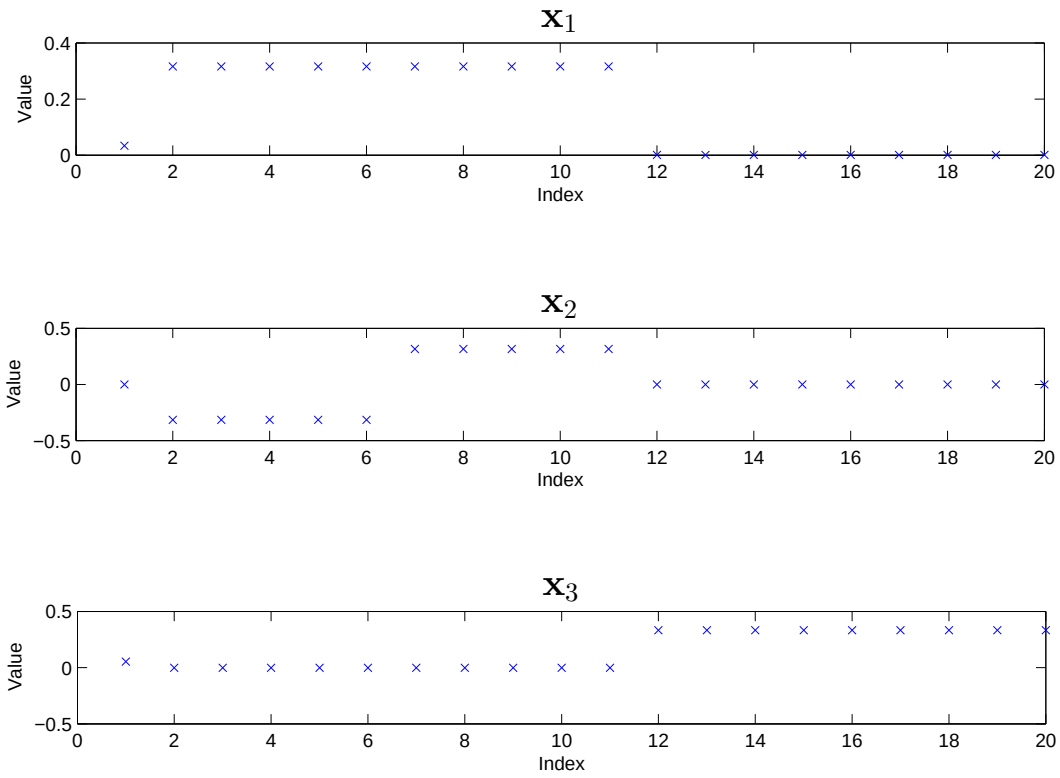


Figure 4.4: First three dominant eigenvectors for the ICM-matrix depicted in Figure 4.3, where $\delta = 1/N_{ego}$ and N_{ego} is the number of nodes involved in the egonet.

This implies that the magnitude of $\mathbf{x}_1(i)$ increases with the size of the clique in which the vertex i belongs to. This is important to the egonet sparsification algorithm discussed in the next section.

We are now in a position to define the edge descriptor sets associated with each node. Each edge descriptor set is comprised of densely connected subnetworks of the egonet. Formally, we define the densely connected subnetworks as follows. We begin by extracting the node's egonet from the network, and form its corresponding adjacency matrix. Using the methods described in the next section, we sparsify this adjacency matrix so that the remaining connections closely resemble the structure of an ICM-matrix. Next, we compute the eigenvectors associated with the largest positive eigenvalues of the sparsified matrix. We then use these eigenvectors to embed the egonet into a metric space, treating the value of each eigenvector at a given index as providing a

spacial ordinate for the corresponding vertex [11]. With each vertex having spacial coordinates, we then apply k-means clustering in order to find clusters of vertices. As each vertex is also connected to the egocentric node, each cluster of vertices represents a cluster of edges to use as a potential edge descriptor set. Finally, before accepting a cluster as an edge descriptor set, we additionally check to that the set of nodes involved has 90% or greater edge density between them to ensure they approximately form a clique. Although these edge descriptor sets will often be referred to as "cliques" for simplicity throughout this chapter, we only require them to be very densely connected rather than fully connected.

To determine both the number of eigenvectors to use for the spectral embedding and the number of clusters, we use the following set of heuristics. Presumably, the largest cliques a node belongs to are the most important ones to accurately capture for describing that node. As larger cliques correspond to larger eigenvalues in an ICM-matrix, we estimate the number of coordinates used for the spectral embedding and the number of clusters as the number of eigenvalues that are greater than one tenth of the largest eigenvalue. This allows us to recover all the largest cliques an egocentric node belongs to, while guarding against involving near nullspace vectors for the clustering process.

4.3.1.2 Perturbing Scaled ICM-Matrices with Random Matrices

The preceding sections described the eigenspace properties of ICM-matrices, where the members of cliques belonging to distinct communities only share the target node as their interconnection. However, such an ideal situation is highly unlikely to arise in practice. In this section, we briefly examine what happens when perturbations are added that introduce random connections between members of differing cliques.

Let v_k again be the set of indices of nodes belonging to the i^{th} clique. Define $G_{A(p)}$ to be the symmetric random matrix such that $G_{A(p)}(i,j) = G_{A(p)}(j,i) = 1$ with probability p whenever $i \in v_k$ and $j \notin v_k$, and $G_{A(p)}(i,j) = 0$ otherwise. Using the Gershgorin Circle Theorem, we have that the expected magnitude of the largest eigenvalue is bounded above by $\gamma = p\ (K_{N_c} - k_{N_c})$; this

is the probability for an intercommunity link multiplied by the number of possible links.

Recall that a scaled ICM-matrix has eigenvalues that are approximately given by $\lambda_p \approx k_i - 1$. This implies that when $\gamma << k_x - 1$ where k_x is the largest clique size we are interested in, then this will not be a strong perturbation and the results from the ideal cases will still apply for practical purposes. However, if an egocentric node belongs to many separate communities, then it will generally be the case that $(K_{N_c} - k_{N_c}) >> k_i$ for any i. In this case, the perturbation will be small when $p << \frac{k_x}{K_{N_c} - k_{N_c}}$. However, as K_{N_c} is the total number of nodes that the egocentric node is connected to and k_x is just the number of nodes in a particular clique size, such a limitation is impractically restrictive. This motivates developing a sparsification algorithm that can remove such connections with high accuracy.

4.3.2 Matrix Sparsification Algorithm

As adding random connections to ICM-matrices can be disastrous with respect to its eigenspace properties, here we develop an algorithm capable of removing such connections with high accuracy. The way we accomplish this is by only including connections in the given egocentric node's ICM-matrix if those connections comprise the largest clique(s) each of the involved nodes belong to within the egonet. As long as the random connections actually behave randomly in that they do not create new large cliques by chance, then only connections involved in the largest cliques each node belongs to will remain.

4.3.2.1 Sparsification Algorithm Outline

(1) For each node present in a given egocentric node's ICM-matrix that is not the egocentric node itself, which we call sub-nodes, extract the sub-egonet for that sub-node. The sub-egonet is the egonet for the sub-node restricted to the set of nodes and edges represented in the ICM-matrix under consideration.

(2) Scale the extracted sub-egonet matrix so that the connections between the given sub-node and the nodes it is attached to are weighted so that $\delta = \frac{1}{N}$, where N is the total number

of nodes in the sub-CM-matrix.

(3) Approximate the dominant eigenvector of this matrix using several iterations of the power method. Once found, determine the edges to keep from the given sub-node and the rest by only including those whose values in the dominant eigenvector are greater than β times the maximum value, where $0 < \beta \leq 1$. As mentioned at the end of Section 4.3.1.1, the nodes that satisfy this are nodes belonging to the larger cliques.

(4) Once this process has been carried out for all of the sub-nodes, symmeterize the matrix by removing any edges that became directed through the sparsification process. This removes most edges bridging cliques in distinct communities by removing the edge to the node causing the bridge, effectively removing its inclusion in the smaller of the two cliques.

(5) Repeat this process until the ICM-matrix no longer changes.

4.3.2.2 Numerical Verification of Sparsification Algorithm

If one starts with an ICM-matrix that has had random connections added of the form discussed in Section 4.3.1.2, then our findings show that the sparsification algorithm yields either an ICM-matrix or a small perturbation thereof for reasonable values of p. By "reasonable values" we mean that the random connections do not completely drown out the planted clique structure. Our initial tests were conducted on perturbed ICM-matrices with multiple cliques of sizes 6, 8, and 10 members each, similar to the one depicted in Figure 4.5, and the parameter β in the algorithm was set to 0.5.

Figure 4.6(a) is a typical example of when the algorithm successfully removes all of the random connections, and Figure 4.6(b) is a typical example of what we found happens when the algorithm "fails". The reason for the quotation marks is that if you examine Figure 4.6(b) closely, you will find that the connections which remain that seemingly should have been pruned stem from the random perturbation process producing cliques of approximately the same size as the planted cliques each given node was set to belong to. The algorithm is still performing as desired, namely

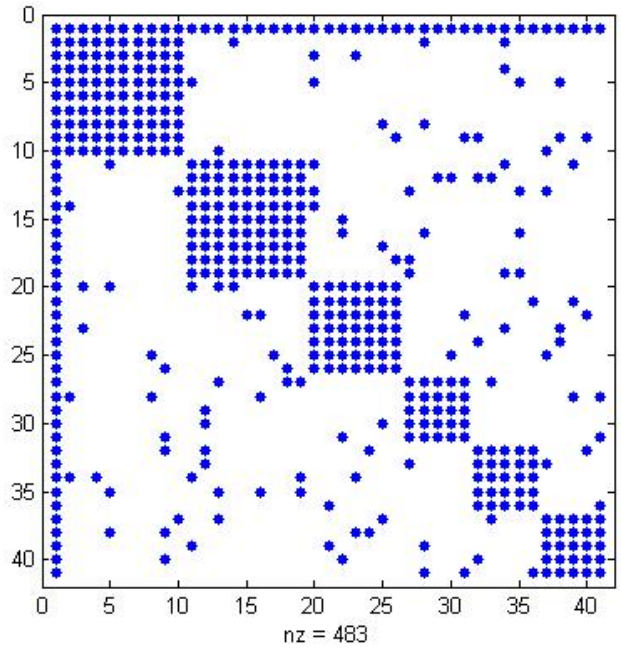


Figure 4.5: An example ICM-matrix with varying clique sizes and random connections added.

removing connections that are not part of a larger clique structure.

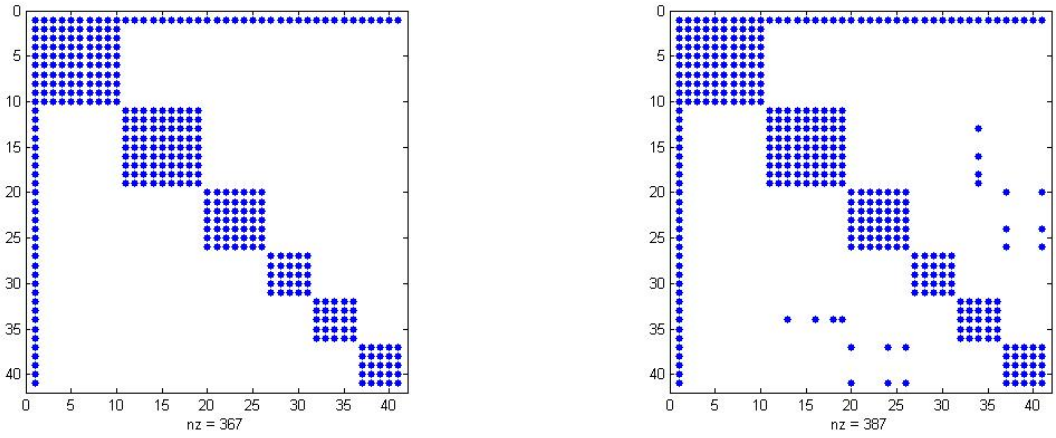
However, measuring the accuracy of the sparsification algorithm is not a straight forward process in such an ICM-matrix where the cliques substantially vary in size. The reason for this is that the random perturbations can easily cause a node planted in a small clique to become part of a larger one created by random connections to a relatively large clique; examples of this can be seen in Figure 4.6(b).

To sidestep these difficulties, we also tested the algorithm using fixed clique sizes of 10 and vary the expected number of outlinks from 0 to 30 (three times the number of inlinks). We measure the accuracy of the algorithm by the percentage of connections removed that came from the random perturbation process, meaning those connections that are between members of separate planted cliques.

Figure 4.10(a) shows the accuracy of the algorithm when there are five such cliques the target node belongs to, and the Figure 4.10(b) shows what happens when we double the total number of

cliques with keeping the clique size set to ten members each. Note that the algorithm performs above 90% accuracy well past the point where a given sub-node has more community outlinks than it does inlinks within the network corresponding to the pre-sparsified adjacency matrix in both cases. The reason the algorithm performs substantially better in the latter case, maintaining above 90% accuracy even when there are nearly three times the number of outlinks as inlinks, is because with more groups there is less of a chance for random connections to cause a node to belong to a larger clique than the one it was assigned to. This helps prevent new large cliques from forming as a result of the random perturbations, meaning it helps our metric for accuracy to remain valid.

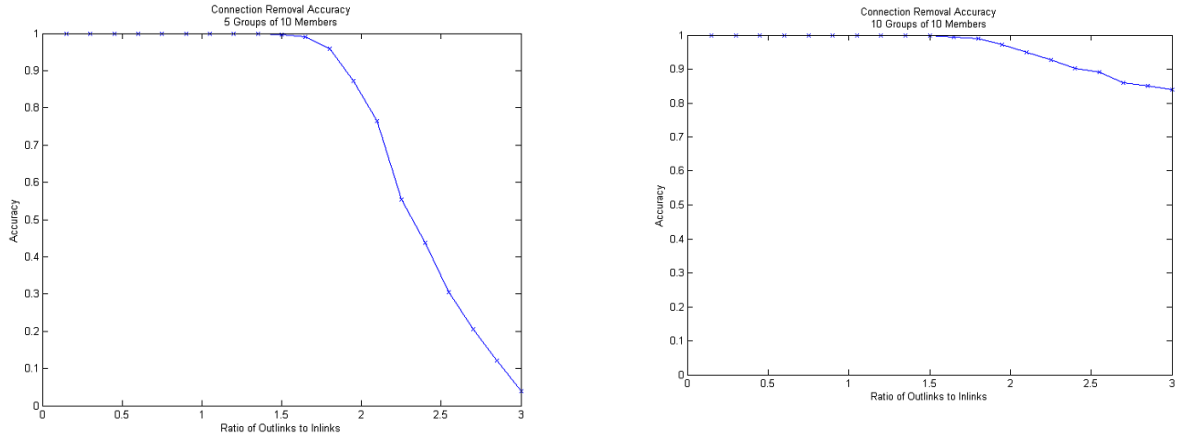
These tests demonstrate that our sparsification algorithm does an excellent job in removing connections between members of different cliques for the ICM-matrix of a given node. Because of this, applying the sparsification algorithm substantially improves the performance of k-means clustering for detecting the cliques a given node belongs to, as we will show in the next section.



(a) Successful Sparsification

(b) "Failed" Sparsification

Figure 4.6: a) An example of the resulting matrix after a successful sparsification. b) An example of the resulting matrix after a "failed" sparsification. However, if you carefully examine the remaining connections, you will find that they stem from the random perturbation process producing cliques of approximately the same size as the planted cliques each given node was set to belong to.



(a) Five Groups of Ten Members

(b) Ten Groups of Ten Members

Figure 4.7: The accuracy of the algorithm as a function of the expected value of the ratio of the inlinks between a given node and the members of its planted clique and the outlinks between a given node and members of other planted cliques. a) This test is carried out on an ICM-matrix composed from five disjoint cliques with ten members each. b) The same test carried out in (a), but with the number of cliques doubled.

4.3.3 Finding Cliques for a Given Node Using K-Means

Once the ICM-Matrix has been sparsified using the methods in the previous section, we find the edge descriptor sets to be used to describe the egocentric node via a spectral clustering method. Our method uses the components of the dominant eigenvectors of the ICM-Matrix with the connections to the target node removed to serve as ordinates for the nodes corresponding to those components. The sparsification process helps cause this matrix to be approximately block diagonal so the eigenvectors naturally partition the nodes into cliques for the reasons explained above.

4.3.3.1 Estimating the Number of Clusters for K-Means

As k-means requires some sort of initial estimate of the number of clusters to seed the algorithm, we need a way to estimate the total number of cliques we wish to try to find using it. To this end, we estimate the number of distinct cliques, N_c, we should use to describe the given

node as the number of eigenvalues above a threshold of the maximum eigenvalue. That is, if an eigenvalue is larger than $\alpha\ \lambda_{max}$ with $0 \leq \alpha < 1$, then it is a member of this set. We then take the total number of cliques to find as the number of elements in that set. The reason this heuristic works well is because the positive eigenvalues of a block diagonal matrix correspond directly to the clique sizes, so in the case where the sparsification process yields completely disjoint cliques (as in Figure 4.6(a)) this provides an automated means to find all of the cliques that are a fraction, α, of the largest clique size. As an example, we would need to set $\alpha \leq \frac{smallest\ clique\ size}{largest\ clique\ size}$ if we want to resolve the smallest blocks, or we could just set $\alpha = 0$ if we want to try to find all of the cliques. The reason for not just setting $\alpha = 0$ by default is that this would include 2-cliques comprised of just the given node and one other; groupings that for any practical purposes would likely be considered as anomolies or outliers. Moreover, the eigenvectors with zero eigenvalues for a block diagonal matrix are not useful for spectral clustering, as their values are essentially random, and these eigenvalues can easily be perturbed to be become positive when random connections are added in. To guard against these effects for the results presented in this thesis, we set $\alpha = 1/10$.

4.3.3.2 Verification Tests for Finding Number of Cliques

Let L_v be a vector containing the number of members in the cliques a target node is connected to and G_v be a vector such that there are $G_v(i)$ groups with $L_v(i)$ members. To tie this back to the notation used previously, note that the sum of the elements in G_v equates to N_c, the total number of cliques, and the inner product $\langle G_v, L_v \rangle$ is the total number of nodes connected to the given node, K_{N_c}.

By setting the probability for a random connection between nodes of differing cliques to be $p = \frac{Number\ of\ Members\ in\ Smallest\ Clique}{Total\ Number\ of\ Nodes}$, we can create a network with sparse enough connections so that the expected number of outlinks for a member of one of the smallest cliques is equal to the number of inlinks within the clique. Doing so prevents the members of the smallest prescribed clique from actually belonging to a larger clique through random connections during testing, meaning it prevents failures of how accuracy is being measured.

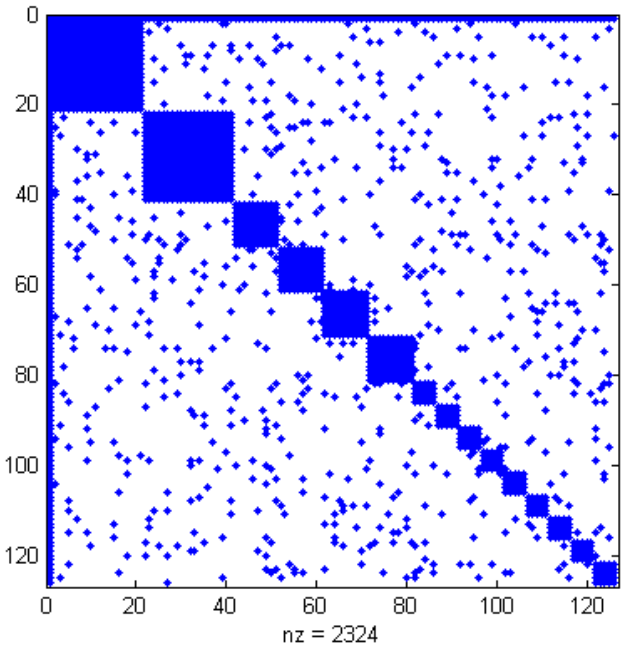


Figure 4.8: An example of a larger scale ICM-matrix with varying clique sizes and random connections of the type we used for clique identification tests.

The first set of tests we discuss deals with a network corresponding to an ICM-matrix with L_v=[20,10,5] and G_v=[2,4,9], meaning there are: two groups of twenty nodes, four groups of ten nodes, and nine groups of five nodes. The purpose of these tests is to examine the ability of the algorithm to identify the number of cliques on the fly and find them under conditions where the clique sizes and numbers vary in a power-law fashion and the between clique connections are sparse enough so as not to ruin the planted clique structure.

To measure the accuracy of our algorithm, we match each clique with the group assigned to the majority of its members and count the number of members in that group. We then say that the accuracy of that grouping is given by $\frac{clique\ members\ in\ group}{total\ members\ in\ clique}$. The overall accuracy is then taken to be the average accuracy over all the groupings, as given by Equation (4.9). This is essentially a set similarity measure of the found groupings of sets of nodes and the planted groupings of sets of nodes.

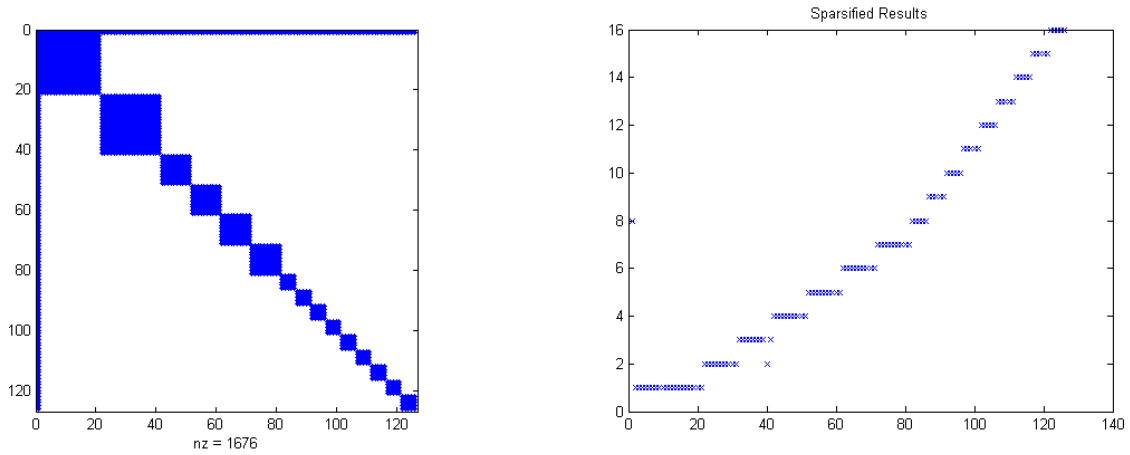
$$Clique\ Finding\ Accuracy = \frac{1}{N_c}\sum_{i=1}^{N_c}\frac{found\ clique\ members\ in\ i^{th}\ planted\ clique}{total\ members\ in\ i^{th}\ planted\ clique} \tag{4.9}$$

We compare finding the cliques using spectral clustering methods using the original matrix (our baseline) against the sparsified matrix. We also examine two different representations of eigenvector coordinates that we considered for implementing k-means. The first is just using the normalized eigenvectors (normalized so they have length 1 in the L^2 norm), and the second is scaling these eigenvectors by their eigenvalues before using them as ordinates. The reason for the latter is that larger cliques have larger corresponding eigenvalues, so multiplying their coordinates in the largest eigenvectors helps separate them from the other nodes. The intention behind this is that larger cliques intuitively serve as better descriptors for the given node, so it is of greater importance to represent them with accuracy than the smaller ones. Moreover, multiplying coordinates by eigenvalues guards against accidentally including too many eigenvectors in the spectral representation, as doing so causes nullspace vectors to be also used in describing nodes. The reason this is important is that the analysis done for the eigenvectors with positive eigenvalues does not carry over for those with zero eigenvalues, and the coordinates these vectors provide are essentially random. However, if we multiply the coordinates in these vectors by their eigenvalues (which are either zero or small perturbations away from zero), then we effectively remove them from the way we describe the nodes for spectral clustering. The results of the algorithms accuracy averaged over twenty tests under various settings for the network of the kind displayed in Figure 4.8 is presented in Table 4.1.

As Table 4.1 shows, sparsifying the matrix trivializes the problem in both eigenvector representations. Although the scaling has a very minor impact on the accuracy of the algorithm, its impact on the quality of the groupings is significant. Figure 4.10 gives characteristic examples of what groupings are found when using the non-sparsified matrix, where the group sizes have been doubled to make the impact on quality more pronounced. From this figure, we see that the found groupings essentially have the same accuracy, which one can roughly infer by noting that the total

Method	Accuracy
Unscaled Baseline	84.4%
Scaled Baseline	83.1 %
Unscaled Sparsified	97.3%
Scaled Sparsified	97.3%

Table 4.1: The results of the algorithms accuracy under various settings for ICM-matrices of the kind displayed in Figure 4.8. The perfect accuracy of the sparsified method is what we would expect based on the results presented in Figure 4.7.



(a) Sparsified Matrix

(b) Scaled Groupings

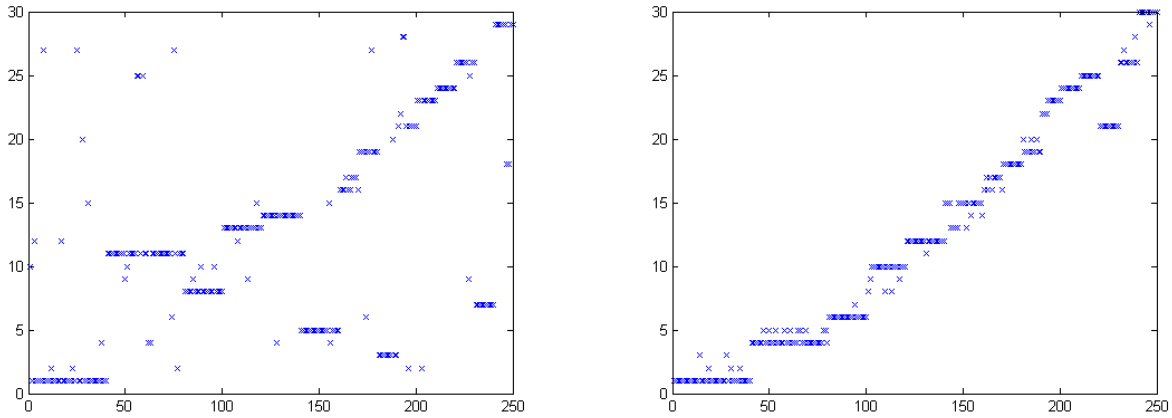
Figure 4.9: a) A characteristic example of the sparsified matrix returned when using scaled eigen-coordinates. Note that all of the random connections have been removed. b) A characteristic example of the groupings found by the algorithm when using the sparsified version of the ICM-matrix. The horizontal axis provides the node index, and the vertical axis indicates which cluster the node was binned in. Note that one of the groups (the second group of 20) has been artificially split into two. Although not ideal for clique detection purposes, splitting cliques has no impact on the performance of the algorithm.

number of groups in each case is about the same. However, when the scaled version incorrectly bins a node into a group separate from the majority of its fellow clique members, it does so in a way that creates a sensible subgroup, whereas the unscaled version distributes them in a random fashion.

4.3.3.3 Member Count and Link Density

Thus far we only have only described a spectral clustering method to cluster the nodes into sets, but this doesn't mean that the sets are guaranteed to be cliques. This is obvious from the groupings coming from the non-sparsified case in Figure 4.10, but the same problem occurs in the sparsifed case when the probability of between clique connections, p, destroys the planted clique structure. For example, if instead of setting $p = \frac{Number\ of\ Members\ in\ Smallest\ Clique}{Total\ Number\ of\ Nodes}$ for the ICM-matrix depicted in Figure 4.8 we set it to four times that amount, then we effectively drown out the planted clique structure for the smaller cliques with noise as can be seen in Figure 4.11(a). Because of this, many of the connections those nodes have will become directed and removed by the sparsification algorithm as shown in Figure 4.11(b). As a result, the groupings found by k-means will lump all of these dangling nodes into one super cluster, as depicted in Figure 4.12.

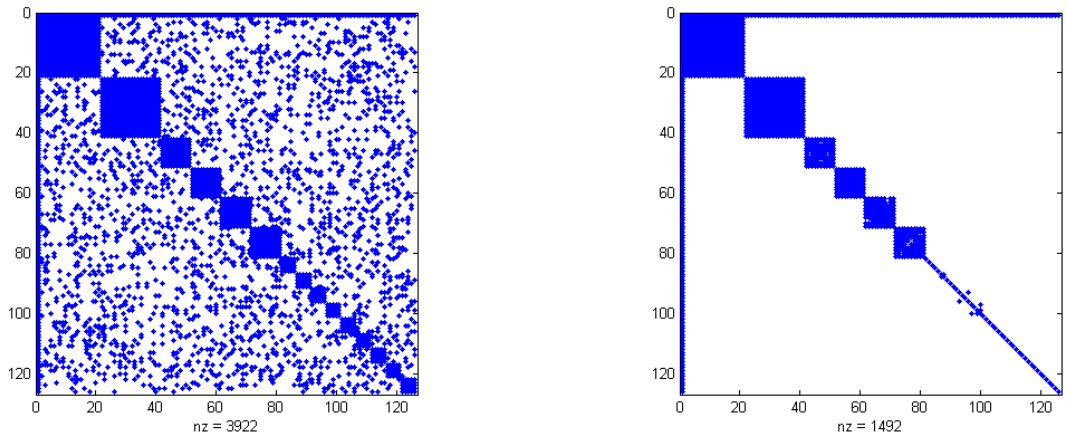
This highlights the importance of two parameters to consider when constructing approximate cliques to describe a given node. The first parameter is the number of members within a found



(a) Unscaled Groupings

(b) Scaled Groupings

Figure 4.10: a) The found cliques using normalized eigenvectors as the ordinates for k-means. b) The found cliques using eigenvectors scaled by their eigenvalues as the ordinates for k-means. Note that although the accuracy of the groupings in (a) and (b) are more or less the same, the quality of the groupings in (b) is higher in that incorrectly grouped nodes form sensible subgroups in (b) moreso than in (a).



(a) Original Matrix

(b) Matrix After Sparsification

Figure 4.11: a) The original ICM-matrix for p set to four times the threshold needed to resolve the smallest cliques. b) The result of applying the sparsification algorithm to the matrix.

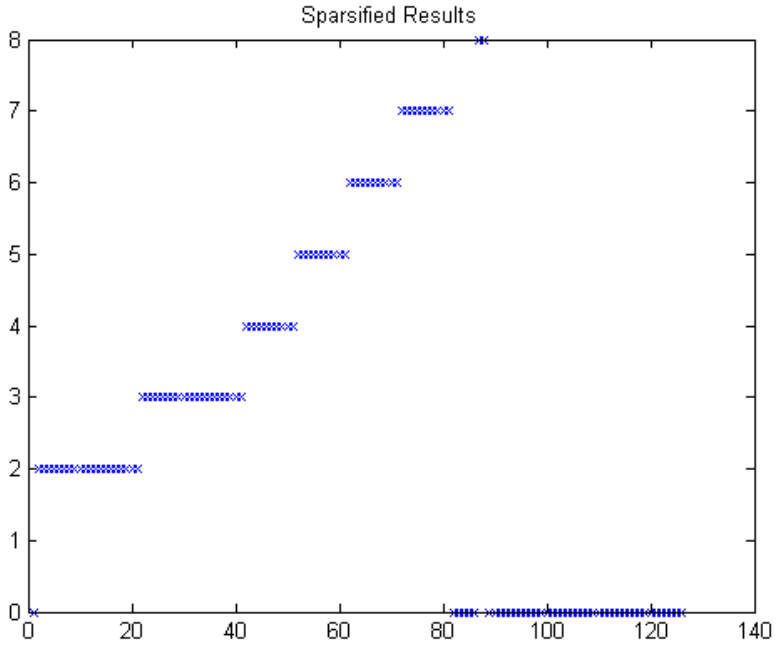


Figure 4.12: Example result of the groupings found by the algorithm when using the sparsified version of the ICM-matrix shown in Figure 4.11(b).

approximate clique. To this end, we only consider clusters as potential representatives of a given node if they have more members than $\lfloor \alpha \ \lambda_{max} \rfloor$. The reason for setting the required number of members to this is that $\alpha \ \lambda_{max}$ is the resolution power of our algorithm, as explained in Section 4.3.3.1. If the number of members in the grouping is below this, there is no reason to believe it is an accurate grouping. The second parameter is edge density. The edge density for a set of nodes forming a true clique is at its maximum possible value, so we incorporate a parameter that considers the ratio of the edge density of a found grouping to this maximal value as a means to gauge how close the group is to being a true clique. Setting this parameter to one would mean that only true cliques are taken into consideration, but this will inevitably throw out sets of edges that approximately form a clique and are useful for describing the given egocentric node. For the results presented in this thesis, the threshold edge density for approximate clique was set to 90%.

4.3.4 Link Density Based Community Expansion

As mentioned previously, instead of viewing communities as being comprised of clusters of nodes, we view them as being comprised of clusters of edges. Specifically, once we have reasonable estimates for the in-community cliques each node belongs to through the methods explained in the previous sections, we then cluster those cliques to form communities. The way we accomplish this is by starting with one of the approximate cliques present in the network as a community base, and successively add cliques to the community that minimize the decrease in link density. The community expansion ceases once the link density drops below a user supplied threshold, and the resulting set of edges is taken to comprise a community. Although this is not the most principled optimization approach from a mathematical standpoint, it is still sensible and also enhances the speed and parallelizability of the algorithm. Another feature we see as appealing at an intuitive level is that it preserves the idea of communities forming as a diffusive process of individual perspectives on the community structure of the network, similar to clique percolation or label passing.

Let G be a graph. Let S_i denote a partition of its edges such that $E(G) = \cup_{i=0}^{N} S_i$, and $S_i \cap S_j = \emptyset$ if $i \neq j$. Also, let $V(S_i)$ denote the set of vertices that are used to define the edges

belonging to S_i. A clique is a set of completely interconnected nodes, so if a clique is comprised of of n nodes, then it contains $\frac{1}{2}n(n-1)$ edges. We use $\mu(S_i)$ to represent the inlink density of the set of edges, S_i. Note that $|V(S_i)| = n$, so μ can be defined as

$$\mu(S_i) = \frac{|S_i|}{\frac{1}{2}|V(S_i)|(|V(S_i)|-1)}. \tag{4.10}$$

Note that if we start with a clique as a community base for S_i, then $\mu(S_i)$ is at its maximum possible value. For any set of edges we add to S_i, $\mu(S_i)$ can only decrease or stay the same, meaning we can make it a monotonically decreasing function as we successively add cliques to the set. Our method for community formation is based on the idea of satisfying $\mu(S_i) = D_i$ with $|V(S_i)|$ as large as possible, where D_i is a user supplied inlink density they believe the community should satisfy. We take a greedy algorithmic approach to this process, where a new approximate clique is added to the community set if it decreases μ by the minimal amount of any clique while still staying above the user supplied threshold.

4.3.5 Community Dectection Algorithm

Notation:

S: The set of edges that belong to the community being formed

s: The set of edges belonging to a possible approximate clique to add

Overview: $\mu(S)$ as defined in Equation (4.10) will be monotonically decreasing if we expand the community by adding new sets of edges to S that incur a minimal decrease. The community expansion stops once there are no more cliques that can be added that satisfy the user supplied cut-off edge density.

Community Detection Algorithm Outline

(1) Take the largest unclustered edge descriptor set remaining as the base for a community, S.

(2) Add to S the largest edge descriptor set, s, that satisfies

   (a) $\mu(S \cup s) > D$, the cut-off density constraint

   (b) $\mu(S \cup s)$ is maximal over possible choices of s. Note that the search need not be over all unclustered cliques, and can be restricted to searching over the cliques attached to nodes which are elements of $V(S)$ without any loss of accuracy.

(3) Continue to add new cliques to S until the cut-off density constraint can no longer be satisfied

(4) Repeat (1) - (3) until no edge descriptor set remains unclustered

(5) Add any unclustered node in the network to the community it shares the most edges with. Repeat this process until no unclustered nodes remain.

(6) Prune out the smaller communities, only keeping as many communities as needed to provide a cover for the graph.

User Supplied Parameters

α: This is a parameter controlling the number of cliques used to represent a given node. The options are to either input an integer or a fraction between zero and one. If a positive integer is supplied, the algorithm seeks to find that number of cliques for the given node. If a fraction is supplied, the algorithm attempts to search for cliques that are α times the size of the largest clique the node belongs to. For the results presented in this thesis, this parameter is held fixed to $\alpha = 0.1$.

D_i: This sets the cut-off density we require a set of edges to satisfy in order to consider it a community.

Hardcoded Parameters

Threshold density for what to consider as an approximate clique: The k-means clustering just clusters nodes that have similar eigenvector coordinates. Although members of the same cliques tend to have similar coordinates for the reasons explained previously, so do all of the nodes that do belong to any particular clique. Because of this, we throw away any clusters that do not satisfy an edge density threshold. The default used for the results in this thesis was set to 90%.

How many nodes a clique must have in order to serve as a community base: Since it is undesirable to start with "cliques" comprised of dangling nodes as community bases, the algorithm requires a threshold number of members an approximate clique must have to fulfill such a role. For the results presented in this thesis, this threshold is set to three.

How many nodes a clique must have in order to serve as a descriptor for a given node: To avoid approximate cliques comprised of too few nodes to meaningfully describe a given node attached to them, we also hardcode the minimum number of members approximate cliques can have to three.

4.4 Community Detection Algorithm Verification

To test our community detection algorithm, we considered four benchmark tests. Two of these are graphs from real world social networks, one the famous Zachary Karate Club network and the other the high school friendship network used in Xie et al. [78]. The other two tests are on sets of synthetically generated networks, coming from the planted l-partition and LFR benchmark tests.

4.4.1 Planted L-Partition Benchmark Tests

The initial set of tests we implemented our algorithm on are what are called planted l-partition tests. These are standard benchmark tests [29] that create networks with planted partitions of nodes so that the nodes in a given set have a higher connection probability than do the nodes in differing sets, and the total number of such partitions is denoted by l. These partitions are then taken as representing communities. The test involves fixing the expected degree of a node, and increasing the expected proportion of those links that are outlinks to other communities. Not only does this make the boundary between communities less well defined in that there are more links between communities, but it also makes communities less well defined by decreasing their inlink density.

The test was originally described so that accuracy was measured on the node level, where a node was correctly classified based on whether or not it fell in the community it was planted in. However, the goal of community detection algorithms in general is obviously to *detect communities* (sets of nodes or edges), not to *classify individual nodes*. These are distinct concepts and it is perplexing how this metric to gauge the quality of community detection algorithms for this test has persisted for so long. To put things as simply as possible, classifying a node is the process of determining whether or not it was placed in the appropriate box; community detection is the process of finding the boxes. Assuming that the communities are defined by the set of nodes they contain, these two ideas are equivalent when the community structure a given algorithm is trying to capture assumes that nodes can belong to one and only one community. This is what

the overwhelming majority of community detection algorithms at that time were doing, so it made no practical difference when the test was first developed. However, choosing to use how a node is classified as the metric to gauge the performance of a community detection algorithm that does not make this simplifying assumption is obviously nonsensical; there is no longer a one to one correspondence between sets of correctly classified nodes and correctly identified communities. In light of this, we instead use F-scores to measure the quality of the results produced by our algorithm, defined by Equation (4.11) below.

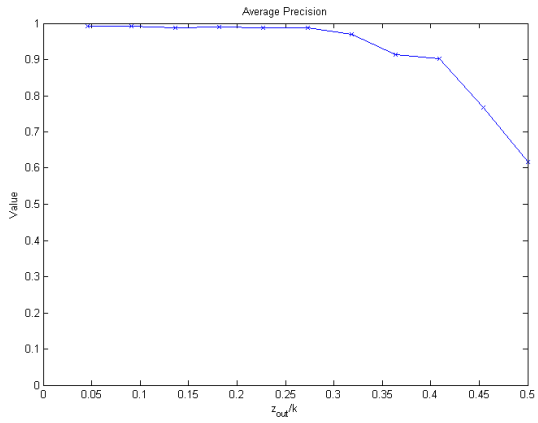
$$F = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{4.11}$$

The precision value for a given found community is the size of the overlap between the found community with the planted community divided by the size of that found community. The recall value for a given found community is the size of the overlap between the found community with the planted community divided by the size of the planted community. The precision and recall values reported are taken as the average precision and recall values when the planted community is paired with the detected community with the highest precision value. The benefit of using this metric to assess our algorithm is that the precision scores reflect the quality of our choice for edge descriptor sets, and the recall scores reflect the quality of our choices for community formation and desired network level properties.

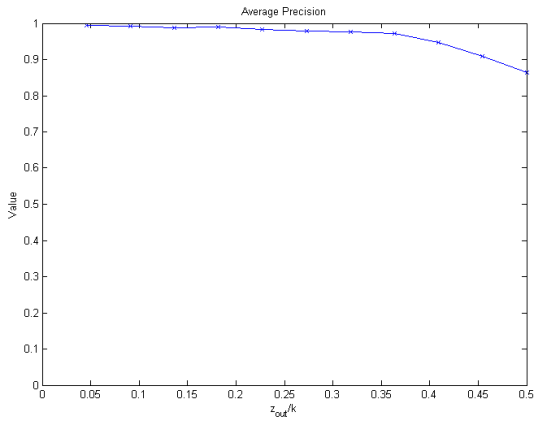
4.4.2 Planted L-Partition Results

We ran several versions of the planted l-partition test: four groups of 32 members, eight groups of 32 members, four groups of 64 members, and eight groups of 64 members. For all of these tests, the expected degree per node is set to be equal to half of the total number of members in each planted community. The first test is the most standard, and allows one to roughly compare our algorithm against a host of others [22]. The remaining tests demonstrate how the algorithm's performance improves when there are more communities and/or larger communities. The results

for precision, recall, F-score, and ratio of communities found are presented in the following figures, where each data point is the averaged results of individual twenty tests.

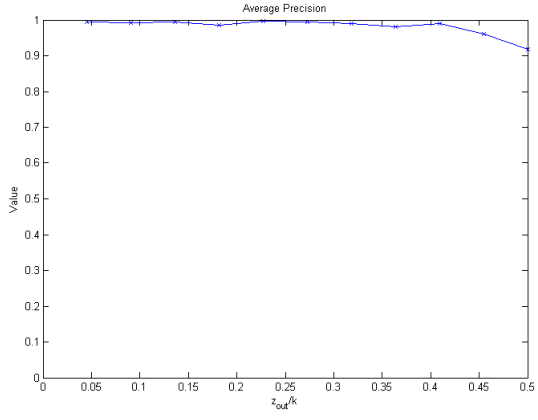


(a) 4 Groups 32 Members: Precision

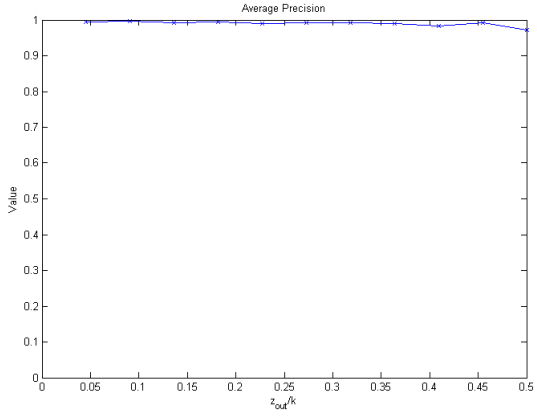


(b) 8 Groups 32 Members: Precision

Figure 4.13: The precision of the communities found by our algorithm for the planted l-partition tests involving groups with 32 members.

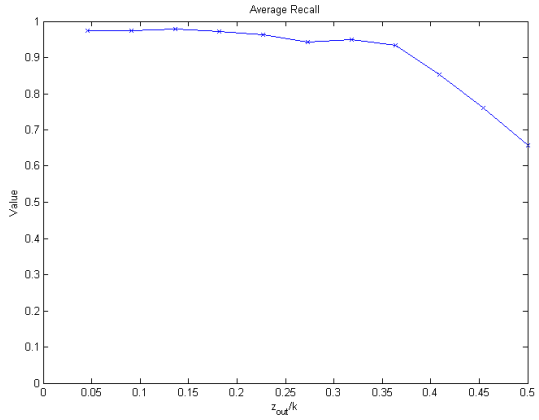


(a) 4 Groups 64 Members: Precision

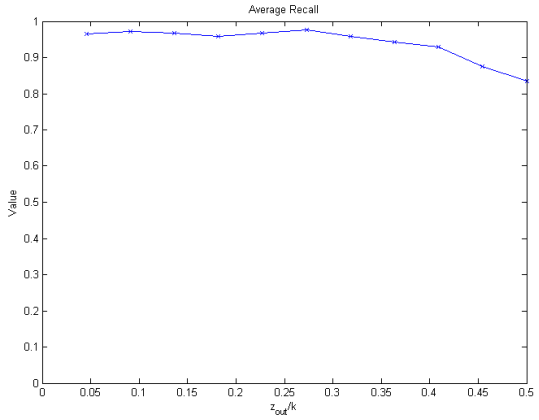


(b) 8 Groups 64 Members: Precision

Figure 4.14: The precision of the communities found by our algorithm for the planted l-partition tests involving groups with 64 members.

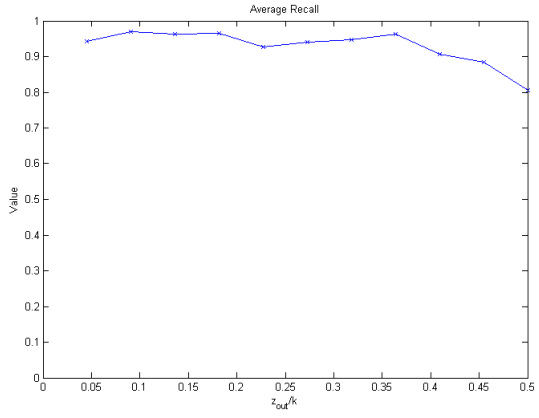


(a) 4 Groups 32 Members: Recall

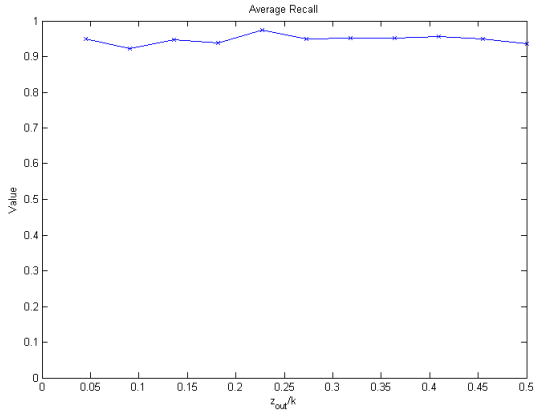


(b) 8 Groups 32 Members: Recall

Figure 4.15: The recall of the communities found by our algorithm for the planted l-partition tests involving groups with 32 members, coming from the recall of the community with the highest precision value.

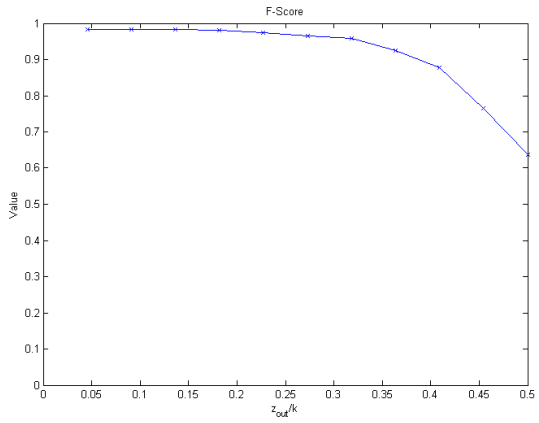


(a) 4 Groups 64 Members: Recall

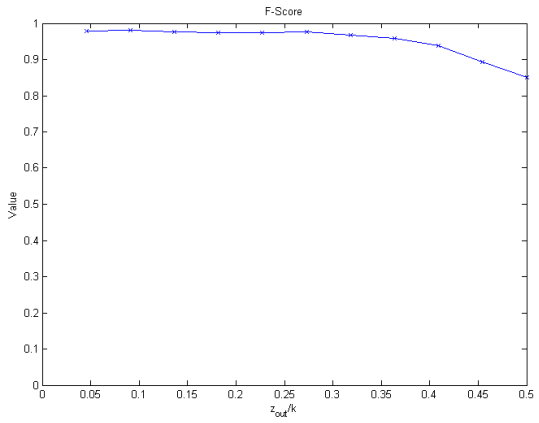


(b) 8 Groups 64 Members: Recall

Figure 4.16: The recall of the communities found by our algorithm for the planted l-partition tests involving groups with 64 members, coming from the recall of the community with the highest precision value.

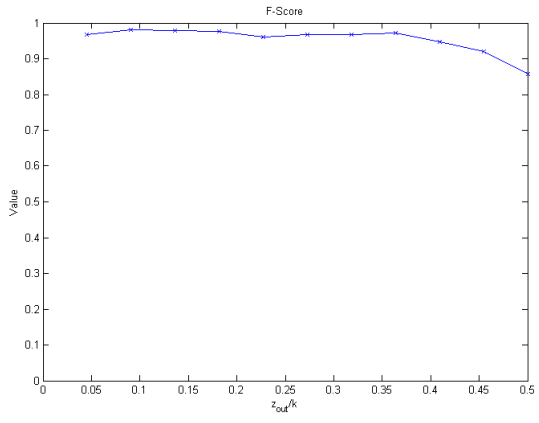


(a) 4 Groups 32 Members: F-score

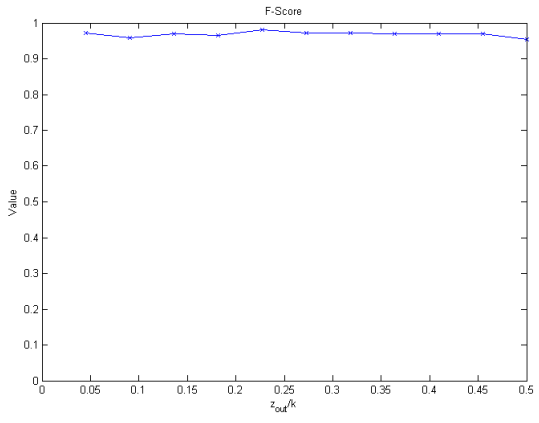


(b) 8 Groups 32 Members: F-score

Figure 4.17: The F-scores of the communities found by our algorithm for the planted l-partition tests involving groups with 32 members.



(a) 4 Groups 64 Members: F-score



(b) 8 Groups 64 Members: F-score

Figure 4.18: The F-scores of the communities found by our algorithm for the planted l-partition tests involving groups with 64 members.

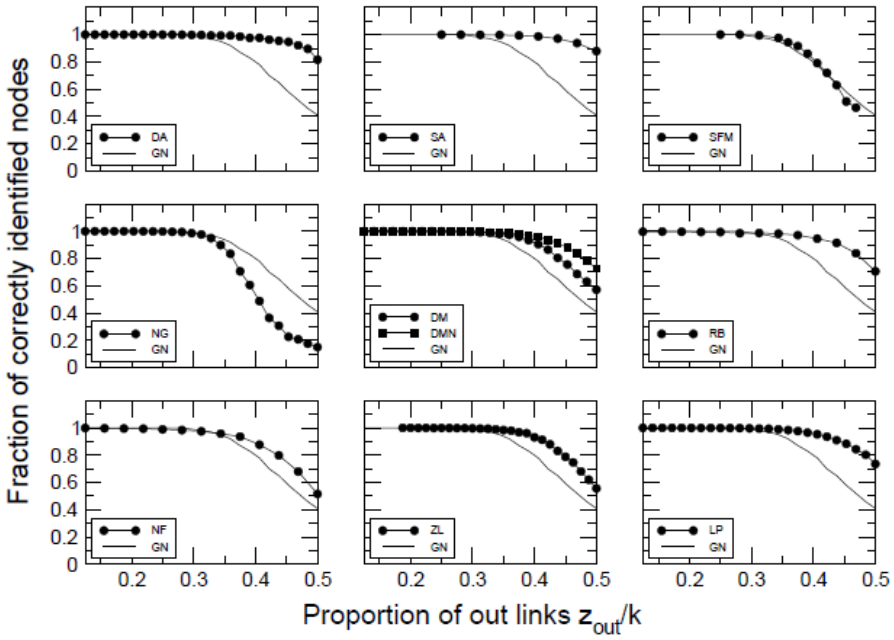


Figure 4.19: The accuracy of a host of algorithms presented in "Comparing community structure identification" [22] for the planted l-partition test with four groups of 32 members. The x-axis is the proportion of connections to outside communities and the y-axis is the fraction of nodes correctly identified by the method. Note that these methods assume that nodes only belong to one community and that the planted community is the "correct" one, so they are measuring correct identification as binning each node with the community it was planted to be a part of.

The results for the four group case shows in Figure 4.13(a) indicate that our algorithm performs about on par with the others demonstrated in Figure 4.19, which is taken from [22]. It needs to be mentioned that it is not a direct comparison as they are plots of slightly different quantities; the penalty for including a node in the wrong community is higher for the results in Figure 4.19 because it counts against both the accuracy of community the node was planted in as well as the accuracy of the community it was grouped with. The reasons why we are not measuring accuracy the same way is discussed at the beginning of the previous subsection.

4.4.3 Zachary Karate Club

Zachary's karate club network is a small social network comprised of the interactions amongst members of a university karate club studied by sociologist Wayne Zachary in the 1970's [79]. During

the period of study, a political issue arose regarding the club's fees which eventually caused the club to fissure into two clubs. The social interactions of the clubs' members outside of the official meetings were examined, and edges between members indicate that they interacted socially outside of the club setting. The ground truth for this test is taken to be which specific club the members joined after the fissure.

This example illustrates a fundamental and intrinsic difficulty with the community detection problem: the definition of a community is problem dependent, and one can only design algorithms that are optimal for certain classes of communities. The communities on this network are defined in terms of who leads them, where the leaders can easily be identified by the two nodes with substantially higher degrees than the average of those they share connections with. This suggests that a node's perspective on community should be defined by the leader(s) it is connected to, and the community scale features is defined by the perspective of its leader node. If one were earnestly interested in solving community detection problems of this type, a very simple approach would be to take the edges involving the two nodes of highest degree as edge descriptor sets, agglomerate these based on which leader node is involved to form communities, and then assign any unclustered nodes to the community they have the most links to. We have implemented this idea, and our algorithm yields a perfect recall value for each group, with F-scores both being over 0.94 for the given gold standard groupings.

Although this type of community structure is not at all what is intended for our algorithm to detect, it is a standard enough test to warrant seeing how it performs nonetheless. In order to apply our algorithm to this network, we first need to get an initial estimate of what the community density should be. To this end, we examine the edge density of the egonets for each node to get a local understanding of the average edge density of the network. Finding that the average egonet link density is 78.2%, we then set the community density to 3/4 of that in order to hold the communities to looser standards. This results in the three clusters of nodes given below, with the precision, recall, and F-scores for these groups are presented in Table 4.2.

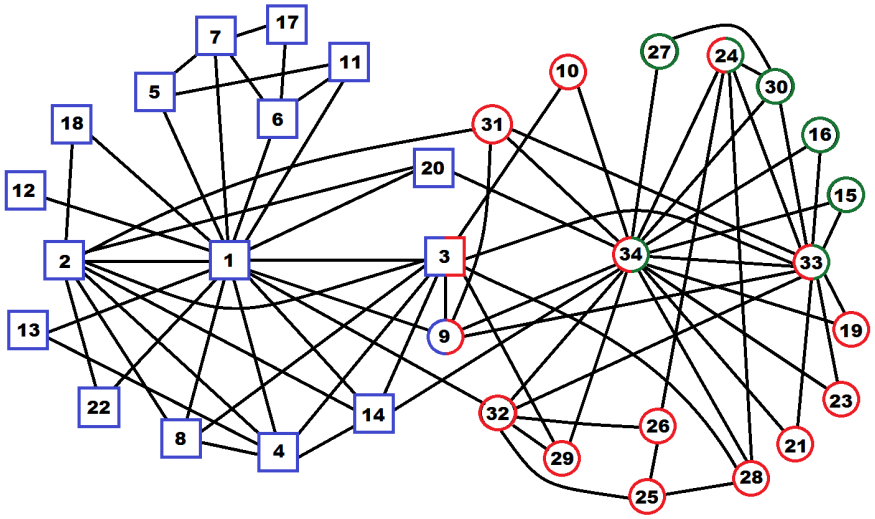


Figure 4.20: The Zachary karate club network. The gold standard grouping for a node is given by its shape, and its found grouping is given by its color(s).

Group 1: 1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 13, 14, 17, 18, 20, 22

Group 2: 3, 9, 10, 19, 21, 23, 24, 25, 26, 28, 29, 31, 32, 33, 34

Group 3: 15, 16, 24, 27, 30, 33, 34

As we can see, the communities produced by the algorithm cause the gold standard grouping, denoted by circles, to be split into two groups. The reason for this splitting is that the network is mainly composed of two subtrees (one for each leader), and therefore the density of connection within each subtree remains low. Our approach, which assumes a more "egalitarian" community structure, is no longer optimal when the network is organized in such a strongly hierarchical way. Splitting the gold standard group allows these new groups to have a higher edge densities of 34% and 63% whereas the community given as the gold standard only has an edge density of 27%. As we can see, although the division is not desired for this particular gold standard grouping, it is still a sensible one with respect to the notion of community our algorithm is designed to capture.

	Precision	Recall	F-Score
Group 1	0.94	1.0	0.97
Group 2	0.93	0.78	0.85
Group 3	1.0	0.39	0.56

Table 4.2: The precision, recall, and F-scores for the detected communities on the karate club network.

4.4.4 High School Friendship Network

The second real world network we analyze is a friendship network of students attending the same high school. This is a network from a project funded by the National Institute of Child Health and Human Development, where the links between students comes from self-reported connections. The dataset is part of the National Longitudinal Study of Adolescent to Adult Health [1] .

The true community partitioning of the network is taken as the grades (7 through 12) the students belong to, meaning there are six communities to be detected. Although the ground truth is taken as six communities, it is understood that the friendship connections for grade 9 demonstrate that the grade can be split into two distinct subgroups with one group composed of black students and the other white students, as can be inferred from Figure 4.21. This is one of the real world networks analyzed in a recent overview of the state of the art for overlapping community detection by Xie et al. [78], and thus allows for a comparison between our algorithm and those discussed in that paper.

For this test, we are going to also use an extended notion of normalized mutual information (NMI) [44, 43] in addition to F-scores to measure the algorithm's performance. The reason for this is that the former is the metric used in the aforementioned overview paper [78] that included this particular test. Although the extended notion of normalized mutual information slightly differs from what is standardly called NMI, we refer to this extended version as just NMI in this chapter for the sake of simplicity. The technical details of how to calculate the NMI for two sets of community labels is explained in the appendix of Ref. [44], but the basic idea underlying NMI is to treat community

[1] http://www.cpc.unc.edu/projects/addhealth/

labels as values of a random variable. Doing so allows one to compare how much the "probability distribution" of a gold standard set of labels differs from the "probability distribution" of labels found by an algorithm, using similarity measure methods meaningful to probability distributions. When communities are allowed to overlap, each node can be represented by a vector whose entries are 1 if the node belongs to the community represented by that index or 0 if the node does not belong to the community represented by that index. The i^{th} entry of this vector can then be thought of as a random variable V_i such that $P(V_i = 1) = n_i/N$ and $P(V_i = 0) = 1 - P(V_i = 1)$, where n_i is the number of members in the i^{th} community and N is the total number of nodes. This is how the probability distributions of the "multivariate random variables" described by each set of labels are defined. These variables are then compared using Equation (4.12) below, where X and Y can denote any pair of random variables.

$$NMI(X,Y) = \frac{2\ I(X,Y)}{H(X) + H(Y)} \tag{4.12}$$

Here, I gives the mutual information between two random variables, and H gives the entropy of a random variable. These are defined by Equations (4.13) and (4.14) below.

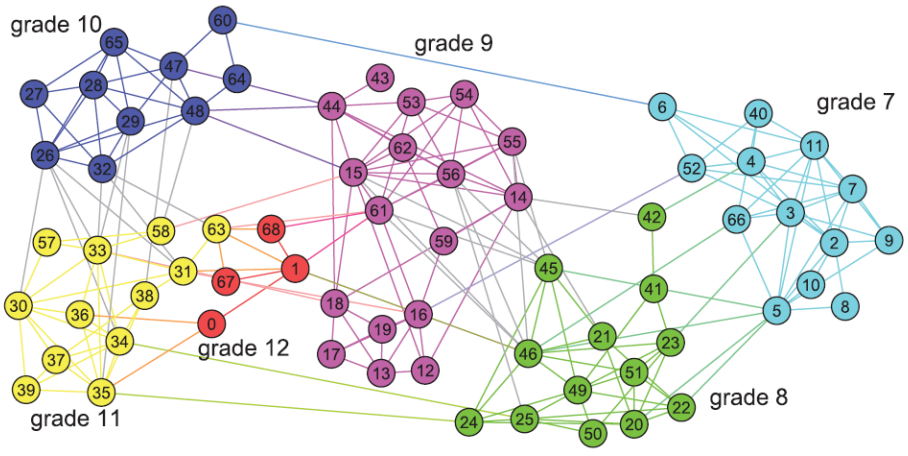


Figure 4.21: The high school friendship network as presented in Xie et al. 2013 [78]. The ground truth for this network is reflected by the color coding of the nodes, and roughly corresponds to the grade each student is in.

$$I(X,Y) = \sum_x \sum_y P(x,y) \ \log(\frac{P(x,y)}{P(x)P(y)}), \tag{4.13}$$

$$H(X) = -\sum_x P(x) \ \log(P(x)), \tag{4.14}$$

where x and y denote the values that X and Y can take.

Our approach to this network is the same as the karate club network discussed in the previous section. We estimate the desired community density by examining the average local edge density coming from each node's egonet, and set the community link density to 3/4 of that so as to hold communities to looser edge density standards. For this network, the average egonet link density is found to be 67.0%, so the community density threshold is set to 50.3%. We then take any nodes that remain unclustered after the community formation process, and assign them to the community they have the most links to.

Results

Precision: 0.79

Recall: 0.82

F-score: 0.80

Number of Communities: 8

NMI: 0.52

Overlapping Nodes: 5, 6, 16, 18, 21, 28, 29, 30, 31, 33, 34, 35, 44, 45, 46, 48, 52, 61, 65, 67

(20 overlapping nodes total)

Also, the precision, recall, and F-scores for each individual group detected by the algorithm is displayed in Figure 4.22.

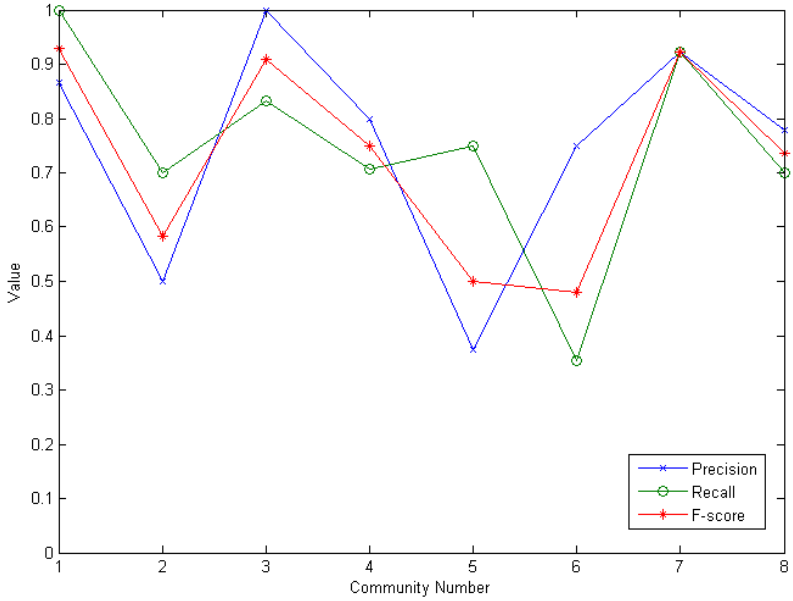


Figure 4.22: The precision, recall, and F-score values for each group detected by the algorithm, where each group is matched with the gold standard community with the highest precision value.

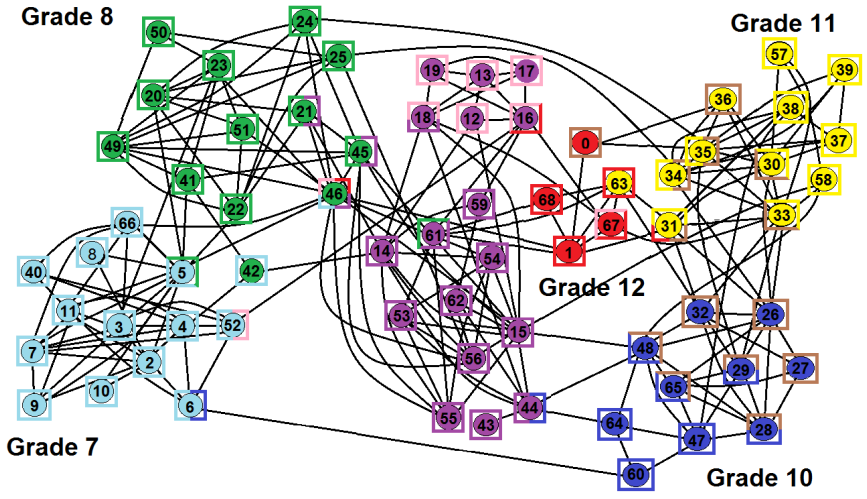


Figure 4.23: The ground truth for this network is reflected by the color coding of the nodes, and the found grouping for each node is reflected by the color(s) of the square surrounding it.

It can be verified from the graph in Figure 4.21 that each of the nodes in the overlap shares connections to multiple grades, with two exceptions. The exceptions are node 28 and node 65, both of which only have connections to other nodes in grade 10. The reason the algorithm claims these nodes belong to two distinct communities comes from their connections to the "community" comprised of the sets of nodes in grades 10 and 11 that are densely connected; these can be seen from the fairly large set of gray colored edges connecting these two grades in Figure 4.21. The results of other algorithms on this network as presented in Xie et al. 2013 [78] is given in Figure 4.24.

As can be seen from Figure 4.24, our algorithm performs about average with regards to the NMI's of the groupings produced by other algorithms. Despite its mediocre NMI score, the groupings found by the algorithm are sensible ones, as can be verified in Figure 4.23. It accurately detects the sub-division in grade 9, as well as the set of nodes falling in grades 9 and 10 that are densely connected to one another. Additionally, if one examines the nodes which are "incorrectly" labeled (e.g. 0, 42, and 63), it is readily apparent that the gold standard groupings do not accurately represent the nature of how these nodes are connected to the network.

Algorithm	Num. of communities	Overlapping nodes	NMI
CFinder	2	{12, 18}	0.1679
CIS	9	total 34	0.7495
COPRA	6	total 14	0.7966
EAGLE	4	{18}	0.4962
Game	10	total 14	0.4673
GCE	6	{0, 21, 45, 46, 61}	0.8333
iLCD	7	{5, 21, 26, 29, 31, 32, 33, 46, 61}	0.3713
LFM	7	{0, 45}	0.8134
Link	20	total 31	0.3155
MOSES	10	total 18	0.5037
NMF	7	{0, 12, 18, 45}	0.643
OSLOM	11	{45, 46}	0.4315
SLPA	6	{1, 42, 45, 59}	0.6788
UEOC	7	{0, 12, 18, 26, 29, 45}	0.8148

Figure 4.24: The results of a number of overlapping community detection algorithms applied to the high school friendship network, as presented in Xie et al. 2013 [78]. The average NMI for this set of algorithms is 0.59.

It is worth mentioning that with this network, as with the karate club network, the connections between nodes are defined by one criteria and the gold standard communities are defined by a completely different criteria. For this network, connections are determined by friendships and communities are determined by what grade people fall in. This is asking community detection algorithms to use the network structure to find communities that are not defined by the network structure, but are believed to be able to be inferred from it. A simple example of why this is misleading can be seen from examining the connections of the four students in grade 12. None of these four nodes forms a clique structure beyond a dyad (a pair of nodes connected by an edge) with any of the other nodes in the same grade, but they do form such larger cliques with nodes in other grades. In particular, note that node 0 only shares one edge with the nodes comprising grade 12, but two edges with grade 11. In light of these facts, we do not see a sensible way to argue that the network's gold standard community for grade 12 is actually a community in any meaningful sense. Although this is just one particular example, it is not difficult to peel through the network to find other peculiarities in gold standard community assignment (e.g. nodes 42 and 63), but such peculiarities are inevitable when creating gold standard communities by a practices that are completely blind to the network it is supposed to be defining communities on.

4.4.5 LFR Benchmark Tests

The final set of tests we consider are the LFR benchmark tests [42] which are designed to construct synthetic networks with built in community structures. The test is a variation of the planted l-partition model, where nodes are no longer required to all have the same expected degree and communities can come in varying sizes. Although more general versions of the test exist in which edge weights and directions are also taken into account, we only focus on the extended version of the test that also takes node membership to multiple communities into account. Nodes belonging to multiple communities are referred to as *overlapping nodes*, where the total number of overlapping nodes in a given network is referred to as O_n and the number of communities an overlapping node belongs to as O_m.

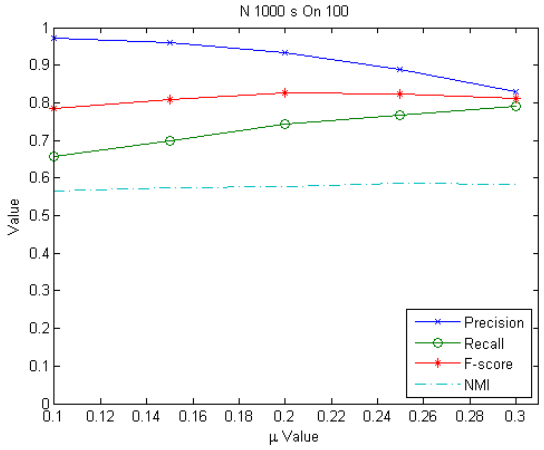
For each parameter set generated via LFR, we generated 20 network instantiations. Following the sets of parameters considered in the overlapping community detection overview by Xie et al. [78], we fixed several of the parameters throughout the tests. Node degree and community sizes are respectively drawn from power law distributions with $\tau_1 = 2$ and $\tau_2 = 1$, the average degree per node is set to $k_{ave} = 10$, and the maximum degree a node can have is set to $k_{max} = 50$. The remaining parameters were varied throughout the tests. we used networks with sizes $N \in \{1000, 5000\}$, and community sizes in both a small range $s = (10, 50)$ and a large range $b = (20, 100)$.

The first subset of tests we examine holds $O_m = 2$, and varies the value of μ, representing the fraction of links through which a node connects to members of other communities, from 0.1 to 0.3 in increments of 0.05. All of the combinations of parameter values for N and community sizes were examined, along with setting O_n to either 10% or 50% of the nodes in the network. For this set of tests, we examine the average precision, recall, F-score, and NMI for each set of communities returned by the algorithm compared against those planted by the test. The cut off density for community expansion is set to $(1 - \mu)$ multiplied by the average egonet density of the graph.

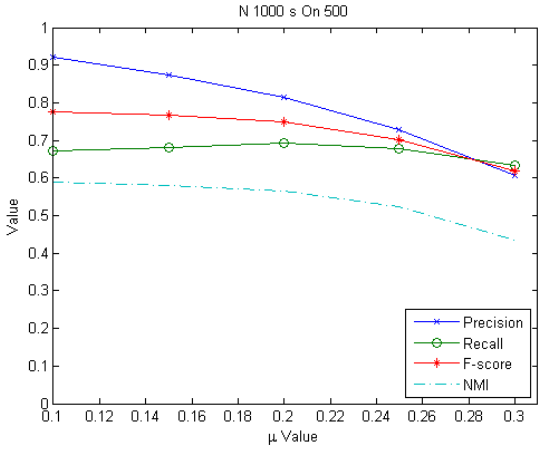
As we can see from Figures 4.25 through 4.28, the trends noted from the planted l-partition tests carry over to this case as well. The algorithm's performance improves with more and or larger communities corresponding to higher N values and community size ranges respectively. Somewhat surprisingly, the algorithm's performance also tends to increase with increasing values of the mixing parameter for the low overlap cases, where On = 10% of the total nodes. The cause of this is that increasing the value of the mixing parameter helps with recall scores by decreasing the edge density threshold for communities more than it hurts accuracy scores through developing cliques in communities other than the one a node was originally assigned to. However, for the high overlap case, increasing the value of the mixing parameter tends to have only minor effects on the recall scores while impairing the accuracy scores.

The second subset of tests we examine instead holds μ fixed at 0.3, and varies O_m from 2 to 8. The results of this test for networks with 5000 nodes, with 10% of the nodes being planted in two communities, are presented in Figure 4.29. From these we can see that the algorithm performs

well with respect to the precision of the communities detected, but that it has a tendency to split up the planted communities into smaller ones, hurting its recall scores.

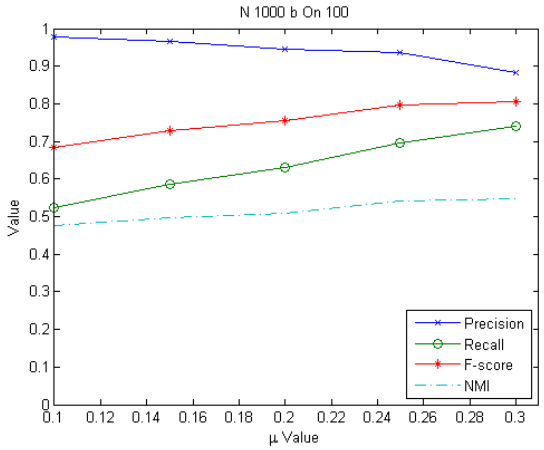


(a) N 1000s On 10%

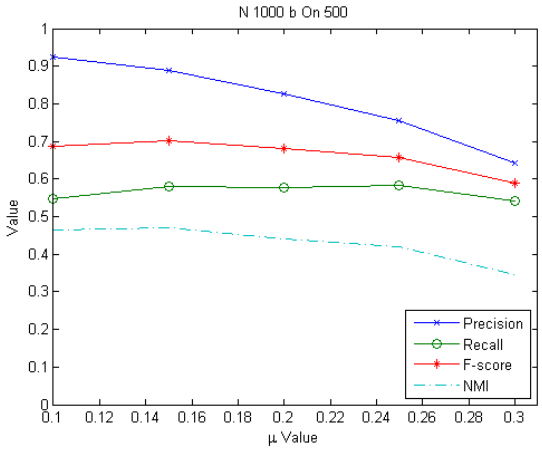


(b) N 1000s On 50%

Figure 4.25: The precision, recall, F-score, and NMI for the LFR tests varying μ for a network with 1000 nodes and using the smaller community size range. a) On = 10% of the total nodes. b) On = 50% of the total nodes.

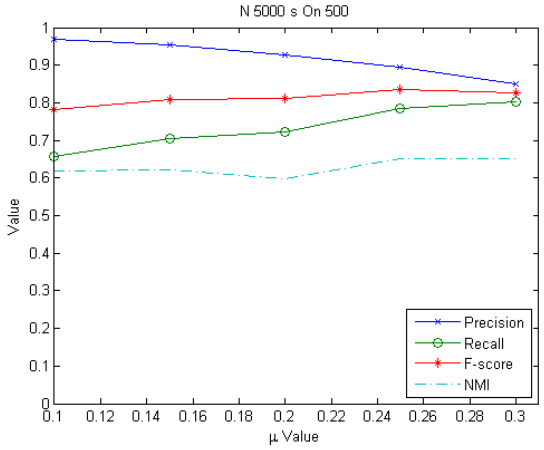


(a) N 1000b On 10%

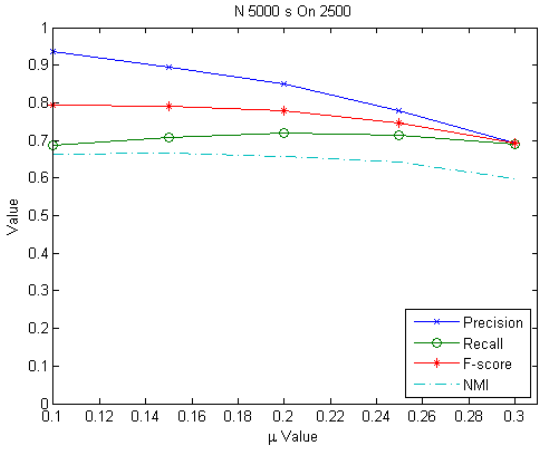


(b) N 1000b On 50%

Figure 4.26: The precision, recall, F-score, and NMI for the LFR tests varying μ for a network with 1000 nodes and using the larger community size range. a) On = 10% of the total nodes. b) On = 50% of the total nodes.

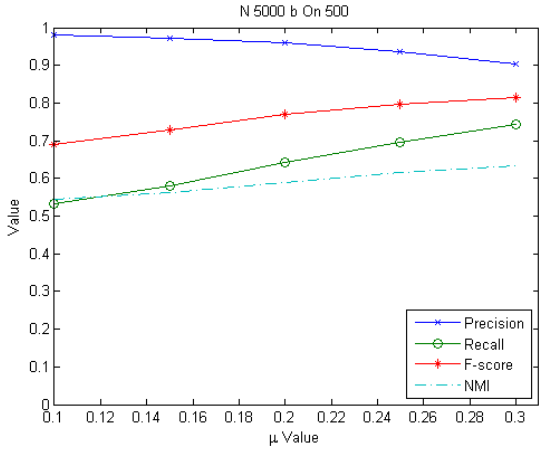


(a) N 5000s On 10%

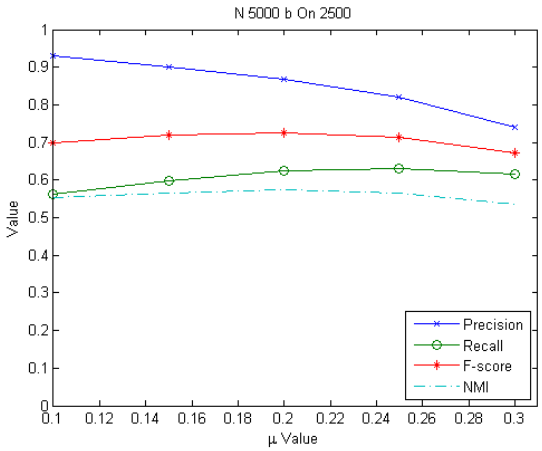


(b) N 5000s On 50%

Figure 4.27: The precision, recall, F-score, and NMI for the LFR tests varying μ for a network with 5000 nodes and using the smaller community size range. a) On = 10% of the total nodes. b) On = 50% of the total nodes.

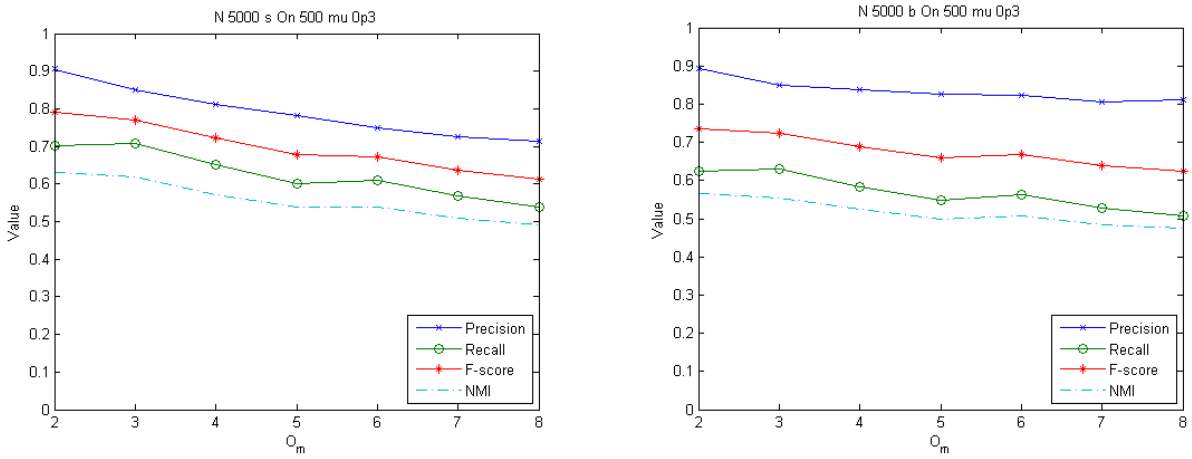


(a) N 5000b On 10%



(b) N 5000b On 50%

Figure 4.28: The precision, recall, F-score, and NMI for the LFR tests varying μ for a network with 5000 nodes and using the larger community size range. a) On = 10% of the total nodes. b) On = 50% of the total nodes.



(a) N 5000s On 10%

(b) N 5000b On 10%

Figure 4.29: The precision, recall, F-score, and NMI for the LFR tests varying O_m for networks with 5000 nodes with 10% of the total nodes belonging to two different communities. a) Using small community size range distribution. b) Using large community size range distribution.

4.4.6 Conclusions and Future Work

This work has focused on developing a computationally inexpensive algorithm capable of detecting overlapping communities in social networks. A novel feature of how we approach the problem is that we define the community structure we are trying to capture based on how that structure would appear at differing scales. The scales specifically considered in this chapter are: the scale of individual nodes, the scale of individual communities, and the scale of the network as a whole. Using the models developed in this work for each of these three scales, we find that applying our algorithm to benchmark tests demonstrates good overall performance. This performance improves with increasing either the number of communities or the sizes of the communities to be detected.

One advantage of our methodology is that it explicitly accounts for multiscale features during the community formation process. This aspect of our approach ensures that the detected communities are always sensible ones with respect to those features. Another distinct advantage of our method is that the way we quantify the features at each scale and tie them together is highly

modular. This allows for the mathematical model of the community structure at any specific scale to be swapped out as appropriate based on the nature of a specific community detection problem.

Future work will focus on further developing the methodology used in our algorithm. One facet meriting further attention is to take advantage of the modularity of our algorithm to incorporate models of alternative features of community structure. The potential advantage of this was demonstrated in Section 4.4.3, where detecting the leader based communities of the karate club network became trivialized by modeling community features as appropriate to the problem. Another avenue to explore is the possibility of chaining together sequences of node versus community level features, where community scale features are treated as node scale features at each higher link in the chain. This allows us to incorporate detection of hierarchical community structures into our algorithm, and further increase its flexibility.

Chapter 5

Conclusions and Future Work

In this thesis, we have examined three problems related to diffusive processes. For each problem, we have used numerical methods to develop a computationally efficient method to either solve the problem directly or augment existing solution methods. We conclude this work with a brief recapitulation of the contributions and conclusions for each of the three problems.

5.1 Contaminant Dispersal

The coarse-scale particle tracking methods outlined in Chapter 2 can be used to augment standard particle tracking methods in order to make them more computationally efficient. We have demonstrated the accuracy of these methods by through numerical experimentation, recovering essentially the same solution as the standard method but with computational costs cut by two to three orders of magnitude.

The computational efficiency of the proposed methods derives from making use of common local domains of practical interest that were simple enough to have analytical representations for the time evolution of concentration distributions within them. One such domain that we considered applies when a particle was in a rock matrix with only one fracture nearby. The other domain applies when a particle was within a fracture that can be considered straight or planar for practical purposes.

In the special case when a particle navigates an isolated planar fracture, we demonstrated that these methods can be combined to efficiently model transport in fractured rock including

the influence of fracture-matrix interactions. Although time-domain random walk methods [9, 21] may be a more computationally efficient option in this particular scenario when the in-fracture velocity fields are uniform, these methods ignore transverse diffusion within fracture planes that causes particles to wander away from streamlines. As noted by Neretnieks [53], such mechanisms are important in modeling radionuclide transport over long time scales, and they are simple to incorporate into our in-fracture method.

The more distinct advantage of our methods are that they are genuine coarse-scale methods for time-stepping, and there is always an understanding as to where the particle lies; the same cannot be said of time-domain random walk methods. Additionally, our methods can be used separately when the assumptions of one method are satisfied but not the other. For example, when in the vicinity of fracture-fracture intersections, reverting to standard particle tracking is more rigorous in the rock matrix, but coarse-scale time-stepping can still be used while particles are within one fracture or the other. Another example comes when a particle is navigating through the rock matrix surrounding a tortuous single fracture; our in-fracture coarse-scale method cannot be applied, but there is nothing to prevent one from using our first passage time approach in this situation.

Future work in this area will focus on utilizing the coarse-scaled methods to augment existing methods for network simulations and also examining their utility in cases with variable fracture apertures. Additionally, we hope to sort out the issues appearing in the adsorption case so that we are able to extend these methods to be applicable to fracture networks where adsorption mechanisms play an important role in contaminant transport.

This work has been submitted for publication in *Applied Mathematical Modelling* [14].

5.2 Glacial Ice Flow

We have found that one can fit empirical data for the relationship between stress and strain rate in ice equally well when using a power law model or a logarithm as the basic functional form for the constitutive law. The favorable features of our proposed log law model are that it avoids the

analytical viscosity blow up inherent in Glen's law, and it fits non-dimensionalized data points more accurately. The benefit of avoiding this singularity is that it can cause large errors to propagate and pollute the solution in the more rapidly moving regions of practical interest.

We explored the utility of the alternative flow law by constructing a model problem, which we then solved using a finite element method. The intention of this work was also to serve as a stepping stone for developing a multigrid solver targeting the problem. Unfortunately, these tests revealed that the data points collected for constructing a model for the flow law are not at all representative of the scales of interest to glacier modelling, and that the only scales of importance to glacier modelling are where Glen's law and the logarithmic flow law differ.

Although Glen's law fits data points more accurately at strain-rate scales relevant to typical glacier dynamics, our model remains accurate and useful for rapidly flowing glaciers. Additionally, there are a number of other shear thinning fluids for which the model may be appropriate, and the technique of fitting a logarithm based model to a power law to avoid viscosity blow up would still be applicable.

A potential alternative direction for future work concerning the ice viscosity blow up would be to exploit the fact that, while the viscosity blows up as the strain rate goes to zero, the product of strain rate and viscosity together goes to zero. The blow up is an artifact resulting from splitting up this single term that goes to zero, given by $\dot{\epsilon}_e$, into two terms. One of these terms blows up to infinity, given by the effective viscosity, and the other rapidly goes to zero, given by the strain rate component. As the combination of these terms is well behaved, it appears that the difficulties arise primarily as a result of trying to force the problem to fit existing fluid flow solvers.

This work has been presented at the 12th Copper Mountain Conference on Iterative Methods [12].

5.3 Community Detection

Our work on the community detection problem focused on developing a computationally inexpensive algorithm capable of detecting overlapping communities in social networks. A novel

feature of how we approached the problem was that we defined the community structure we were trying to capture based on how that structure would appear at multiple scales. The scales we specifically considered in Chapter 4 were: the scale of individual nodes, the scale of individual communities, and the scale of the network as a whole. Using the models we developed for each of these three scales, we found that applying our algorithm to benchmark tests demonstrated that the algorithm performs well on the networks it was intended for.

One of the highlights of our methodology is that we explicitly accounted for multiscale features during the community formation process. This aspect of our approach ensured that the detected communities are always sensible ones with respect to those features. Another highlight of our method is that the way we quantified the features at each scale and tied them together was highly modular. This allowed for the mathematical model of the community structure at any specific scale to be swapped out as appropriate based on the nature of a specific community detection problem, as was demonstrated for the karate club network. Additionally, our algorithm was one of the very few overlapping community detection algorithms that scales linearly with the number of nodes in the network, and is simple to parallelize.

Future work in this area will focus on further developing the methodology used in our algorithm. Currently, it is only equipped to detect communities of one particular type; those where clique membership is indicative of community membership and communities are defined in terms of inlink density. To make our algorithm broadly applicable, this must be extended to include differing notions of community structures. Another avenue to explore is chaining together sequences of node versus community level features, where community scale features are treated as node scale features at each higher link in the chain. This will allow us to incorporate detection of hierarchical community structures into our algorithm, allowing for further flexibility.

This work has been submitted for publication in *Social Network Analysis and Mining* [13].

Bibliography

[1] Ph Ackerer and W Kinzelbach. Modélisation du transport de contaminant par la méthode de marche au hasard: Influence des variations du champ d'écoulement au cours du temps sur la dispersion. The stochastic approach to subsurface flow: Montvillargenne, pages 517–529, 1985.

[2] Rutherford Aris. On the dispersion of a solute in a fluid flowing through a tube. Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences, 235(1200):67–77, 1956.

[3] Benjamin Auffarth. Spectral graph clustering. Course Report. http://www-lehre. inf. uos. de/˜ bauffart/spectral. pdf, 2007.

[4] Benjamin Auffarth. Spectral graph clustering. Universitat de Barcelona, course report for Technicas Avanzadas de Aprendizaj, at Universitat Politecnica de Catalunya, 2007.

[5] Brian Ball, Brian Karrer, and MEJ Newman. Efficient and principled method for detecting communities in networks. Physical Review E, 84(3):036103, 2011.

[6] Jacob Bear, Chin-Fu Tsang, and Ghislain De Marsily. Flow and contaminant transport in fractured rock. Academic Press, 1993.

[7] Rabi N Bhattacharya and Edward C Waymire. Stochastic processes with applications, volume 61. Siam, 2009.

[8] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment, 2008(10):P10008, 2008.

[9] Jacques Bodin, Gilles Porel, Fred Delay, Fabrice Ubertosi, Stéphane Bernard, and Jean-Raynald De Dreuzy. Simulation and analysis of solute transport in 2d fracture/pipe networks: The solfrac program. Journal of contaminant hydrology, 89(1):1–28, 2007.

[10] Béla Bollobás. Modern graph theory, volume 184. Springer, 1998.

[11] Matthew Brand and Kun Huang. A unifying theorem for spectral embedding and clustering. In Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics, 2003.

[12] Michael Brutz. A least-squares finite element method for a glacier model using a generalized glen's flow law. Presented at the 12th Copper Mountain Conference on Iterative Methods, 2012.

[13] Michael Brutz and François Meyer. A modular multiscale approach to overlapping community detection. Social Network Analysis and Mining, submitted.

[14] Michael Brutz and Harihar Rajaram. Coarse-scale particle tracking approaches for contaminant transport in fractured rock. Applied Mathematical Modelling, submitted.

[15] Michael Brutz and Harihar Rajaram. Coarse-scale particle tracking approaches for contaminant transport in fractured rock. Applied Mathematical Modelling, submitted.

[16] John M Chambers, Colin L Mallows, and BW Stuck. A method for simulating stable random variables. Journal of the American Statistical Association, 71(354):340–344, 1976.

[17] Jingchun Chen and Bo Yuan. Detecting functional modules in the yeast protein–protein interaction network. Bioinformatics, 22(18):2283–2290, 2006.

[18] Hua Cheng, Vladimir Cvetkovic, and JO Selroos. Hydrodynamic control of tracer retention in heterogeneous rock fractures. Water resources research, 39(5), 2003.

[19] Aaron Clauset, Mark EJ Newman, and Cristopher Moore. Finding community structure in very large networks. Physical review E, 70(6):066111, 2004.

[20] Tom Clemo and Leslie Smith. A hierarchical model for solute transport in fractured media. Water Resources Research, 33(8):1763–1783, 1997.

[21] Vladimir Cvetkovic, S Painter, N Outters, and JO Selroos. Stochastic simulation of radionuclide migration in discretely fractured rock near the äspö hard rock laboratory. Water resources research, 40(2), 2004.

[22] Leon Danon, Albert Diaz-Guilera, Jordi Duch, and Alex Arenas. Comparing community structure identification. Journal of Statistical Mechanics: Theory and Experiment, 2005(09):P09008, 2005.

[23] Hans De Sterck, Thomas A Manteuffel, Stephen F McCormick, Killian Miller, James Pearson, John Ruge, and Geoffrey Sanders. Smoothed aggregation multigrid for markov chains. SIAM Journal on Scientific Computing, 32(1):40–61, 2010.

[24] Hans De Sterck, Thomas A Manteuffel, Stephen F McCormick, Quoc Nguyen, and John Ruge. Multilevel adaptive aggregation for markov chains, with application to web ranking. SIAM Journal on Scientific Computing, 30(5):2235–2262, 2008.

[25] Russell L Detwiler, Harihar Rajaram, and Robert J Glass. Solute transport in variable-aperture fractures: An investigation of the relative importance of taylor dispersion and macrodispersion. Water Resources Research, 36(7):1611–1625, 2000.

[26] Alfred E Dunlop and Brian W Kernighan. A procedure for placement of standard cell vlsi circuits. IEEE Transactions on Computer-Aided Design, 4(1):92–98, 1985.

[27] TS Evans and R Lambiotte. Line graphs, link partitions, and overlapping communities. Physical Review E, 80(1):016105, 2009.

[28] Santo Fortunato. Community detection in graphs. Physics Reports, 486(3):75–174, 2010.

[29] Michelle Girvan and Mark EJ Newman. Community structure in social and biological networks. arXiv preprint cond-mat/0112110, 2001.

[30] J.W. Glen. The creep of polycrystalline ice. Proceedings of the Royal Society of London, 228(1175):519–538, March 1955.

[31] JW Glen. The flow law of ice: A discussion of the assumptions made in glacier theory, their experimental foundations and consequences. IASH Publ, 47:171–183, 1958.

[32] D.L. Goldsby and D.L. Kohlstedt. Superplastic deformation of ice: Experimental observations. Journal of Geophysical Research, 106(B6):11,017–11,030, 2001.

[33] Gene H Golub and Charles F Van Loan. Matrix computations, volume 3. JHU Press, 2012.

[34] Prem K Gopalan and David M Blei. Efficient discovery of overlapping communities in massive networks. Proceedings of the National Academy of Sciences, 110(36):14534–14539, 2013.

[35] Steve Gregory. Finding overlapping communities in networks by label propagation. New Journal of Physics, 12(10):103018, 2010.

[36] Ralf Greve and Heinz Blatter. Dynamics of ice sheets and glaciers. Springer, 2009.

[37] E Hakami and N Barton. Aperture measurements and flow experiments using transparent replicas of rock joints. Rock Joints, pages 383–390, 1990.

[38] T.H. Jacka. The time and strain required for development of minimum strain rates in ice. Cold Regions Science and Technology, 8(3):261–268, 1984.

[39] PM Jardine, WE Sanford, JP Gwo, OC Reedy, DS Hicks, JS Riggs, and WB Bailey. Quantifying diffusive mass transfer in fractured shale bedrock. Water Resources Research, 35(7):2015–2030, 1999.

[40] R.E. Johnson and R.M. Mcmeeking. Near-surface flow in glaciers obeying glen's law. The Quarterly Journal of Mechanics and Applied Mathematics, 37.4:273–291, 1984.

[41] Eric M LaBolle, Graham E Fogg, and Andrew FB Tompson. Random-walk simulation of transport in heterogeneous porous media: Local mass-conservation problem and implementation methods. Water Resources Research, 32(3):583–593, 1996.

[42] Andrea Lancichinetti and Santo Fortunato. Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. Physical Review E, 80(1):016118, 2009.

[43] Andrea Lancichinetti and Santo Fortunato. Community detection algorithms: a comparative analysis. Physical review E, 80(5):056117, 2009.

[44] Andrea Lancichinetti, Santo Fortunato, and János Kertész. Detecting the overlapping and hierarchical community structure in complex networks. New Journal of Physics, 11(3):033015, 2009.

[45] W. Leng, L.L. Ju, M. Gunzburger, S. Price, and T. Ringler. A parallel high-order accurate finite element nonlinear stokes ice sheet model and benchmark experiments. JGR-Earth Surface, 117, 2012.

[46] Wei Leng, Lili Ju, Max Gunzburger, Stephen Price, and Todd Ringler. A parallel high-order accurate finite element nonlinear stokes ice sheet model and benchmark experiments. Journal of Geophysical Research: Earth Surface (2003–2012), 117(F1), 2012.

[47] L.A. Lliboutry. The dynamics of temperate glaciers from the detailed viewpoint. J. Glaciology, 8:185–205, 1969.

[48] Piotr Maloszewski and Andrzej Zuber. Mathematical modeling of tracer behavior in short-term experiments in fissured rocks. Water Resources Research, 26(7):1517–1528, 1990.

[49] M. Mellor and D.M. Cole. Deformation and failure of ice under constant stress or constant strain-rate. Cold Regions Science and Technology, 5:201–219, 1981.

[50] M. Mellor and D.M. Cole. Stress/strain/time relations for ice under uniaxial compression. Cold Regions Science and Technology, 6:207–230, 1983.

[51] M. Mellor and R. Testa. Creep of ice under low stress. Journal of Glaciology, 8(52):147–152, 1969.

[52] M. Mellor and R. Testa. Effect of temperature on the creep of ice. Journal of Glaciology, 8:131–145, 1969.

[53] Ivars Neretnieks. Channeling with diffusion into stagnant water and into a matrix in series. Water resources research, 42(11), 2006.

[54] Mark EJ Newman and Michelle Girvan. Finding and evaluating community structure in networks. Physical review E, 69(2):026113, 2004.

[55] Krzysztof Nowicki and Tom A B Snijders. Estimation and prediction for stochastic blockstructures. Journal of the American Statistical Association, 96(455):1077–1087, 2001.

[56] J.F. Nye. The flow law of ice from measurements in glacier tunnels, laboratory experiments and the jungfraufirn borehole experiment. Proceedings of the Royal Society of London, 219(1139):477–489, October 1953.

[57] National Research Council (US). Committee on Fracture Characterization and Fluid Flow. Rock fractures and fluid flow: contemporary understanding and applications. Natl Academy Pr, 1996.

[58] M Necati Ozisik. Heat conduction. Wiley, 1993.

[59] Gergely Palla, Imre Derényi, Illés Farkas, and Tamás Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. Nature, 435(7043):814–818, 2005.

[60] P. Pimienta and P. Duval. Rate controlling processes in the creep of polar glacier ice. Journal de Physique, 48.

[61] Alex Pothen. Graph partitioning algorithms with applications to scientific computing. In Parallel Numerical Algorithms, pages 323–368. Springer, 1997.

[62] Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. Near linear time algorithm to detect community structures in large-scale networks. Physical Review E, 76(3):036106, 2007.

[63] Bradley S Rees and Keith B Gallagher. Overlapping community detection by collective friendship group inference. In Advances in Social Networks Analysis and Mining (ASONAM), 2010 International Conference on, pages 375–379. IEEE, 2010.

[64] Bradley S Rees and Keith B Gallagher. Overlapping community detection using a community optimized graph swarm. Social Network Analysis and Mining, 2(4):405–417, 2012.

[65] Jörg Reichardt and Stefan Bornholdt. Statistical mechanics of community detection. Physical Review E, 74(1):016110, 2006.

[66] Paul Reimus, Greg Pohll, Todd Mihevc, Jenny Chapman, Marc Haga, Brad Lyles, Sean Kosinski, Rich Niswonger, and Peter Sanders. Testing and parameterizing a conceptual model for solute transport in a fractured granite using multiple tracers in a forced-gradient test. Water Resources Research, 39(12), 2003.

[67] Martin Rosvall and Carl T Bergstrom. Maps of random walks on complex networks reveal community structure. Proceedings of the National Academy of Sciences, 105(4):1118–1123, 2008.

[68] Jan-Olof Selroos, Douglas D Walker, Anders Ström, Björn Gylling, and Sven Follin. Comparison of alternative modelling approaches for groundwater flow in fractured rock. Journal of Hydrology, 257(1):174–188, 2002.

[69] N.K. Sinha. Short-term rheology of polycrystalline ice. J. Glaciology, 21(85):457–473, 1978.

[70] G.D. Smith and L.W. Morland. Viscous relations for the steady creep of polycrystalline ice. Cold Regions Science and Technology, 5:141–150, 1981.

[71] S. Steinemann. Flow and recrystallisation of ice. IASH 39, Tome IV, Assembleee Generale de Rome 1954:449–462, 1956.

[72] EA Sudicky and EO Frind. Contaminant transport in fractured porous media: Analytical solutions for a system of parallel fractures. Water Resources Research, 18(6):1634–1642, 1982.

[73] Andrew FB Tompson and Lynn W Gelhar. Numerical simulation of solute transport in three-dimensional, randomly heterogeneous porous media. Water Resources Research, 26(10):2541–2562, 1990.

[74] Chin-Fu Tsang. Tracer transport in fracture systems. Flow and contaminant transport in fractured rock, pages 237–266, 1993.

[75] C. J. Van Der Veen. Fundamentals of Glacier Dynamics. Balkema, Rotterdam, 1999.

[76] Ken Wakita and Toshiyuki Tsurumi. Finding community structure in a mega-scale social networking service. In Proceedings of IADIS international conference on WWW/Internet 2007, pages 153–162, 2007.

[77] Fang Wu and Bernardo A Huberman. Finding communities in linear time: a physics approach. The European Physical Journal B-Condensed Matter and Complex Systems, 38(2):331–338, 2004.

[78] Jierui Xie, Stephen Kelley, and Boleslaw K Szymanski. Overlapping community detection in networks: The state-of-the-art and comparative study. ACM Computing Surveys (CSUR), 45(4):43, 2013.

[79] W Zachary. An information flow modelfor conflict and fission in small groups1. Journal of anthropological research, 33(4):452–473, 1977.

[80] A Zafarani and RL Detwiler. Solute transport in three-dimensional variable-aperture discrete fracture networks. In AGU Fall Meeting Abstracts, volume 1, page 04, 2012.