UNIVERSITY OF CALIFORNIA, SAN DIEGO

Fractional Diffusion: Numerical Methods and Applications in Neuroscience

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Bioengineering with a specialization in Multi-Scale Biology

by

Nirupama Bhattacharya

Committee in charge:

Professor Gabriel A. Silva, Chair
Professor Henry Abarbanel
Professor Gert Cauwenberghs
Professor Todd Coleman
Professor Marcos Intaglietta

2014

DEDICATION

Thank you to my dearest family and friends for all your love and support over the years.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

Chapter 3, in part, has been submitted for publication of the material as it may appear in the Journal of Computational Physics, 2015, Christopher L. MacDonald, Nirupama Bhattacharya, Brian P. Sprouse, Gabriel A. Silva. The dissertation author was a co-author of this paper.

Chapter 4, in part, is currently being prepared for submission for publication of the material. Nirupama Bhattacharya and Gabriel A. Silva. The dissertation author was the primary investigator and author of this paper.

Chapter 5, in part, is currently being prepared for submission for publication of the material. Nirupama Bhattacharya and Gabriel A. Silva. The dissertation author was the primary investigator and author of this paper.

VITA

2009	B.S. in Electrical Engineering with a specialization in Computer Software, with Minor in Physics, Stanford University, Stanford
2011	M.S. in Bioengineering, University of California, San Diego
2014	Ph.D. in Bioengineering with a specialization in Multi-Scale Biology, University of California, San Diego

PUBLICATIONS

Nirupama Bhattacharya, Gabriel A. Silva, "Stability and Complexity Analyses of Finite Difference Algorithms for the Time-fractional Diffusion Equation," 2015. [*Manuscript in preparation*]

Nirupama Bhattacharya, Gabriel A. Silva, "An Efficient Finite Difference Approach to Solving the Time-fractional Diffusion Equation," 2015. [*Manuscript in preparation*]

Christopher L. MacDonald, Nirupama Bhattacharya, Brian P. Sprouse, Gabriel A. Silva, "Efficient computation of the Grünwald fractional diffusion derivative using adaptive time step memory," 2015. [*Submitted*]

Dr. S. S. Pai, N. Bhattacharya and Dr. V. K. Nagpal, "Development of Probabilistic Structural Analysis of Annular Deformation Resistance Welding Process," ASME Turbo Expo, June 2010, IGTI Conference, Glasgow, UK 2010.

ABSTRACT OF THE DISSERTATION

Fractional Diffusion: Numerical Methods and Applications in Neuroscience

by

Nirupama Bhattacharya

Doctor of Philosophy in Bioengineering with a specialization in Multi-Scale Biology

University of California, San Diego, 2014

Professor Gabriel A. Silva, Chair

In biological contexts, experimental evidence suggests that classical diffusion is not the best description in instances of complex biophysical transport. Instead, anomalous diffusion has been shown to occur in various circumstances, potentially caused by such underlying mechanisms as active transport, macromolecular crowding in a complex and tortuous extracellular or intracellular environment, or complex media geometry.

Elegant ways of simulating these complicated transport processes are to connect the spatial characteristics of a medium (porosity or tortuosity of a complex extracellular environment), to fractional order operators. Some approaches include special random walk models representing crowded or disordered media; at the continuum limit, these random walk models approach fractional differential equations (FDEs), including variations of the fractional diffusion equation.

Fractional differential equations are an extension of classical integer-order differential equations, and in recent decades have been increasingly used to model the dynamics of complex systems in a wide variety of fields including science, engineering, and finance. However, finding tractable and closed form analytical solutions to FDEs, including the fractional diffusion equation and its variants, is generally extremely difficult and often not feasible, and especially so when integrating these equations into more complex physical models with multiple other components; therefore, the development of stable and accurate numerical methods is vital.

In this thesis we explore the topic of anomalous diffusion and the fractional diffusion equation from multiple perspectives. We begin by connecting the micromolecular behavior of diffusing particles undergoing anomalous diffusion, to the general derivation of the fractional diffusion equation. We then develop numerical approaches to efficiently solve the time-fractional diffusion equation, and characterize these methods in terms of accuracy, stability, and algorithmic complexity.

We then make use of these numerical methods by applying fractional diffusion to a model of the signaling events leading up the induction of long-term depression (LTD). We leverage the fact that the fractional diffusion equation can capture the complex geometry in which diffusing particles travel, and use this to simplify an existing model of LTD induction; furthermore, we show that our modified model is capable of retaining the most important functionality of the original model.

Chapter 1

Introduction

1.1 Fractional Calculus, Fractional Differential Equations, and Applications in Science and Engineering

The idea of non-integer order differentiation and integration has existed for almost as long as the foundational ideas of classical calculus have been, and over the centuries the field of fractional calculus has been developed by many of the same mathematicians that made fundamental advancements in classical calculus. Leibniz mentions fractional calculus in a letter to L'Hospital as early as 1695, and the theory was further developed by Liouville, Riemann, Grunwald, Letnikov, Riemann, and many others [39].

Analogous to ordinary and partial differential equations used to model physical phenomena using classical calculus, fractional differential equations (FDEs) are an extension of the theory of fractional calculus. However, because of the difficulty in finding closed form analytical solutions to FDEs, there has been limited work done in applying them to physical problems, until recent decades. With the advent of increasing computational capacity of personal computers and advances in numerical methods, there has been a resurgence in interests and development of solving FDEs, as it has been shown that they can appropriately be applied to many branches of science and engineering to represent complex physical processes whose dynamics are not captured by classical integer order partial differential equations.

Fractional differential equations have been useful for modeling such uniquely different applications as advection-dispersion transport phenomena [2], wave propagation in bone and other rigid porous materials [46, 10], viscoelasticity [48] edge detection in image processing [33], and neurodynamics and modeling of signal processing in neuronal dendrites [15]. Richard Magin discusses a wide array of fractional calculus modeling applications in the field of bioengineering alone, including cell signaling across membranes, feedback control in neural systems, neurodynamics, capacitor and dielectric models, the behavior of viscoelastic materials, circuit models of electrode interfaces, cell biomechanics, electrochemistry (especially relating to cardiac tissue-electrode interfaces), among many other biomedical applications [30, 31].

1.2 Fractional Diffusion

As we have seen, fractional differential equations are being applied to various fields in science and engineering. In particular, there has been a lot of recent work with fractional differential equations to model complex biophysical transport phenomena, including diffusion. Experimental evidence suggests that classical Brownian diffusion according to Fick's laws, may not accurately model many types of biological data. Various cases of anomalous diffusion in such places as white matter neural fiber tracts, may be caused by active transport, macromolecular crowding in a complex and tortuous extracellular or intracellular environment, or a complex media geometry [3, 44, 47]. Elegant ways of simulating these complicated transport processes are to connect the spatial characteristics of a medium (porosity or tortuosity of a complex extracellular environment), to fractional order operators. Some approaches include specific random walk models to represent crowded or disordered media; at the continuum limit, these random walk models approach variations of the fractional diffusion equation [53, 54, 55, 14, 51, 36, 13]. Finding tractable and closed form analytical solutions to the fractional diffusion equation and its variants, is generally extremely difficult and often not feasible, and especially so when integrating these equations into more complex physical models with multiple other components; therefore, the development of stable and accurate numerical methods is vital.

This thesis is an effort to fully explore the fractional diffusion equation, from its roots in the statistical behavior of diffusing particles, to various numerical methods of solving this equation in an efficient way that balances tradeoffs between accuracy, stability, and algorithmic complexity.

In Chapter 2 we explore the theoretical foundations of the fractional diffusion equation. We begin with the random walk scenario often used to illustrate classical diffusion, and by making adjustments on our set of assumptions, we connect the random walk scenario with special probability distributions from which we can then derive the fractional diffusion equation.

In Chapters 3-5 we explore several computational algorithms for efficiently solving the time-fractional diffusion equation, and consider various aspects of these algorithms, including computational time, accuracy, stability, and the difficulty or ease involved in integrating these numerical methods with larger models that often have several other complex components aside from the fractional diffusion element.

In Chapter 6, we explore the application of fractional diffusion in capturing the complex geometry of spiny dendrites in Purkinje neurons, and how this affects the transport of second messenger signaling molecules such as IP_3 and calcium, in the induction of long-term-depression (LTD). Hernjak et. al have built a model of the events leading up to LTD, which takes into account the complex geometry of spiny dendrites in Purkinje neurons by explicitly modeling dendritic and spine compartments, and the flux between the two compartments through the narrow bottleneck that is the spine neck. We hypothesize that the use of fractional diffusion can reflect the dynamics caused by this geometric complexity, and attempt to create a simplified, modified model that can still capture the main functionality of the original, including the simulation of parallel fiber (PF) stimulation and climbing fiber (CF) activation of the Purkinje neuron.

Lastly, in Chapter 7 we summarize several fronts for future investigations, including continued study of how fractional diffusion of IP_3 , can affect the spread of LTD along spiny dendrites of Purkinje cells. In addition, we explore the concept of fractional diffusion to describe calcium signaling at several spatial scales in neurons and other excitable media, including cardiac myocytes. And finally we briefly revisit our original motivation for exploring fractional diffusion in the first place, which was observed diffusion profiles of ATP and its potential role in calcium signaling in astrocyte networks.

Chapter 2

From Random Walk to the Fractional Diffusion Equation

2.1 Introduction to the Random Walk Framework

The random walk framework is a well known construct used to connect the micromolecular level behavior of discrete particles or objects, to the average macromolecular behavior of the group of objects, as a continuum. We consider a one-dimensional example where a particle has set step size ($\Delta z = l$), and at each timestep, can go either to the right ($+\Delta z$) or to the left ($-\Delta z$) neighboring sites, with equal probability for either option. Beginning from $z_0 = 0$, after N steps, the net displacement of the particle is given by $r(N) = \sum_{i=1}^{N} \Delta z_i$. Δz_i is a random variable and since each direction has equal probability of being chosen in this scenario, $\langle \Delta z_i \rangle = 0$.

We can now derive an expression for the mean squared displacement:

$$\begin{aligned} r^2(N) &= \sum_{j=1}^{N} \Delta z_j \sum_{j=1}^{N} \Delta z_k \\ &= \sum_{j=1,k=j}^{N} \Delta z_j^2 + \sum_{j=1,k\neq j}^{N} \Delta z_j \Delta z_k \end{aligned} \tag{2.1.1}$$

$$\left\langle r^2(N) \right\rangle = Nl^2 + \left\langle \sum_{j=1,k\neq j}^{N} \Delta z_j \Delta z_k \right\rangle \tag{2.1.2}$$

Since Δz_j and Δz_k are random variables, and are independent of each other (i.e., the result at each step is independent of the results at all other steps), the second term in Eq. 2.1.2 vanishes, leaving us with a simple expression for the root mean squared (*RMS*) displacement: $\left\langle r^2(N) \right\rangle = Nl^2 \Rightarrow RMS\ = l\sqrt{N}$

If we add the concept of particle collisions to our simple random walk scenario, we can express the distance travelled between two collisions as $l = \langle v \rangle \tau$ where $\langle v \rangle$ is the average speed of a given particle (averaged over all timesteps), and τ is the collision time. Over a simulation time t, we have $N = t/\tau$ collisions, or steps in the scenario. We can now write the expression for mean squared displacement: $\left\langle r^2(N) \right\rangle = l \langle v \rangle t$. In a three dimensional system that is isotropic and at equilibrium, we can assume

$$\left\langle r_x^2(N) \right\rangle = \left\langle r_y^2(N) \right\rangle = \left\langle r_z^2(N) \right\rangle = \frac{\left\langle r^2(N) \right\rangle}{3}$$

and therefore

$$\begin{aligned} \left\langle r^2(N) \right\rangle &= 3 \langle v \rangle lt && (2.1.3) \\ &= Dt && (2.1.4) \end{aligned}$$

where D is the diffusion coefficient and is defined in this context as $D = 3 < v > l$. Notice that for classical diffusion there is a clear linear relationship between the mean squared displacement of the particle, and the elapsed time. We will see in the next section that anomalous diffusion refers to systems where this relationship is nonlinear.

The diffusion coefficient and the general phenomena of classical diffusion can be explored through various other formalisms, including Langevin's equation, which involves the physics of forces acting on particles with mass and velocity, and also including Fick's laws, which deal with particle flux and from which the classical diffusion equation can also be derived.

2.2 Einstein's Formalism

We now arrive at Einstein's Formalism, which relates the random walk scenario to probability distributions. We can think of the solution to the random walk scenario as a probability distribution $P(z,t)$ which gives the probability of a particle being at position z, at time t. We can write

$$P(z,t) = \int_{\infty}^{\infty} P(z-\Delta z, t-\Delta t)\, q_{\Delta z}(\Delta z)\, d\Delta z \tag{2.2.1}$$

where $P(z-\Delta z, t-\Delta t)$ is the probability of the particle being at location $z-\Delta z$, at time $t-\Delta t$ (one step away, at one timestep earlier), $q_{\Delta z}$ is the probability distribution of random walk steps of all possible sizes (which means $q_{\Delta z}(\Delta z)$ gives us the probability of making a step of length Δz), and the integral signifies summing our results over all possible steps.

For classical diffusion we assume that Δt is small, and Δz is small compared to the size of the system, which suggests that $q_{\Delta z}(\Delta z)$ is zero for large Δz, and nonzero for small Δz. With these assumptions we expand the term $P(z-\Delta z, t-\Delta t)$ as a Taylor series expansion, around location z and time t:

$$\begin{aligned} P(z,t) &= \int_{\infty}^{\infty} P(z,t)\, q_{\Delta z}(\Delta z)\, d\Delta z \\ &- \int_{\infty}^{\infty} \Delta t \frac{\partial P(z,t)}{\partial t} q_{\Delta z}(\Delta z)\, d\Delta z \\ &- \int_{\infty}^{\infty} \Delta z \frac{\partial P(z,t)}{\partial z} q_{\Delta z}(\Delta z)\, d\Delta z \\ &+ \int_{\infty}^{\infty} \frac{\Delta z^2}{2} \frac{\partial^2 P(z,t)}{\partial z^2} q_{\Delta z}(\Delta z)\, d\Delta z \end{aligned}$$

We normalize the distribution $q_{\Delta z}$ $\left(\int q_{\Delta z} d\Delta z = 1\right)$ and assume it is symmetric, so that the mean $\int \Delta z q_{\Delta z} d\Delta z = 0$. Lastly, we define the variance as the second moment of the step distribution function: $\sigma_{\Delta z}^2 = \int \Delta z^2 q_{\Delta z}(\Delta z)\, d\Delta z$. From these assumptions and definitions, we can then recover the classical diffusion equation

$$\frac{\partial P(z,t)}{\partial t} = \frac{\sigma_{\Delta z}^2}{2\Delta t} \frac{\partial^2 P(z,t)}{\partial z^2} \tag{2.2.2}$$

From the Central Limit Theorem, we can find that the form of the solution to Eq. 2.2.2 is a Gaussian if we make the following assumptions:

- $q_{\Delta z}(\Delta z)$ has a finite mean $\mu_{\Delta z}$ and variance $\sigma_{\Delta z}^2$
- all Δz_i are mutually independent
- the number of steps N is large

2.3 Continuous Time Random Walk and the Fractional Diffusion Equation

We return to the idea of anomalous diffusion and qualitatively compare it to classical diffusion in terms of particle trajectories. Normal diffusion occurs in systems close to equilibrium and although step sizes are small and irregular, particle trajectories look relatively homogenous over time. Anomalous diffusion often occurs in systems further from equilibrium and we notice two new characteristics of particle trajectories:

- long uninterrupted flights
- local traps where a particle might stay for a long time

As mentioned in the last section, the classification of whether a system undergoes classical or anomalous diffusion, is determined by the relationship between mean squared displacement of a particle, and time:

$$\left\langle r^2(t)\right\rangle =\sim t^\gamma \tag{2.3.1}$$

where $\left\langle r^2(t)\right\rangle$ is the mean squared displacement of the particle after time t has elapsed. This value is proportional (through the diffusion coefficient D), to t^γ. The case for $\gamma = 1$ represents normal diffusion, with $\gamma \neq 1$ representing anomalous diffusion. $\gamma < 1$ refers to subdiffusion and $\gamma > 1$ refers to superdiffusion.

One way in which anomalous diffusion behavior has been modeled is through the continuous time random walk framework, which can then be used to derive the fractional diffusion equation in space and time.

In the spatial dimension, if we allow particles to take large steps (up to the size of the system), we can have unbounded step sizes for infinite systems. We can quantify this by choosing the Lévy distribution for our step size probability distribution $q_{\Delta z}(\Delta z)$, which has nonzero probabilities for large Δz. We also make time a continuous variable that evolves dynamically so that for large or infinite step sizes we can have corresponding large or infinite timesteps. Together this relaxation on the limits of step sizes and timesteps, completes the definition of the continuous time random walk scheme (CTRW).

From the CTRW relaxations on step sizes and timesteps, we can derive different forms of the probability distribution function $P(z,t)$. If we assume Δz is small, we can recover the classical diffusion equation again, and as stated in the last section, find that the form of the solution is a Gaussian function. If we assume Δz is not small, then we can derive equations representing superdiffusion or subdiffusion, depending on what our actual $q_{\Delta z}(\Delta z)$ distribution is. We also note that from the previous discussion of unbounded step sizes and time steps, CTRW can be described as a non-local or non-Markovian process, and it is thus consistent that this framework is used to derive a fractional diffusion equation that represents anomalous diffusion, a non-local process far from equilibrium.

At this point we can introduce particular forms for $q_{\Delta z}(\Delta z)$ and $q_{\Delta t}(\Delta t)$. For the spatial dimension we can set the step size distribution as a Lévy distribution, which is defined in Fourier space as

$$\hat{q}^{\vartheta}_{\Delta z}(k) = exp\left(-a|k|^{\vartheta}\right) \tag{2.3.2}$$

for $0 < \vartheta \leq 2$. For $\vartheta = 2$ we recover the Gaussian distribution and for $\vartheta = 1$ we recover the Cauchy distribution. See Fig. 2.3.1 for the shape of Lévy distributions for various λ parameters. We can see that as λ is decreased, the 'fatter' the 'tails' of the distributions; this is why the Lévy distributions are often referred to as a class of

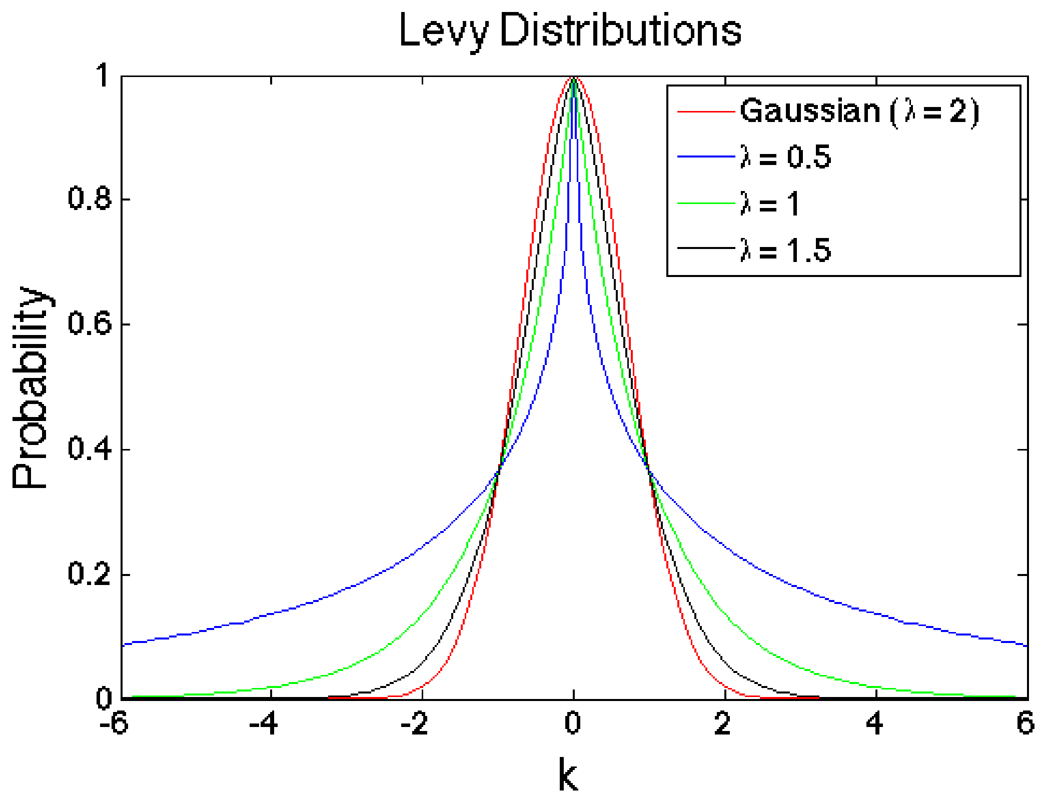


Figure 2.3.1: The shape of various Lévy distributions in Fourier space. As required, we recover the Gaussian distribution when $\lambda = 2$. As λ decreases, the 'fatter' the 'tails' of the distribution become; that is, the higher the probabilities for larger magnitude values of k. Note that these plots are for distributions that have not been normalized, and therefore the true probability will be scaled by some factor that is dependent on λ.

long-tailed distributions.

Although there is no closed form expression for the Lévy distributions in space, we can approximate the form of the distribution by

$$q_{\Delta z}(\Delta z) \sim |\Delta z|^{-1-\vartheta} \tag{2.3.3}$$

Although there are scaling and normalization factors involved, qualitatively the shape of the distributions in real space, are the same as in Fourier space.

We can see how these distributions corresponds to the particle trajectory characteristic of long uninterrupted flights, which are essentially represented by large step sizes Δz. The long-tailed nature of the Lévy distribution assures us that the probability of a particle taking such large step sizes, is no longer extremely improbable, as it would be in a system undergoing classical diffusion. In addition, the variance σ_{ϑ}^{2} is infinite for $\vartheta < 2$ and the mean is infinite for $\vartheta \leq 1$ For $\vartheta = 2$ we have a finite variance and since we recover the Gaussian distribution, we can derive from this the classical diffusion equation.

For the time domain, we use an asymmetrical Lévy distribution which is appropriate defined in Laplacian space as

$$q_{\Delta t}^{\gamma}(s) = exp(-bs^{\gamma}) \tag{2.3.4}$$

for $0 < \alpha \leq 1$. See Fig. 2.3.2 for the shapes of these asymmetrical versions of the two sided Lévy distributions.

In the time domain we can approximate this as

$$q_{\Delta t} \sim t^{-1-\gamma} \tag{2.3.5}$$

Analogous to the spatial case, although there are scaling and normalization factors involved, qualitatively the shape of the distributions in the real time domain, are the same as in Laplace space; this corresponds to increased likelihood of long time steps, which can be interpreted as long wait times between jumps.

Using Fourier and Laplace transforms, and the space and time distributions given in Eq. 2.3.2 2.3.4, we can derive the fractional diffusion equation

$$\frac{\partial^{\gamma} P(z,t)}{\partial t^{\gamma}} = \frac{a}{b}\frac{\partial^{\vartheta} P(z,t)}{\partial z^{\vartheta}} \tag{2.3.6}$$

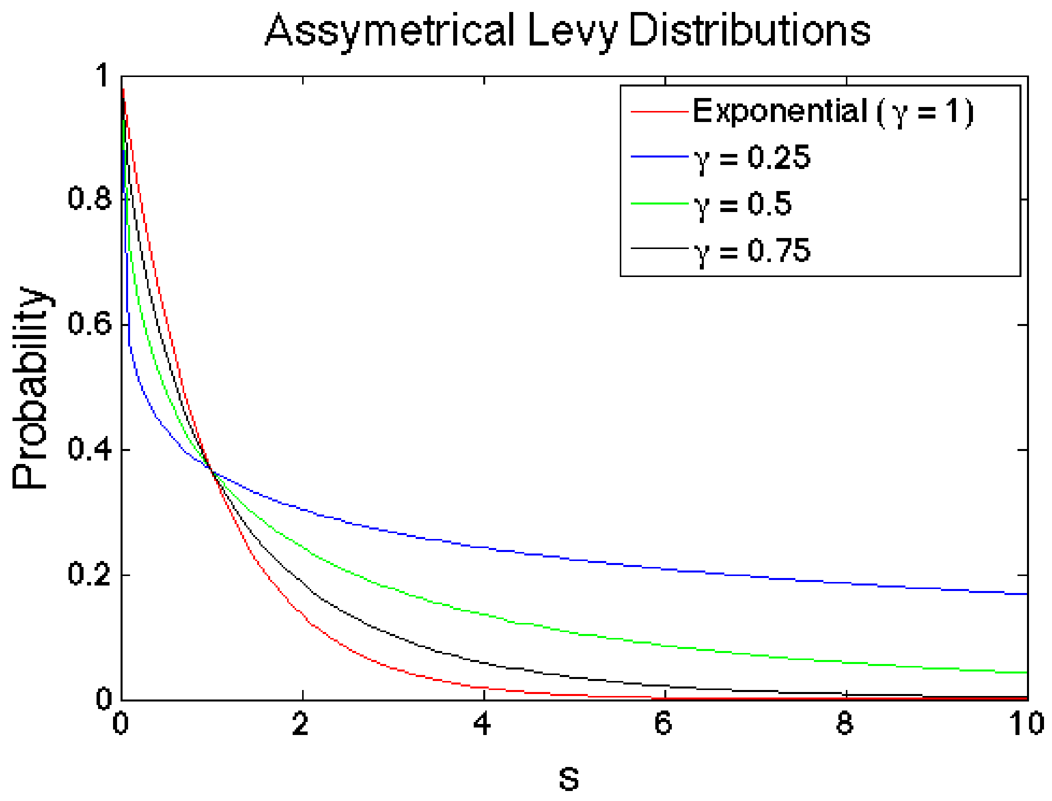


Figure 2.3.2: The shape of various Lévy distributions in Laplace space. As required, we recover an exponential decay distribution when $\gamma = 1$. As γ decreases, the 'fatter' the 'tails' of the distribution become; that is, the higher the probabilities for larger values of s. Note that these plots are for distributions that have not been normalized, and therefore the true probability will be scaled by some factor that is dependent on γ.

Further details about the derivation of Eq. 2.3.6 and its variants, are outside the scope of this thesis and we refer the reader to [51, 36] for a more complete analysis. We also note that γ in Eq. 2.3.6 is the same parameter as in 2.3.1; the discussions in [51, 36] reestablish this nonlinear relationship between mean squared displacement, and elapsed time, from the probability distribution functions in equations 2.3.2 and 2.3.4. However, we do emphasize the following concepts. It can be seen over the course of this section that there is a clear connection between probability distribution functions, the classical diffusion equation, and the space and time-fractional diffusion equation. The exact path of the derivation and the resulting partial differential equation is dependent on the assumptions we make about the geometry of the system and corresponding effect this has on particle step size Δz and timestep Δt between steps. For classical diffusion, we make the assumptions of small Δz and Δt, independent walks in all directions and between steps, and Markovian local processes in systems close to equilibrium. These assumptions allow us to use the Central Limit Theorem to tie the random walk behavior on a micromolecular scale Gaussian function.

In contrast, for certain geometries and diffusive contexts, we can relax several of these assumptions and find appropriate probability distribution functions for step size and timestep. These distributions should reflect certain details of the system, such as the presence of external fields, the nature of the media through which particles are diffusing, and the behavior of the particles themselves (for example, intermolecular attractive/repulsive forces). Once the appropriate probability distribution functions are found, we can determine the form of fractional diffusion equation, which can be much more complicated than given in Eq. 2.3.6 if it is to reflect additional components of the system such as advection, the presence of external force fields, and other details.

The focus of the remainder of this thesis will be on numerical methods to solve the two-dimensional time-fractional diffusion equation, which is representative of anomalous diffusion. The equation is a particular form of Eq. 2.3.6 with $\vartheta = 2$:

$$\frac{\partial^{\gamma} P(\vec{z},t)}{\partial t^{\gamma}} = D\nabla^2 P(\vec{z},t) \tag{2.3.7}$$

where D is the diffusion coefficient (with units $\frac{spatial\ units^2}{time\ units^{\gamma}}$). From here forth we connect anomalous diffusion with the time-fractional diffusion, unless otherwise explicitly stated (as in Chapter 7 where we briefly touch on the space fractional diffusion).

2.4 Fundamental Solution

Although it is difficult to find closed form solutions for Eq. 2.3.7, for the fundamental problem in unbounded space where the initial condition is $P(z,0) = \delta(z)$, we can write a closed form solution (impulse response) in terms of Fox's H-function [36]:

$$P(z,t) = \frac{1}{\sqrt{4Dt^{\gamma}}} H^{1,0}_{1,1}\left[\frac{|z^2|}{\sqrt{Dt^{\gamma}}} \middle| \begin{array}{c} (1-\gamma/2, \gamma/2) \\ (0,1) \end{array} \right] \tag{2.4.1}$$

where the Fox function is an analytical representation of Lévy distributions in real space, and is a general case of many of the special functions occurring in math and statistics. It can be expressed in computational form as

$$\begin{aligned} H^{m,n}_{p,q}(x) &= H^{m,n}_{p,q}\left[x \middle| \begin{array}{c} (a_p, A_p) \\ (b_q, B_q) \end{array} \right] \\ &= \sum_{h=1}^{m} \sum_{v=0}^{\infty} \frac{\prod_{j=1, j\neq h}^{m} \Gamma\left(b_j - B_j (b_h + v)/B_h\right)}{\prod_{j=m+1}^{q} \Gamma\left(1 - b_j + B_j (b_h + v)/B_h\right)} \\ &\quad \times \frac{\prod_{j=1}^{n} \Gamma\left(1 - a_j + A_j (b_h + v)/B_h\right)}{\prod_{j=n+1}^{p} \Gamma\left(a_j - A_j (b_h + v)/B_h\right)} \\ &\quad \times \frac{(-1)^v x^{(b_h+v)/B_h}}{v! B_h} \end{aligned} \tag{2.4.2}$$

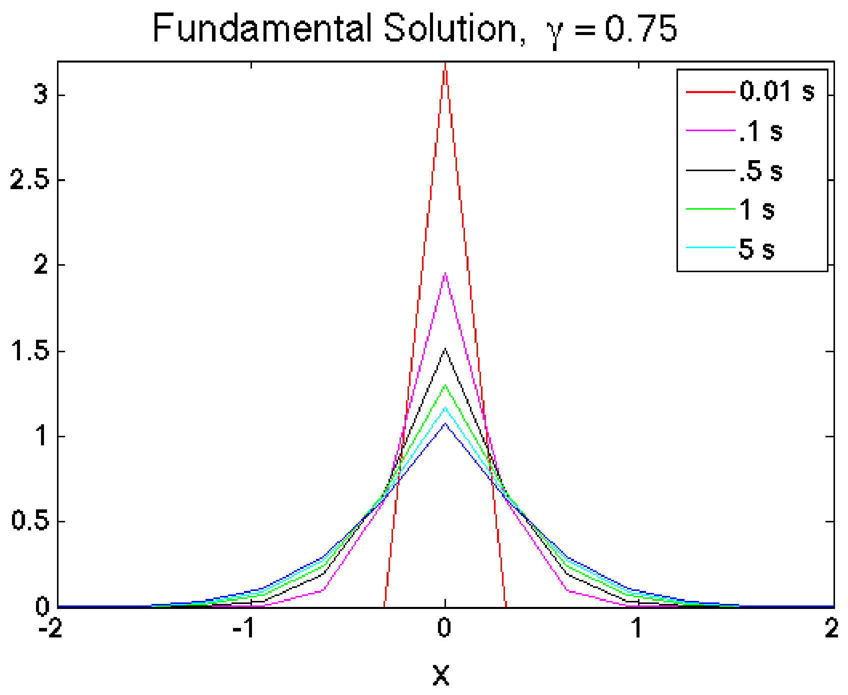


Figure 2.4.1: The fundamental solution for a set $\gamma = 0.75$, at various times. It is clear that the peaked center of the diffusion profile persists even after a considerable amount of time has passed.

Figs. 2.4.1 and 2.4.2 show the fundamental solution for various fractional order, and at various times. It is clear that anomalous diffusion for the specific initial conditions represented by the fundamental solution, has a characteristic peaked profile that is distinct from the classical Gaussian diffusion profile. In addition, the lower the fractional order γ (or, correspondingly, the higher the degree of anomalous diffusion), the longer the peaked profile persists and the more slowly material diffuses away from the initial concentration profile. We use the fundamental solution as a basis for estimating error of the one-dimensional versions of several of our algorithms, and we will revisit this solution in Chapter 4.

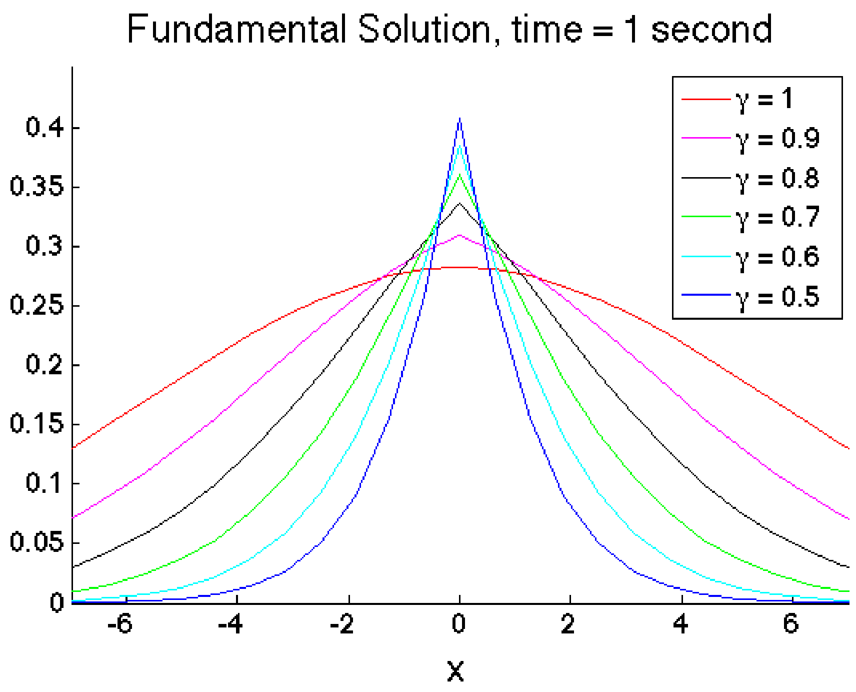


Figure 2.4.2: The fundamental solution for a set time = 1 second, for various γ values. It is clear that the smaller the value of γ, the more pointed and narrow the diffusion profile. Or, in the other words, the longer material stays near the point of origin and slower it diffuses away. Again, we see a persistent peaked nature of the profile, except for $\gamma = 1$, which of course corresponds to Gaussian diffusion.

Chapter 3

The Fractional Diffusion Equation and Adaptive Time Step Memory

3.1 Introduction

Here we investigate the numerical implementation and computational performance of a fractional reaction-diffusion equation. The continuous diffusion equation has been the focal point of transport modeling in physical and biological systems for over a hundred years, since first proposed by Adolf Fick in 1855. The practical motivation for the present work arose from considering cell signaling data between Muller neural glial cells in the neural sensory retina. The neural retina is a direct extension and part of the brain itself. Muller cells in the retina can communicate in a paracrine fashion by secreting adenosine triphosphate (ATP) that diffuses into the extracellular space that then triggers intracellular calcium waves. In some cases these signaling events can produce long range regenerative intercellular signaling in networks of cells; see [38, 35, 56]. In particular, from [38], we observed a peaked diffusion profile for ATP that persisted over long time periods; this is a typical characteristic of anomalous diffusion, and we suspected that the diffusion of ATP in this physiological system was subdiffusive. Modeling subdiffusive neurophysiological transport processes necessitates the use of fractional differential equations. However, the numerical implementation of such equations involves non-local fractional derivative operators that require taking into account the entire past history of the system in order to compute the state of the system at the current time step. This produces a significant computational burden that may limit the practicality of the models. In the present paper we introduce an ‘adaptive time step’ approach for computing the contribution of the memory effect associated with the history of a system in a way that makes it both numerically and resource (i.e. computer memory) efficient and robust. While our algorithms can be applied to any diffusion application modeled as a fractional process, they offer particular value for modeling complex processes with long histories, such as lengthy molecular or cellular simulations.

Approaches for numerically approximating solutions to fractional diffusion equations have been extensively studied [6, 18, 22, 32, 34, 41, 57], but in general there is always a trade off between computational efficiency, complexity, and the accuracy of the resultant approximations. There has also been a considerable amount of work done on developing fast convolution quadrature algorithms, which is relevant to fractional differential equations because the non-local nature of fractional calculus operators results in either a continuous or discrete convolution of some form. In particular, Lubich and others [26, 24, 25, 27, 28, 45, 23] have built a generalized and broadly applicable convolution quadrature framework to numerically approximate a continuous convolution integral with a wide array of possible convolution kernel functions (including oscillatory kernels, singular kernels, kernels with multiple time scales, and unknown kernels with known Laplace transforms), while achieving good algorithmic performance with respect to complexity and memory requirements. However, their algorithms are necessarily very complicated in order to handle a wide range of functions and expressions in the convolution while limiting storage requirements and scaling of arithmetic operations. They involve approximating a continuous convolution based on numerical inversion of the Laplace transform of the convolution kernel function using contour integrals discretized along appropriately chosen Talbot contours or hyperbolas. The scaling of the number of required arithmetic operations involved in the convolution, and overall memory requirements, are both reduced by splitting up the continuous convolution integral or discrete convolution quadrature summation into a carefully chosen series of smaller integrals or summations, and solving a set of ordinary differential equations one time step at a time without storing the entire history of the solutions. For methods explicitly involving a discrete convolution summation, the quadrature weights are calculated using the Laplace transform of the original kernel and linear multistep methods for solving differential equations. FFT techniques can be applied to calculate these weights simultaneously and further increase the efficiency of the algorithm [25]. These authors have demonstrated the success of their framework by simulating various differential equations involving convolutions, including a one-dimensional fractional diffusion equation [45]. In [19], Li takes an alternative approach from this framework and focuses on a fast time-stepping algorithm specifically for fractional integrals by constructing an efficient Q point quadrature, but does not focus on how this fits into numerical algorithms representing larger mathematical models.

The methods we introduce here are also very efficient, significantly reducing simulation times, computational overhead, and required memory resources, without significantly affecting the computed accuracy of the final solution. However, in contrast to broad generalized frameworks, our algorithms are focused on solving fractional differential equations involving nonlocal fractional derivative operators. We approximate the fractional derivative based on the Grünwald-Letnikov definition instead of pursuing a general quadrature formula approximation to the Riemann-Liouville integral definition of the fractional derivative. The Grünwald-Letnikov derivative is used in many numerical schemes for discretizing fractional diffusion equations from a continuous Riemann-Liouville approach [7, 57, 40], and involves a discrete convolution between a ‘weight or coefficient function and the function for which we are interested in taking the derivative. The mathematics and theory of this form of a weighting function are well established in the literature [40, 26]. Building on this foundation avoids the need for domain transformations, contour integration or involved theory. Our algorithms were specifically developed for real world applied diffusion and transport physics and engineering problems, often where there are practical measurement limitations to the types and quality (e.g. granularity) of the data that can be collected or observed. For situations such as these, our methods for handling discrete convolutions associated with fractional derivatives are more intuitive and accessible than other generalized mathematical methods, where it is often not obvious or clear how to implement and apply a generalized approach to a specific physical problem under a defined set of constraints. The approaches we introduce here can be incorporated with various combinations of finite difference spatial discretizations and time marching schemes of a larger mathematical model, in a straightforward way.

The increased efficiency and memory savings in the approaches we describe here lie in the way the discrete summation is calculated. The conceptual basis that results in a saving of computational time without a huge tradeoff in accuracy is in the interpretation of the weight function as a measure of the importance of the history of a system. The more recent history of the system is more important in determining the future state of the system, and therefore we make use of an ‘adaptive time step memory’ method by changing the interval of the backwards time summation in the Grünwald-Letnikov fractional derivative. Instead of incorporating every previous time point into the most current calculation, only certain points are recalled based upon their proximity to the current point in time, weighted according to the sparsity of the time points. This substantially reduces the computational overhead needed to recalculate the prior history at every time step, yet maintains a high level of accuracy compared to other approximation methods that make use of the Grünwald-Letnikov definition. We consider two adaptive step approaches - one based on an arithmetic sequence and another based on a power law, that when combined with a linked-list computational data structure approach yield $O(log_2(N))$ active memory requirements. This is a significant improvement over keeping all N steps of the system's history. We compare our adaptive time step approach with the 'short memory' method described by Volterra ([52]) and Podlubny et al [40], and examine differences in simulation times and errors under similar conditions for both. In the last section we sketch out a 'smart' adaptive memory method based on a continuous form of the Grünwald-Letnikov that will be able to accommodate more dynamic past histories, i.e., histories with sharp and abrupt changes relative to the time step being considered, that produce larger error in the discrete methods we discuss.

3.2 The Grünwald-Letnikov Derivative

3.2.1 Derivation of the Fractional Diffusion Equation

We begin by considering the standard diffusion equation with no reaction term

$$\frac{\partial u}{\partial t} = \alpha \nabla^2 u, \tag{3.2.1}$$

where α is the diffusion coefficient. The time-fractional version of equation 3.2.1 is given by

$$\frac{\partial^\gamma u}{\partial t^\gamma} = \alpha \nabla^2 u. \tag{3.2.2}$$

Here α has units $\frac{spatial\ unit^2}{time\ unit^\gamma}$. This is the simplest form of the fractional diffusion equation. The real values taken by γ in equation 3.2.2 dictate the diffusion regimes obtained by the equation. Values of $0 < \gamma < 1$ result in subdiffusion, characterized by a slow diffusion profile relative to regular diffusion. When $\gamma > 1$, an oscillatory component emerges in the diffusion profile resulting in superdiffusion. As required, when $\gamma = 1$ the fractional diffusion equation reduces to standard diffusion that results in a Gaussian profile. When $\gamma = 2$ it reduces to the standard wave equation. Using the relation

$$\frac{\partial u}{\partial t} = \frac{\partial^{1-\gamma}}{\partial t^{1-\gamma}} \frac{\partial^\gamma u}{\partial t^\gamma} \tag{3.2.3}$$

and substituting into equation 3.2.2 yields

$$\frac{\partial u}{\partial t} = \frac{\partial^{1-\gamma}}{\partial t^{1-\gamma}} \alpha \nabla^2 u \tag{3.2.4a}$$

$$\frac{\partial u}{\partial t} = \alpha D^{1-\gamma} \nabla^2 u \tag{3.2.4b}$$

where $D^{1-\gamma}$ denotes the fractional derivative operator which we discuss at length in the next section.

If we take into account consumption or generation processes, then we can rewrite the diffusion equation as

$$\frac{\partial u(\vec{x},t)}{\partial t} = \alpha D^{1-\gamma} \nabla^2 u(\vec{x},t) + f(u), \quad \vec{x} = \{x_1, x_2 ... x_N\} \tag{3.2.5}$$

where $\vec{x}$ is an N dimensional vector and *f(u)* represents a consumption or generation term. We now consider the fractional derivative operator $D^{1-\gamma}$.

3.2.2 Definition of the Grünwald-Letnikov Derivative

The a^{th} order fractional derivative of a function, denoted by $D^a f$, extends the order of the differential operator from the set of integers to the set of real numbers. The operator can be mathematically defined in several ways, including standard descriptions like the Riemann-Liouville definition, the Grünwald-Letnikov definition, and others. For numerical simulations, we find the Grünwald-Letnikov derivative to be convenient, since it is based on the standard differential operator but made applicable to arbitrary order a with a discrete summation and binomial coefficient term:

$$D^a f(x) = \lim_{h\to 0} h^{-a} \sum_{m=0}^{x/h} (-1)^m \binom{a}{m} f(x-mh). \tag{3.2.6}$$

Expanding the binomial coefficient yields

$$\binom{a}{m} = \frac{a!}{m!(n-m)!} \tag{3.2.7}$$

and expressing the factorial components of the binomial coefficient by the gamma function gives

$$\binom{a}{m} = \frac{\Gamma(a+1)}{m!\Gamma(a+1-m)}. \tag{3.2.8}$$

Combining equations 3.2.6 and 3.2.8 yields the Grünwald-Letnikov derivative for real numbers:

$$D^a f(x) = \lim_{h\to 0} h^{-a} \sum_{m=0}^{x/h} \frac{(-1)^m \Gamma(a+1)}{m!\Gamma(a+1-m)} f(x-mh) \quad a \in \mathbb{R},\ a \neq -\mathbb{N}_1. \tag{3.2.9}$$

It should be noted that computation of the Grünwald-Letnikov derivative requires knowledge of the entire past history of the system due to the summation operator. This means that this fractional derivative, unlike standard derivatives, is no longer local but global. Computing the value of a real derivative requires knowledge of the system's past history. The derivative must take into account all past points from $m = 0$, the current point, all the way back to the beginning point $m = x/h$. This requirement places significant computational demands on the numerical implementation of the Grünwald-Letnikov derivative, and is the principle technical motivation for the results presented here.

In the rest of the chapter we explore the implementation of equation 3.2.9 to a diffusion regime constrained to $0 < \gamma \equiv a \leq 2$ (as explained in section 3.2.1):

$$D^{\gamma} f(x) = \lim_{h \to 0} h^{-\gamma} \sum_{m=0}^{x/h} \frac{(-1)^m \Gamma(\gamma+1)}{m! \Gamma(\gamma+1-m)} f(x-mh) \text{ for } 0 < \gamma \leq 2. \tag{3.2.10}$$

Applying equation 3.2.10 to the fractional diffusion equation (Eq. 3.2.4b) yields:

$$\frac{\partial u}{\partial t} = \lim_{\tau \to 0} \tau^{\gamma-1} \alpha \sum_{m=0}^{t/\tau} \frac{(-1)^m \Gamma(2-\gamma)}{m! \Gamma(2-\gamma-m)} \nabla^2 u(t-m\tau) \tag{3.2.11}$$

where t represents instantaneous time, and τ is the time step.

Next, define a function $\psi(\gamma, m)$ such that

$$\psi(\gamma, m) = \frac{(-1)^m \Gamma(2-\gamma)}{m! \Gamma(2-\gamma-m)}. \tag{3.2.12}$$

Given that the standard definition of the gamma function implies $\Gamma(a) = \frac{1}{a}\Gamma(a+1)$, equation 3.2.12 can be reduced to a recursive relationship. Consider the first four terms in the chain from $m = 0$ to $m = 3$:

$$\begin{aligned}
\psi(\gamma,0) &= \frac{(-1)^0 \cdot \Gamma(2-\gamma)}{(0!) \cdot \Gamma(2-\gamma-0)} = 1 \\
\psi(\gamma,1) &= \frac{(-1)^1 \cdot \Gamma(2-\gamma)}{(1!) \cdot \Gamma(2-\gamma-1)} = \frac{(-1) \cdot (-1)^0 \cdot \Gamma(2-\gamma)}{(1) \cdot (0!) \cdot \frac{1}{(2-\gamma-1)} \cdot \Gamma(2-\gamma-0)} \\
\psi(\gamma,2) &= \frac{(-1)^2 \cdot \Gamma(2-\gamma)}{(2!) \cdot \Gamma(2-\gamma-2)} = \frac{(-1) \cdot (-1)^1 \cdot \Gamma(2-\gamma)}{(2) \cdot (1!) \cdot \frac{1}{(2-\gamma-2)} \cdot \Gamma(2-\gamma-1)} \\
\psi(\gamma,3) &= \frac{(-1)^3 \cdot \Gamma(2-\gamma)}{(3!) \cdot \Gamma(2-\gamma-3)} = \frac{(-1) \cdot (-1)^2 \cdot \Gamma(2-\gamma)}{(3) \cdot (2!) \cdot \frac{1}{(2-\gamma-3)} \cdot \Gamma(2-\gamma-2)}
\end{aligned}$$

Examining this system of equations suggests that for any $\psi(\gamma,m)$:

$$\psi(\gamma,m) = \frac{(-1)^m \cdot \Gamma(2-\gamma)}{(m!) \cdot \Gamma(2-\gamma-m)} = \frac{(-1) \cdot (-1)^{(m-1)} \cdot \Gamma(2-\gamma)}{(m) \cdot (m-1)! \cdot \frac{1}{(2-\gamma-m)} \cdot \Gamma(2-\gamma-(m-1))}. \quad (3.2.13)$$

Because the function $\psi(\gamma,m)$ is dependent on $\psi(\gamma,m-1)$, an iterative relationship forms that scales by $\frac{-(2-\gamma-m)}{m}$:

$$\psi(\gamma,m) = -\psi(\gamma,m-1)\frac{2-\gamma-m}{m}. \quad (3.2.14)$$

This recursive function is valid for all γ including subdiffusion, standard diffusion and superdiffusion, so this equation is general over all regimes. Because the entire history of the system must be taken into account when computing the current time step, $\psi(\gamma,m)$ is used for many m values, many times over the course of the simulation, and can be precomputed for values of $m = 0$ to $m = N$, where N is the total number of time points, resulting in a significant savings in computational performance. A similar simplification has been previously discussed and used in [40]. Taking $\psi(\gamma, m)$ into account yields the final form of the fractional diffusion equation, given by

$$\frac{\partial u}{\partial t} = \lim_{\tau \to 0} \tau^{\gamma-1} \alpha \sum_{m=0}^{t/\tau} \psi(\gamma, m) \nabla^2 u(t - m\tau). \tag{3.2.15}$$

We can also arrive at equation 3.2.15 by an alternative approach. Begin by considering again the Grünwald-Letnikov derivative in terms of binomial notation:

$$D^{1-\gamma} f(t) = \lim_{\tau \to 0} \tau^{\gamma-1} \sum_{m=0}^{t/\tau} (-1)^m \binom{1-\gamma}{m} f(t - m\tau). \tag{3.2.16}$$

Applying Eq. 3.2.16 to the fractional diffusion equation (Eq. 3.2.4b) yields

$$\frac{\partial u}{\partial t} = \lim_{\tau \to 0} \tau^{\gamma-1} \alpha \sum_{m=0}^{t/\tau} (-1)^m \binom{1-\gamma}{m} \nabla^2 u(\vec{x}, t - m\tau) \tag{3.2.17}$$

Knowing the gamma function definition of the factorial operator, $a! = \Gamma(a+1)$, the binomial coefficient can be expanded to accommodate real numbers (also *c.f.* equation 3.2.8):

$$\binom{1-\gamma}{m} = \frac{(1-\gamma)!}{m!(1-\gamma-m)!} = \frac{\Gamma(2-\gamma)}{m!\Gamma(2-\gamma-m)}. \tag{3.2.18}$$

Note how combining equations 3.2.17 and 3.2.18 yields equation 3.2.11 as required.

Next, defining the function $\psi(\gamma, m)$ (equation 3.2.12) using binomial notation gives

$$\psi(\gamma, m) = (-1)^m \binom{1-\gamma}{m} \tag{3.2.19}$$

and substituting the relation $\Gamma(a) = (a-1)\Gamma(a-1)$ into Eq. 3.2.18 results in

$$\psi(\gamma,m) = \frac{(-1)^{m-1}\Gamma(2-\gamma)}{(m-1)!\Gamma(2-\gamma-(m-1))}\frac{-1}{m\frac{1}{2-\gamma-m}} \tag{3.2.20}$$

yielding the iterative relationship in equation 3.2.14 above:

$$\psi(\gamma,m) = -\psi(\gamma,m-1)\frac{2-\gamma-m}{m}.$$

With the substitution of $\psi(\gamma,m)$ into eq. 3.2.17 we once again recover the time-fractional diffusion equation in the form

$$\frac{\partial u}{\partial t} = \lim_{\tau\to 0}\tau^{\gamma-1}\alpha\sum_{m=0}^{t/\tau}\psi(\gamma,m)\nabla^2 u(\vec{x},t-m\tau). \tag{3.2.21}$$

3.2.3 Discretization of the Fractional Diffusion Equation

Using

$$\nabla^2 u(\vec{x},t-m\tau) = \frac{\partial^2 u(\vec{x},t-m\tau)}{\partial x_1^2} + \frac{\partial^2 u(\vec{x},t-m\tau)}{\partial x_2^2} \tag{3.2.22}$$

as the formulation in two spatial dimensions, one can discretize the function into a finite difference based FTCS scheme (forward time centered space) on a grid $u^n_{j,l}$ (where $n = t/\Delta_t, j = x_1/\Delta_x, l = x_2/\Delta_x$, Δ_x is the grid spacing in both directions assuming an equally spaced grid, and Δ_t is the time step), using the relations ([42])

$$\begin{aligned}
\frac{\partial^2 u(\vec{x},t-m\tau)}{\partial x_1^2} &= \frac{u^{n-m}_{j+1,l} - 2u^{n-m}_{j,l} + u^{n-m}_{j-1,l}}{\Delta_x^2} \\
\frac{\partial^2 u(\vec{x},t-m\tau)}{\partial x_2^2} &= \frac{u^{n-m}_{j,l+1} - 2u^{n-m}_{j,l} + u^{n-m}_{j,l-1}}{\Delta_x^2} \\
\frac{\partial u(\vec{x},t)}{\partial t} &= \frac{u^{n+1}_{j,l} - u^{n}_{j,l}}{\Delta_t}.
\end{aligned} \tag{3.2.23}$$

In the discrete limit where $\tau \to \Delta_t$, we approximate the fractional diffusion equation (equations 3.2.15 and 3.2.21) with the following finite difference expression, which has a first-order approximation $O(\tau)$ (Chapter 7 in [40], [26]):

$$\frac{u_{j,l}^{n+1}-u_{j,l}^{n}}{\Delta_t}=\alpha\frac{\Delta_t^{\gamma-1}}{\Delta_x^2}\sum_{m=0}^{n}\psi(\gamma,m)\delta_{j,l}^{n-m} \tag{3.2.24}$$

where $\delta_{j,l}^{n-m}$ is the finite difference kernel given by

$$\delta_{j,l}^{n-m}=\left(u_{j+1,l}^{n-m}+u_{j-1,l}^{n-m}-4u_{j,l}^{n-m}+u_{j,l+1}^{n-m}+u_{j,l-1}^{n-m}\right). \tag{3.2.25}$$

Adding a consumption/generation term is straightforward in this implementation. For example, take an exponential decay term given by

$$\frac{\partial u}{\partial t}=-\beta u \tag{3.2.26}$$

with the complementary finite difference relation

$$\frac{u_{j,l}^{n+1}-u_{j,l}^{n}}{\Delta_t}=-\beta u_{j,l}^{n}. \tag{3.2.27}$$

Incorporating Eq. 3.2.26 into the form of eq. 3.2.5 results in

$$\frac{\partial u}{\partial t}=\alpha D_t^{1-\gamma}\nabla^2 u-\beta u, \tag{3.2.28}$$

which gives the full finite difference implementation in two dimensions

$$\frac{u_{j,l}^{n+1}-u_{j,l}^{n}}{\Delta_t}=\alpha\frac{\Delta_t^{\gamma-1}}{\Delta_x^2}\sum_{m=0}^{n}\psi(\gamma,m)\delta_{j,l}^{n-m}-\beta u_{j,l}^{n} \tag{3.2.29}$$

Figure 3.2.1 shows the results of four simulations for different values of γ. Simulations were run on a 100 x 100 grid with initial conditions $U_{50,50}^{0}$ = 0.1, $U_{51,50}^{0}=U_{50,51}^{0}=U_{49,50}^{0}=U_{50,49}^{0}$ = 0.05 and zero elsewhere. Dirichlet boundary conditions were

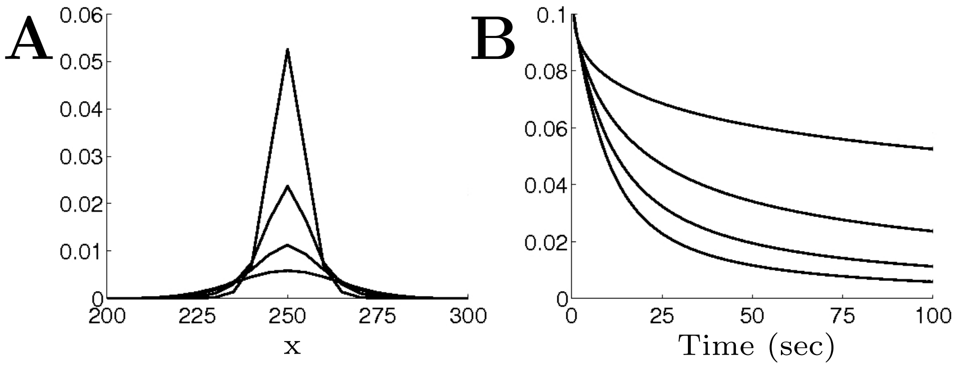


Figure 3.2.1: Simulation results for $\gamma = 0.5,\ 0.75,\ 0.9,\ 1.0$ (for traces from top to bottom) in one dimensional space (panel A) and time (panel B). As the subdiffusion regime is approached the profile becomes more and more hypergaussian.

implemented and the simulation edge set to zero. Simulations were run for $t = 100$ seconds with $\alpha = 1$, $\beta = 0$, and $\Delta_x = 5$.

3.3 Adaptive Time Step Memory as an Arithmetic Sequence

3.3.1 Derivation

In Eq. 3.2.29 each iteration requires the re-calculation and summation of every previous time point convolved with $\psi(\gamma, m)$. This becomes increasingly cumbersome for large times, which require significant numbers of computations and memory storage requirements. To address this, Podlubny et. al ([40]) introduced the 'short memory' principle which assumes that for a large t the role of the previous steps or 'history' of the equation become less and less important as the convolved $\psi(\gamma, m)$ shrinks towards zero. This would then result in approximating eq. 3.2.29 by truncating the backwards summation, only taking into account times on the interval $[t - L, t]$ instead of $[0, t]$, where L is defined as the 'memory length' (eq. 7.4 in [40]; Fig. 3.3.1). While computationally efficient, this approach leads to errors in the final solution since not all points are counted in the summation. Despite the resultant errors, this numerical method represents a powerful and simple approach for providing a reasonable trade off between computational overhead and numerical accuracy. In the context of the implementation derived here, it would result in a discretization scheme given by

$$\frac{u_{j,l}^{n+1} - u_{j,l}^{n}}{\Delta_t} = \alpha \frac{\Delta_t^{\gamma-1}}{\Delta_x^2} \sum_{m=0}^{min(L/\Delta_t,n)} \psi(\gamma,m)\delta_{j,l}^{n-m} - \beta u_{j,l}^{n} \tag{3.3.1}$$

As an alternative to the method of Podlubny, Ford and Simpson proposed an alternative ‘nested mesh’ variant that gives a good approximation to the true solution at a reasonable computational cost ([12]). However, this method is exclusive to the Caputo fractional derivative. Here we introduce an approach that is applicable to the more general Grünwald-Letnikov definition. Like these other methods, it also shortens computational times but at the same time results in much greater accuracy than the use of ‘short memory.’ We achieve this by introducing the concept of an ‘adaptive memory’ into the Grünwald-Letnikov discretization.

The underlying principle of the adaptive time approach is that relative to the current time point previous time points contribute different amounts to the summation. Values relatively closer to the current time point will have a greater contribution to the current numerical calculation than values many time points back due to the multiplier $\psi(\gamma,m)$. For smooth functions, as m increases and $|\psi(\gamma,m)|$ decreases, neighboring points in the summation exhibit only small differences. Consequently, one can take advantage of this and utilize an ‘adaptive memory’ approach in which neighboring values at prior time points are grouped together in defined increments and the median taken as a representative contribution for the increment weighted according to the length of the increment to account for the skipped points. This results in fewer time points that need to be computed in a summation. Algorithmically, for an arbitrary a time steps back from the current time point k for which the history of the system is being computed, consider an initial interval $[0,a]$ for which all time points within this interval are used in the summation and therefore contribute to the the Grünwald-Letnikov discretization. Let subsequent time intervals become longer the further away from n they are and within them only every other d time points are used in the summation term, i.e., only the median values of a temporally shifting calculation increment of length d along the current interval are considered. As subsequent intervals become longer, so does d, thereby requiring less points to compute (Fig. 3.3.1).

Definition 3.3.1. Let n be the current iterative time step in a reaction diffusion process for which a Grünwald-Letnikov discretization is being computed. Consider an arbitrary time point a in the history of the system backwards from n. For $i \in \mathbb{N}_1, i \neq 1$, define an interval of this past history by

$$I = [a^{i-1} + i, a^i] \tag{3.3.2}$$

where $\mathbb{N}_1$ represents the set of natural numbers beginning from one. Given how the indices i are defined, the very first interval backwards from n is independent of equation 3.3.2 and is given by $[0,a]$. This is considered the base interval. Subsequent intervals are defined as a function of this base, i.e., as a function of a and determined by eq. 3.3.2. Let i_{max} be the value of i such that $n \in I_{max} = [a^{i_{max}-1} + i_{max}, a^{i_{max}}]$. The complete set of intervals then is defined as $\zeta = \{I = [a^{i-1} + i, a^i] : i \in \mathbb{N}_1, i \neq 1, i \leq i_{max}\}$.

Definition 3.3.2. For the set of intervals ζ as defined in Definition 3.1, $D = \{d = 2i - 1 : i \in \mathbb{N}_1, i \neq 1\}$ is the set of distances d by which the corresponding intervals in ζ are sampled at.

Theorem 3.3.3. *In general, for two-dimensional diffusion without consumption or generation terms for any interval as defined in Definition 3.1, the Grünwald-Letnikov* *discretization with adaptive time step memory is given by*

$$\frac{u_{j,l}^{n+1} - u_{j,l}^{n}}{\Delta_t} = \alpha \frac{\Delta_t^{\gamma-1}}{\Delta_x^2} \left[\sum_{m=0}^{a} \psi(\gamma, m) \delta_{j,l}^{n-m} + \cdots \right. \tag{3.3.3}$$

$$\left. \sum_{i=2}^{i_{max}} \sum_{m_i=a^{i-1}+i}^{a^i} (2i-1)\psi(\gamma, m_i)\delta_{j,l}^{n-m_i} + \sum_{p=m_{max}+i_{max}}^{n} \psi(\gamma, p)\delta_{j,l}^{n-p} \right]$$

where $p \in \mathbb{N}_1$, and for each i (i.e. for each interval) $M = \{m_i = a^{i-1} + (2i-1)\eta - i + 1 : \eta \in \mathbb{N}_1 \ \& \ m_i \leq m_{max}\}$ is the set of time points over which $\psi(\gamma, m)\delta_{j,l}^{n-m}$ is evaluated. Since the time point n may be less than the full length of the last interval I_{max}, $|m_{max}| \leq |n - i_{max}|$ represents the maximum value in I_{max} that is evaluated, i.e. the last element in the set M for I_{max}.

Proof. The first summation represents the basis interval and the limits of the summation operator imply the contribution of every time point, i.e., $m \in \mathbb{N}_1$. For intervals beyond a: any arithmetic sequence defined by a recursive process $\nu_\eta = \nu_{\eta-1} + d$, $\eta \in \mathbb{N}_1$ for some distance d, the η^{th} value in the sequence can be explicitly calculated as $\nu_\eta = \nu_1 + (\eta - 1)d$ given knowledge of the sequence's starting value ν_1. For the set ζ this implies that $\nu_1 = a^{i-1} + i$ and $d = 2i - 1$ for a given i. This then yields $\nu_\eta = a^{i-1} + i + (\eta - 1)(2i - 1) = a^{i-1} + (2i-1)\eta - i + 1 := m_i$ as required. The outer summation collects the summations of all intervals that belong to the set ζ. The last summation on the interval $[m_{max} + i_{max}, n]$ ensures that the final point(s) of the backwards summation are still incorporated into the computation even if the arbitrarily chosen value of a does not generate a final increment length that ends on n. $\square$

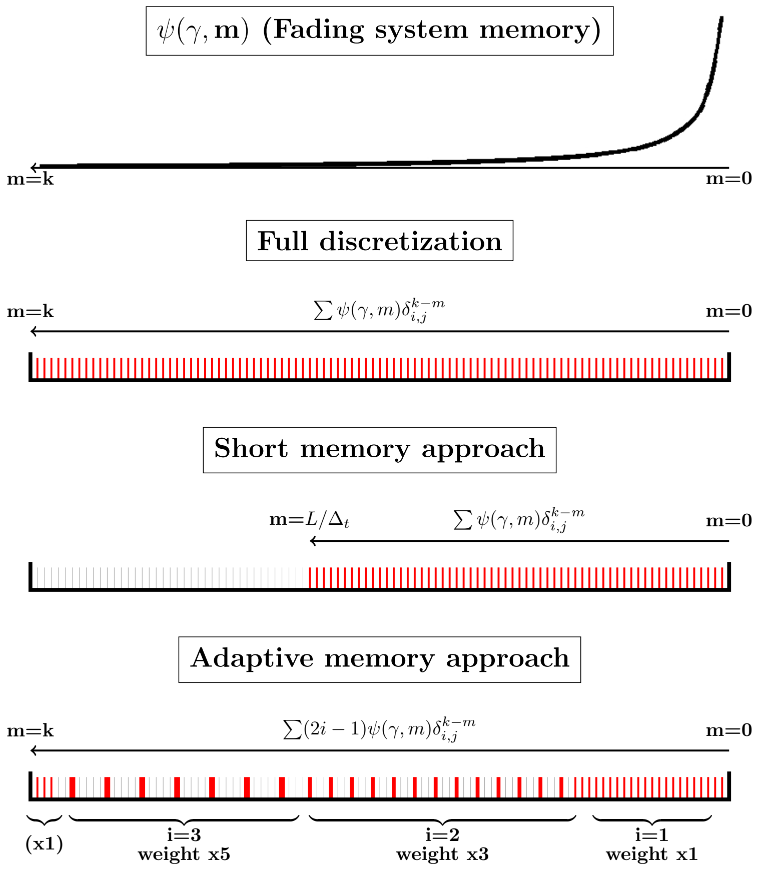
Note that D is not explicitly needed for computing equation 3.3.3 because the distances d are implicitly taken into account by ζ. Using the median value of each increment avoids the need for interpolation between time points. The implementation of the adaptive memory method described here is necessarily limited to smooth functions due to the assumption that the neighbors of the median values used in the summation do not vary much over the increment being considered. This method essentially allows for a contribution by all previous time points to the current calculation, yet reduces computational times by minimizing the total number of calculations.

3.3.2 Comparison to Short Memory Methods

The results of using various L (for short memory) and interval steps a (for adaptive memory) are shown in Fig. 3.3.2. Increasing the values of L and a resulted in a decrease in the error of the estimated results but at the cost of increased computation times. Conversely, decreasing the values of L and a resulted in a decrease in computation times, but at the expense of accuracy. In all cases however, the adaptive memory method had a significantly lower error for the same simulation time, and also reached a clear asymptotic minimum error much faster than the minimum error achieved by the short memory method. In these simulations, $\alpha = 1$, $\beta = 0$, $\Delta_t = 1$, $\Delta_x = 10$, using a 20 x 20 grid, and ran for $t = 1500$ where $U^0_{10,10} = 10$. The error for the 'short memory' method increased comparatively quickly and worsened as $\gamma \to 1$. This was due to the fact that the evolution of the solution was initially quite fast at times near $t = 0$, which were the first time points cut by the 'short memory' algorithm.

3.4 Adaptive Time Step as a Power Law

While computationally efficient and intuitively straightforward, the major drawback to the adaptive memory method computed as an arithmetic sequence, is the necessary amount of allocated memory. The backwards time summation grid changes with every step such that every point in the history needs to be retained and stored during the course of a simulation. For high dimensional diffusion this limits the practical applicability of



$\psi(\gamma, \mathrm{m})$ (Fading system memory)
m=k
m=0
Full discretization
m=k
$\sum \psi(\gamma, m)\delta_{i,j}^{k-m}$
m=0
Short memory approach
m=L/Δ_t
$\sum \psi(\gamma, m)\delta_{i,j}^{k-m}$
m=0
Adaptive memory approach
m=k
$\sum (2i-1)\psi(\gamma, m)\delta_{i,j}^{k-m}$
m=0
(x1)
i=3
weight x5
i=2
weight x3
i=1
weight x1

Figure 3.3.1: Short memory and adaptive memory methods for estimating the the Grünwald-Letnikov discretization. Both approximations rely on the sharply decreasing function $|\psi(\gamma, m)|$ as m increases to omit points from the backwards summation. While short memory defines a sharp cut off of points, adaptive memory provides a weighted sampling of points for the entire history of the system. Points included in the computation by each method are highlighted in red. The shape of ψ is different for $\gamma < 1$ and $\gamma > 1$, but the shape of $|\psi|$ remains a monotonically decreasing function (as m increases) for both cases, and remains consistent with the principle that more recent time points contribute (whether positively or negatively) more to the solution at the next time step, than time points further back in the history of the system. See text for details.

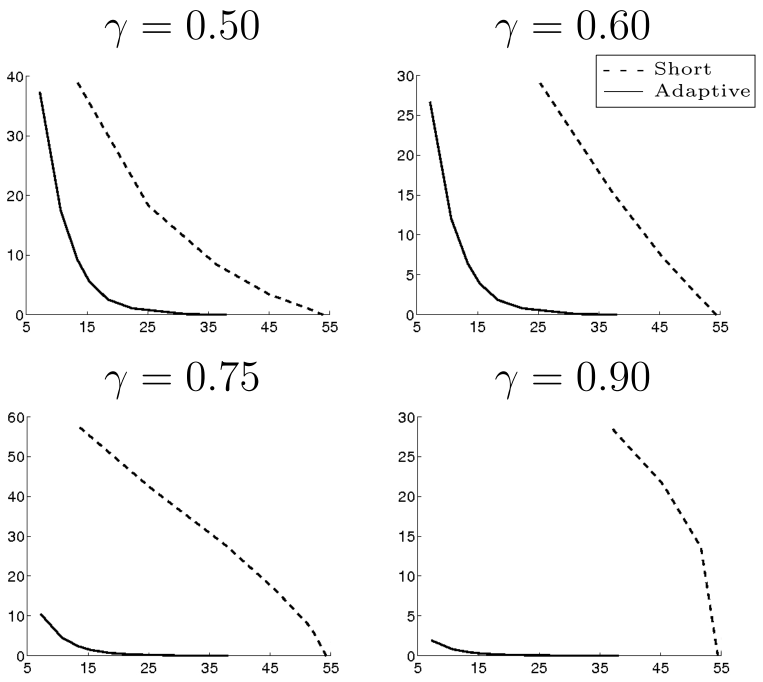


Figure 3.3.2: Comparison of the error between adaptive memory and the short memory as a function of the calculation time (x-axis: computation run time in seconds) expressed as a percentage error relative to the computed value for the full non-shortened approach (y-axis). Four different values of γ are shown.

the method. For example, if one were to solve the three dimensional diffusion equation on just a 100 x 100 x 100 set of grid points, that would require the storage of 1,000,000 data points per time step, which over the course of a long simulation would overwhelm the memory capacity of most computers in addition to greatly slowing simulation times. The same memory issues arise when considering another popular approach used for solving the fractional diffusion equation, discussed in [41], that uses an implicit matrix solution to simultaneously solve all points in time using a triangular strip matrix formulation. This later approach is very powerful but does not take advantage of any short memory algorithms. In this section we improve on our results and develop a framework that uses a power law-based rule for eliminating past history points without sacrificing numerical accuracy. This approach, in combination with a linked-list based method, minimizes the storage of the past history of the function. This results in decreasing the amount of total memory allocated to run a simulation of N time steps from $O(N)$ to $O(log_2(N))$, resulting in a tremendous memory savings.

Given a simulation of N time steps and number of grid points X/Δ_x, where X is the grid width, storing the entire past history would require $\frac{X}{\Delta_x}N$ points to be stored, which grows linearly with N. On the other hand, an adaptive mesh with a power law growth in spacing (in this case $1, 2, 4...$), results in $\frac{X}{\Delta_x}log_2(N)$ points being stored in memory, which grows with the log of the number of points N. The advantage of using such a power law scaling is that one can *a priori* calculate memory points which will not be used at every time step of the simulation and de-allocate the memory assigned to storing those time points. With the adaptive memory method, past points needed for each subsequent time step change with each iteration, necessitating that all time points be stored for future calculations. The use of a self similar power law allows the elimination of points that will not be needed at any future time in the calculation. A comparison of the implementation of this method with the full implementation using every time point,

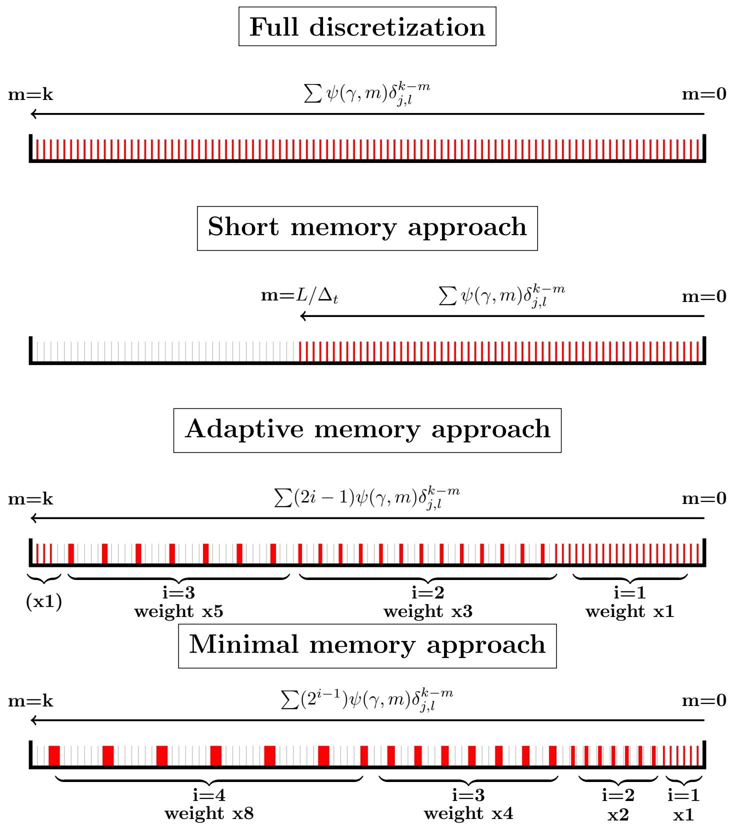


Figure 3.4.1: Comparison of the full discretization, short memory approach, and adaptive memory approach, to the minimal memory implementation.

is shown in Figure 3.4.1.

3.4.1 Set Theoretic Implementation

In one spatial dimension, an FTCS discretization of the fractional diffusion equation on a grid u_j^n where $n = t/\Delta_t, j = x_1/\Delta_x$, can be written as

$$\frac{u_j^{n+1} - u_j^n}{\Delta_t} = \alpha \frac{\Delta_t^{\gamma-1}}{\Delta_x^2} \sum_{m=0}^{i} \psi(\gamma, m)\delta_j^{n-m}. \tag{3.4.1}$$

(*c.f.* equations 3.2.23 to 3.2.25 above and [42]), where

$$\delta_j^n = \left(u_{j+1}^n - 2u_j^n + u_{j-1}^n\right) \tag{3.4.2}$$

Definition 3.4.1. Define a well-ordered set U in time consisting of elements u_j^i ordered by the point in time $i = t/\Delta_t$ at which that grid point occurred. The least elements in this set are then the points where u_j^0, and the greatest are the points u_j^n, where the current time point is $n = t_{current}/\Delta_t$. For all integers A and B, if $A > B$, then $u_j^A > u_j^B$.

Given the numerical scheme in 3.4.1, the set U can be expressed as the recursive algorithm

$$u_j^{n+1} = u_j^n + \alpha\frac{\Delta_t^\gamma}{\Delta_x^2}\sum_{\{u_j^i \in U\}} \psi(\gamma, n-i)\delta_j^i. \tag{3.4.3}$$

where u_j^0 is the set of initial conditions at $t = 0$.

Taking advantage of this result we can state the following lemma for the short memory approach:

Lemma 3.4.2. *Assume a short memory scheme. The elements $u^i < u^{n-\frac{L}{\Delta_t}}$ in U for a time step i are not used in future computations and can be removed from U.*

The set U can then become a list of sets U^i with only the necessary elements to complete the recursive relation in each step i.

Proof. The short memory approach shown in Figure 3.4.1 can be written as

$$u_j^{n+1} = u_j^n + \alpha\frac{\Delta_t^\gamma}{\Delta_x^2}\sum_{\{u_j^i \in U : u_j^i > u^{n-\frac{L}{\Delta_t}}\}} \psi(\gamma, n-i)\delta_j^i. \tag{3.4.4}$$

This recursive relation can be written as

$$\begin{aligned} u_j^{n+1} &= u_j^n + \alpha \frac{\Delta_t^\gamma}{\Delta_x^2} \sum_{\{u_j^i \in U^n\}} \psi(\gamma, n-i)\delta_j^i. \\ U^{n+1} &:= \{u^i \in U^n : u^i > u^{n-\frac{L}{\Delta_t}}\} + \{u^{n+1}\} \end{aligned} \tag{3.4.5}$$

with $U^0 := \{u^0\}$. As the function evolves, elements in U given by $u^i < u^{n-\frac{L}{\Delta_t}}$ will never be used again and can be removed. □

Numerically only the set U^n needs to be stored in memory for the current time point n and all points after. As discussed above, by its construction adaptive memory time step as an arithmetic sequence necessitates a shifting of the calculated window of points, and as such the entire history of the system needs to be stored in memory for the calculation at all time points. However, we can construct an adaptive memory as a power law, such that once we know what past history points need to be calculated, all other points in between will never need to be calculated. This then requires only enough memory to store the known and computed past intervals of the systems history, allowing the deallocation and recovery of much of the stored memory. This results in less memory requirements which translates into much faster computations.

Definition 3.4.3. Define a parameter η which determines the ‘reset interval’. This represents the number of points in the past history to store in the current time point set U^n at each weight.

Definition 3.4.4. Define weighting sets W^i with elements w^i ordered in the same manner as the sets U^i.

Algorithm 3.4.1. Assume an adaptive memory time step power law scheme. Assume $w^0 = 1$. When there are more than η points in the set W^i of any given weight, define a subset of W^i as the elements in W^i of the given weight. Then from this subset, the weight

of the least element is doubled, and the second lowest element is is removed from W^i altogether. The time marching scheme is given as follows:

$$\begin{aligned}
u_j^{n+1} &= u_j^n + \alpha \frac{\Delta_t^\gamma}{\Delta_x^2} \sum_{\{u_j^i \in U^n\}} \psi(\gamma, n-i) w^i \delta_j^i. \\
w^{n+1} &= 1 \\
W^{n+1} &:= \{w^i \in W^n\} + \{w^{n+1}\} \\
U^{n+1} &:= \{u^i \in U^n\} + \{u^{n+1}\}
\end{aligned} \tag{3.4.6}$$

so that when the number N elements of a given weight is $N > \eta$ the algorithm is condensed.

3.4.2 Numerical Implementation

From an applied perspective, to keep a well ordered list of points in U we make use of linked lists as the data structure. A linked list is one of the fundamental data structures used in computer programming. It consists of nodes that can contain data and their own data structures, and also pointers from each node to the next and/or previous nodes (Fig. 3.4.2A). This means one node can be removed from the set and deallocated from memory while the order of the set is maintained. In our case each node is an item u of the set U describing a time point necessary for the current time step, with the entire list representing all points u that make up the current iteration U^n. Once a time point u is no longer necessary for the simulation, it can be removed (Fig. 3.4.2B). New computed time points, u^{n+1} are added into the list (Fig. 3.4.2C). The data structure is initialized as illustrated in Fig. 3.4.2D. At each time step, a new structure containing u^{n+1} and w^{n+1} is added onto the end of the list. When the elements of a specific weight ω grow larger than the limit η, the first two values of the weight ω are condensed into one value, with the weight doubling and one value being removed from the memory. We show as an example the fourth step in a simulation when $\eta = 3$ (Fig. 3.4.2E). As this process iterates in time, one has a condensed list of only the time points that will be needed for the rest of the simulation in the list. For example, Fig. 3.4.2F shows the transition to the 17^{th} step of a simulation.

3.5 A Smart Adaptive Time Step Memory Algorithm

Regular diffusion relies on the assumption of a Markovian process that is not necessarily present in natural systems. One approach to modeling these non-Markovian processes is using the fractional diffusion equation. Mathematically such methods have been around for decades but it is relatively recently that they have been applied to the natural sciences. Computationally efficient methods for numerically evaluating these equations are a necessity if fractional calculus models are to be applied to modeling real world physical and biological processes. It should be noted that while in this work the simulations were done in the subdiffusive regime for a simple point source, the methods we derive here are directly applicable to more complex sources or superdiffusion ($\gamma > 1$). However, complex fast-evolving past histories in the form of a forcing function $(f(u))$ or oscillations generated in a superdiffusion regime will result in much larger errors for both the short and adaptive memory methods. In the case of the adaptive memory method introduced in this chapter, this is due to its 'open-loop'-like algorithm that blindly increases the spacing between points in the summation as the calculation goes further back in time and which does not take into account the speed of evolution of the equation. Adaptive time approaches for regular differential equations often make the integration step size a function of the derivative, i.e., more closely spaced time steps are

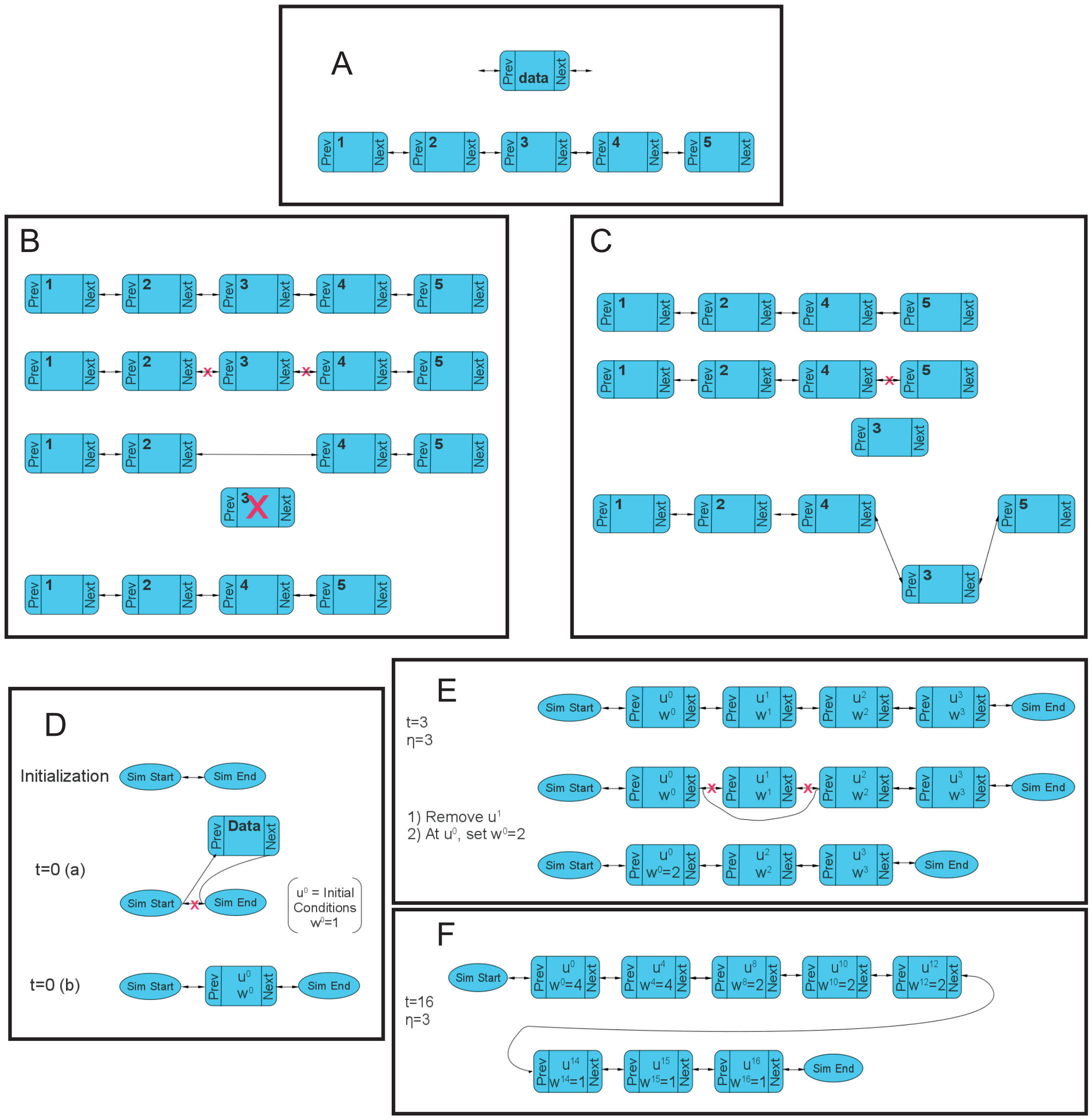


Figure 3.4.2: Overview of the use of linked list data structures for the algorithmic numerical implementation of adaptive memory time step as a power law scheme. See the text for details.

used when the function is oscillating quickly and cannot be averaged without loss of accuracy, and widely spaced time steps are used where the function is smooth. In the current implementation we have assumed that the past history function $\psi(\gamma, m)\delta_{i,j}^{n-m}$ is smooth. In this last section we extend the original implementation to develop a 'smart' 'closed-loop'-like algorithm where the step size of the backwards summation is dependent on the derivative of the past history function, i.e., a form of feedback. This optimizes the computational speed of the simulation while reducing the error due to the averaging of time points in the backwards summation, ultimately resulting in low errors for both high frequency forcing functions in the subdiffusion regime and for oscillations intrinsic to the superdiffusion regime.

3.5.1 Approximating the Discrete Grünwald-Letnikov Series as a Continuous Time Integral

Our analytical approach for smart adaptive memory makes use of the continuous Grünwald-Letnikov integral. In this form, we can then define a minimum threshold error function based on the derivative that ensures that no period in the history of the system is misrepresented up to the defined error.

Recalling equation 3.2.29, the discretized form of the fractional diffusion equation, the summation term can be defined as a Riemann sum, which in the limit as $\Delta_t \to 0$ approaches a continuous time integral. The benefits of an integral formulation is that the backwards integration would be separate from a defined time grid for the series, and higher order methods for integrating, such as Simpson's rule, could be used. This would be impossible in discrete time since it is necessary to project and interpolate between points on the grid.

Defining a Riemann sum S over an interval x_1 to x_n,

$$S = \sum_{i=0}^{n} g(y_i)(x_i - x_{i-1}) \tag{3.5.1}$$

there is a correspondence between the discrete summation and the integral

$$\frac{u_{j,l}^{n+1} - u^n}{\Delta_t} = \Delta_t^{\gamma-1} \sum_{m=0}^{t/\tau} \psi(\gamma, m) f(u_{n-m}).$$

such that,

$$\begin{aligned} g(y_i) &= \psi(\gamma, m) f(u_{n-m}) \\ i &= m \\ n &= t/\tau \\ x_i &= m = 0 \ldots n \end{aligned}$$

where the width of each segment is 1. As $\Delta_t \to 0$, this sum gets closer and closer to, and can be approximated by, the continuous time integral

$$\int_{\tau=0}^{t} \psi(\gamma, \tau/\Delta_t) f(u(t-\tau) d\tau \tag{3.5.2}$$

which allows the discretized version to be rewritten in continuous form as

$$\frac{u_{j,l}^{n+1} - u^n}{\Delta_t} = \Delta_t^{\gamma-1} \int_{\tau=0}^{t} \psi(\gamma, \tau/\Delta_t) f(u(t-\tau)) d\tau \tag{3.5.3}$$

The function $f(u(t-\tau))$ can be interpolated from the previously calculated values of u. The original definition of $\psi(\gamma, m)$ is only defined for $m \in \mathbb{N}_0$, and so needs to extend into the continuous domain. With an analytical continuous function representing ψ, one is then free to rediscretize the continuous integral in the most optimal way to solve the problem efficiently.

3.5.2 Extension of $\psi(\gamma, m)$ to the Positive Real Domain

We are interested in a function $\Psi(\gamma, r)$ that is an extension of $\psi(\gamma, m)$ into the positive real domain and is defined such that for all $r \in \mathbb{N}_0$, $\psi(\gamma, r) = \psi(\gamma, m)$. We begin with a basic linear interpolation of $\psi(\gamma, m)$ over non-integer space, and the result is shown in Fig. 3.5.1A for various values of γ. While a linear interpolation provides a reasonable approximation of ψ, it is not a smooth function and it is a first order approximation that obviously does not work well for areas of ψ that have a high second derivative (e.g., $r \ll 1$, for $\gamma < 1$). Since we don't have the ability to increase the number of points we are basing our interpolation on, we consider other options to obtain a more accurate and smoother approximation.

Next, we consider simply extending the original definition of $\psi(\gamma, m)$ and expanding it to all points r by using the gamma function to re-express the factorial. With r extending to non-integer space in the positive real domain, and with the $(-1)^r$ term, we get an oscillating complex function. A plot of the real part of the function,

$$\Psi(\gamma, r) = Real\left\{\frac{(-1)^r\Gamma(2-\gamma)}{\Gamma(r+1)\Gamma(2-\gamma-r)}\right\}, \tag{3.5.4}$$

shows that this method results in a poor approximation of ψ due to the oscillatory behavior (Fig. 3.5.1B.)

Another possibility for deriving a continuous function Ψ is to consider a rational polynomial function. One can rewrite the recursive series $\psi(\gamma, m)$ as a truncated approximation using the relation from equation 3.2.14:

$$\begin{aligned}\psi(\gamma, m) &= -\psi(\gamma, m-1)\frac{2-\gamma-m}{m} \\ \psi(\gamma, 0) &= 1\end{aligned}$$

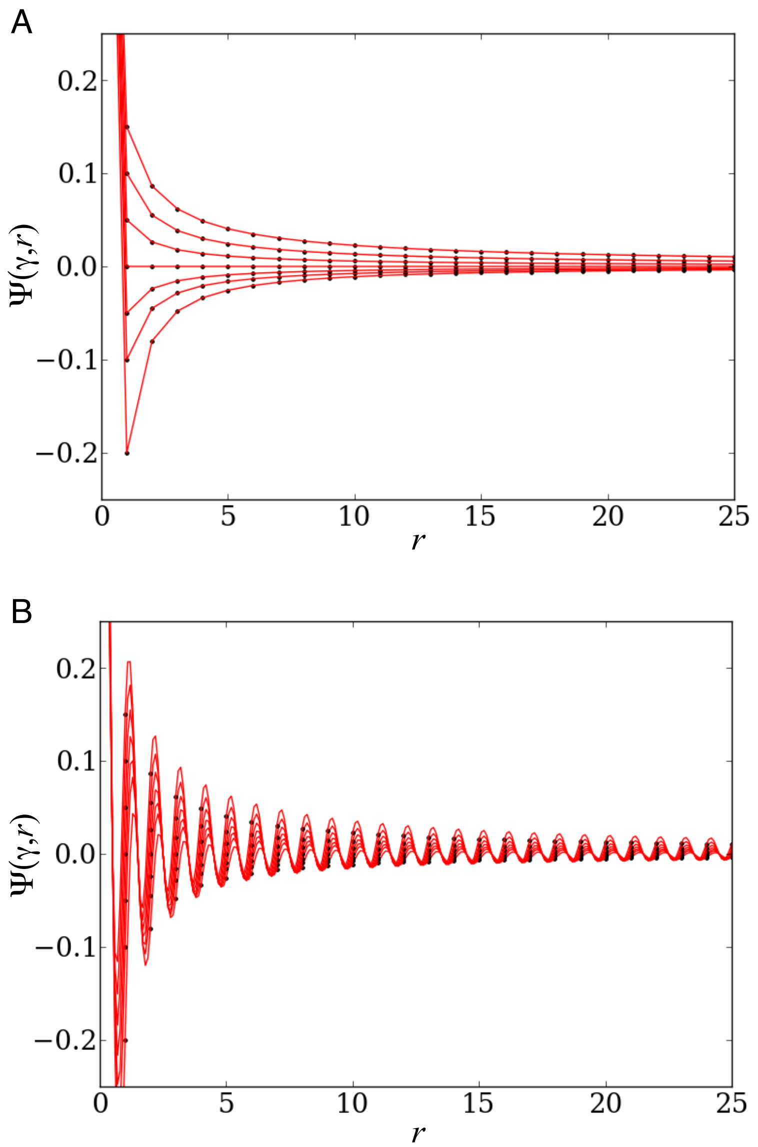


Figure 3.5.1: Computing a continuous version of $\psi(\gamma, m)$ for various values of γ. In all subplots, $\gamma = (.85, .90..., 1.10, 1.15)$ from the bottom trace to the top. Exact function ψ is denoted by discrete points. **A)** $\Psi(\gamma, r)$ as a linear interpolation of the exact $\psi(\gamma, m)$. **B)** Extending the definitions of $\psi(\gamma, m)$ to all r in the positive real domain.

This can be written in terms of a finite product as

$$\psi(\gamma,m) = \frac{(-1)^m\Gamma(2-\gamma)}{m!\Gamma(2-\gamma-m)} = \cdots \tag{3.5.5}$$
$$-\psi(\gamma,m-1)\frac{2-\gamma-m}{m} = \prod_{n=1}^{m}\left[\frac{\gamma+n-2}{n}\right] = \prod_{n=1}^{m}\left[1+\frac{\gamma-2}{n}\right]$$

Then, using a transform to show that an infinite rational function will intersect all these points, define a rational function so that

$$\Psi(\gamma,r) = R_{\alpha,\beta} = \frac{P_\alpha}{Q_\beta} \tag{3.5.6}$$

where

$$R_{\alpha,\beta} = \frac{p_0+p_1r+p_2r^2...p_\alpha r^\alpha}{q_0+q_1r+q_2r^2...q_\beta r^\beta}. \tag{3.5.7}$$

As $(\alpha,\beta)\to\infty$, $\Psi(\gamma,r)$ will approach $\psi(\gamma,m)$ for all $r\in\mathbb{N}_0$.

An infinite rational expression, however,would be too costly to compute. One can truncate the expression however, and get a closed-form expression with a very close fit that approaches the analytical recursive $\psi(\gamma,m)$. As $m\to\infty$, $\psi\to 0$, which implies that $\beta>\alpha$ for this truncation.

Given the number of coefficients $p_0,...,p_\alpha,q_0,...,q_\beta$ and flexibility in choosing α and β, there are multiple possible solutions to equation 3.5.7. But for all cases we consider the basic set of constraints given by the system of equations:

$$\begin{aligned}
\psi(\gamma,0) &= \Psi(\gamma,0) \\
\psi(\gamma,1) &= \Psi(\gamma,1) \\
\psi(\gamma,2) &= \Psi(\gamma,2) \\
&\cdots \\
\psi(\gamma,M) &= \Psi(\gamma,M)
\end{aligned} \tag{3.5.8}$$

where M is an integer.

The values of α and β are also important considerations, since higher order polynomials will yield a more accurate approximation. Although the resulting function Ψ will increase in complexity along with accuracy, a powerful advantage of a $\psi \to \Psi$ transform that uses a finite rational polynomial function is that we will obtain a continuous version of the recursive function ψ, that is a closed-form expression.

No matter the exact method or transform used to obtain the new function $\Psi(\gamma,r)$, once derived, we can then directly insert it into the numerical implementation. We can drop all points in the past history function ($\Psi(\gamma,m)f(u_{k-m})$) in the regions where the second derivative is below a certain threshold (i.e., where the function is slowly changing), and integrate the function on the resultant mesh using equation 3.5.3, with the substitution of the continuous $\Psi(\gamma,r)$, with $r = \tau/\Delta_t$:

$$\frac{u_{j,l}^{n+1} - u^n}{\Delta_t} = \Delta_t^{\gamma-1} \int_{\tau=0}^{t} \Psi(\gamma,\tau/\Delta_t) f(u(t-\tau)) d\tau \tag{3.5.9}$$

As discussed before, this smart adaptive step extension to the original algorithm will allow us to minimize errors by using smaller integration step sizes when the past history function is quickly changing due to a fast-evolving external forcing function to the system, or oscillating behavior integral to the dynamics of the system itself. In addition, this approach will be able to take advantage of the large body of existing literature and numerical methods solutions packages for solving continuous equations.

Acknowledgments

The authors wish to thank Dr. Richard Magin and Dr. Igor Podlubny for helpful discussions during the course of this work. This work was supported by NIH NS054736 and ARO 63795EGII.

Chapter 3, in part, has been submitted for publication of the material as it may appear in the Journal of Computational Physics, 2015, Christopher L. MacDonald, Nirupama Bhattacharya, Brian P. Sprouse, Gabriel A. Silva. The dissertation author was a co-author of this paper.

Chapter 4

Stability and Complexity Analyses of Finite Difference Algorithms for the Time-fractional Diffusion Equation

4.1 Introduction

As we have established in Chapter 2, the time-fractional diffusion equation is an FDE that represents the underlying physical mechanism of anomalous diffusion. In the same way that the classical diffusion equation can be derived from statistical analysis of particle interactions, one can also show how the micromolecular behavior of particles can, under a different set of fundamental statistical assumptions, be represented on a macro scale by the fractional diffusion equation of the form:

$$\frac{\partial^\gamma u(\vec{x},t)}{\partial t^\gamma} = \alpha \frac{\partial^2 u(\vec{x},t)}{\partial x^2} + \beta \frac{\partial^2 u(\vec{x},t)}{\partial y^2} \tag{4.1.1}$$

where $\gamma < 1$ denotes the subdiffusion regime, $\gamma = 1$ denotes classical Gaussian diffusion, $\gamma > 1$ denotes the superdiffusion regime, and α and β represent the diffusion coefficients in the x and y directions, respectively (in $\frac{spatial\ unit^2}{time\ unit^\gamma}$).

Finding tractable analytical solutions to FDEs such as Eq. 4.1.1 is often much more involved than solving for the solutions of integer order differential equations, and in many cases it is not possible to frame solutions in a closed form expression that can be easily simulated or visually represented. Therefore the development of numerical methods is vital; over the last decade there have been numerous numerical algorithms developed for FDEs like the time-fractional diffusion equation, and there is often a tradeoff between calculation speed and efficiency, complexity, stability, and accuracy. In Chapter 2 we walk through the discretization of Eq. 4.1.1 into the full 2D fractional FTCS (Forward Time Central Space) equation, making use of the Grünwald-Letnikov definition of the fractional derivative, which allows us to use a discrete summation in our numerical algorithm. In Chapter 3, we also derive an adaptive time step algorithm which builds on the foundation of the full two-dimensional fractional FTCS equation but improves calculation speed and efficiency, while maintaining order of accuracy. However, both the FTCS equation and adaptive algorithms are explicit methods with a limited stability regime. Here we fully explore and characterize the stability of these two finite difference schemes in order to easily obtain bounds on important parameters like time step and spatial discretization and grid size; selecting these parameters appropriately is crucial for obtaining accurate simulations.

There are numerous approaches to analyzing the stability of finite difference schemes, including modified wavenumber analysis, matrix eigenvalue analysis, and other more mathematically complex methods derived from stability definitions involving matrix norms. Several methods, including matrix analysis, are widely applicable but sometimes impossible to approach analytically or without the aid of iterative procedures. However, since the numerical algorithms in question are both fully discretized in space and time, and linear with constant coefficients (in their homogenous versions), as we will see in the next section, an appropriate choice is a von Neumann stability analysis. We assume the general solution at an arbitrary location and time step, is a linear combination of special solutions that are represented by a function that is separable in its temporal and spatial variables. We find that our stability analysis and expressions of parameter bounds agree very well with our results, where we simulate several examples with various parameter combinations.

In addition to stability analysis, a complexity analysis is another important metric used to characterize the efficiency of numerical algorithms, as measured by execution time of the algorithm as a function of some input variable. In the case of the algorithms described in this paper, we might be interested in how the execution time might vary with N_x, N_y(grid size of the simulation in the x and y directions, respectfully), or N, the number of timesteps. The relationship between execution time and input variable is usually given in Big-O notation using a worst case scenario, or average case scenario which is often a better reflection of the average behavior of the algorithm run time. In Section 4.3, we explore the time complexity of the same two algorithms for which we complete the stability analysis, and in addition, analyze an alternate version of the adaptive timestep algorithm that involves a linked list implementation (introduced in Chapter 3) that yields better algorithmic efficiency. For mathematical simplicity we analyze our algorithms according to worst case scenarios and interpret the results as a proof of bounding behavior - the actual run time of the algorithms may be more efficient, but never less efficient. Section 4.3 shows simulated data that verifies our theoretical complexity analyses.

4.2 Stability Analysis of the Two-dimensional Fractional FTCS Discretization

4.2.1 Full Implementation

We begin by considering the circumstances under which the 2D fractional FTCS full finite difference discretized equation is stable, and consider the advantages and disadvantages of several different analytical approaches. We begin by restating the FTCS discretized equation:

$$\begin{aligned} u_{j,l}^{n+1} - u_{j,l}^{n} &= \Delta_t^{\gamma} \sum_{m=0}^{n} \psi(\gamma,m) \left(\frac{\alpha}{\Delta x^2} \delta x_{j,l}^{n-m} + \frac{\beta}{\Delta y^2} \delta y_{j,l}^{n-m} \right) && (4.2.1) \\ \delta x_{j,l}^{n} &= u_{j-1,l}^{n} - u_{j,l}^{n} + u_{j+1,l}^{n} && (4.2.2) \\ \delta y_{j,l}^{n} &= u_{j,l-1}^{n} - 2u_{j,l}^{n} + u_{j,l+1}^{n} && (4.2.3) \end{aligned}$$

where Δ_t is the time step, and Δx and Δy are spatial grid sizes. $\psi(\gamma,m)$, which we refer to as a 'memory function', represents a binomial term $(-1)^m \begin{pmatrix} 1-\gamma \\ m \end{pmatrix}$, and results from the use of the Grünwald-Letnikov definition of the fractional derivative, as discussed in Chapter 2 and in [40].

Matrix Stability Analysis

Matrix stability analysis is a method of analysis that can be applied to any problem without particular assumptions or constraints on coefficients, and includes the effects of boundary conditions [37]. We would like to reformulate Eq. 4.2.1 into a matrix equation in the form $W^{n+1} = LW^n + f^n$ where the W matrices holds u values at times $n+1$ and n, the L matrix represents a difference operator, and f^n is a constant vector that takes into account boundary conditions at time step n (here we assume boundary conditions are known for all timesteps).

For example, for $n = 0$, we have:

$$u_{j,l}^{1} = u_{j,l}^{0} + \Delta_t^{\gamma} \psi(\gamma, 0) \left(\frac{\alpha}{\Delta x^2} \delta x_{j,l}^{0} + \frac{\beta}{\Delta y^2} \delta y_{j,l}^{0} \right)$$

Then the matrix equation is formulated as follows. Suppose our grid of W values varies from

$$u_{j,l}^{n} \mid j = 0, 1, 2, ..., N_x;\ l = 0, 1, 2, ..., N_y$$

and

$$u_{j,l}^{n} \mid j = 0, N_x \text{ or } l = 0, N_y$$

denotes the boundaries.

We order grid values u into a vector such that W at some time step n is

$$W^n = \begin{bmatrix} u_{0,0} \\ u_{0,1} \\ u_{0,2} \\ \vdots \\ u_{0,N_y} \\ u_{1,0} \\ u_{1,1} \\ u_{1,2} \\ \vdots \\ u_{1,N_y} \\ \vdots \\ u_{N_x,0} \\ u_{N_x,1} \\ u_{N_x,2} \\ \vdots \\ u_{N_x,N_y} \end{bmatrix}^n \tag{4.2.4}$$

Our matrix equation now becomes $W^1 = L_0 W^0 + f^0$ where the matrix L is an $N_x N_y$ by $N_x N_y$

matrix that consists of block matrices T (a tridiagonal matrix) and R (a diagonal matrix):

$$L_0 = \begin{bmatrix} T_0 & R_0 & 0 & \cdots & 0 \\ R_0 & T_0 & \ddots & \ddots & \vdots \\ 0 & R_0 & \ddots & R_0 & 0 \\ \vdots & \ddots & \ddots & T_0 & R_0 \\ 0 & \cdots & 0 & R_0 & T_0 \end{bmatrix}$$

$$T_0 = \begin{bmatrix} 1-2r_x\psi(\gamma,0)-2r_y\psi(\gamma,0) & r_y\psi(\gamma,0) & 0 & \cdots & 0 \\ r_y\psi(\gamma,0) & & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & & 0 \\ \vdots & \ddots & & & r_y\psi(\gamma,0) \\ 0 & \cdots & 0 & r_y\psi(\gamma,0) & 1-2r_x\psi(\gamma,0)-2r_y\psi(\gamma,0) \end{bmatrix}$$

$$R_0 = \begin{bmatrix} r_x\psi(\gamma,0) & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & r_x\psi(\gamma,0) \end{bmatrix}$$

where $r_x = \frac{\alpha\Delta_t^\gamma}{\Delta x^2}$ and $r_y = \frac{\beta\Delta_t^\gamma}{\Delta y^2}$.

For $n = 1$ we now must take into account multiple past time points, and our matrix equation becomes

$$
\begin{aligned}
W^2 &= L_1 W^0 + L_0 W^1 + f^{01} \\
L_1 &= \begin{bmatrix} T_1 & R_1 & 0 & \cdots & 0 \\ R_1 & T_1 & \ddots & \ddots & \vdots \\ 0 & R_1 & \ddots & R_1 & 0 \\ \vdots & \ddots & \ddots & T_1 & R_1 \\ 0 & \cdots & 0 & R_1 & T_1 \end{bmatrix}, \\
T_1 &= \begin{bmatrix} 1-2r_x\psi(\gamma,1)-2r_y\psi(\gamma,1) & r_y\psi(\gamma,1) & 0 & \cdots & 0 \\ r_y\psi(\gamma,1) & & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & & 0 \\ \vdots & \ddots & & & r_y\psi(\gamma,1) \\ 0 & \cdots & 0 & r_y\psi(\gamma,1) & 1-2r_x\psi(\gamma,1)-2r_y\psi(\gamma,1) \end{bmatrix} \\
R_1 &= \begin{bmatrix} r_x\psi(\gamma,1) & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & r_x\psi(\gamma,1) \end{bmatrix}
\end{aligned}
$$

and f^{01} takes into account the boundary conditions from timesteps 0 and 1. We can repurpose this multistep matrix equation into a single step equation:

$$
\begin{aligned}
V^2 &= LV^1 + f^{01} \text{ with} \\
V^2 &= \begin{bmatrix} W^2 \\ W^1 \end{bmatrix}, V^1 = \begin{bmatrix} W^1 \\ W^0 \end{bmatrix}, L = \begin{bmatrix} L_0 & L_1 \\ I & 0 \end{bmatrix}
\end{aligned}
$$

and I is simply the identity matrix. For $n = 2$ the complexity of the matrices involved continues to grow with

$$
\begin{aligned}
V^3 &= LV^2 + f^{012} \\
\begin{bmatrix} W^3 \\ W^2 \\ W^1 \end{bmatrix} &= \begin{bmatrix} L_0 & L_1 & L_2 \\ I & 0 & 0 \\ 0 & I & 0 \end{bmatrix} \begin{bmatrix} W^2 \\ W^1 \\ W^0 \end{bmatrix} + f^{012}
\end{aligned}
$$

where $L_0, L_1, L_2...$ are block matrices which consist of tridiagonal and diagonal submatrices, as demonstrated in previous steps. In general, we have

$$V^{n+1} = LV^n + f^{0...n} \tag{4.2.5}$$

with

$$\begin{bmatrix} W^{n+1} \\ \vdots \\ \vdots \\ \vdots \\ W^1 \end{bmatrix} = \begin{bmatrix} L_0 & L_1 & \cdots & L_{n-1} & L_n \\ I & 0 & & \cdots & 0 \\ 0 & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \\ 0 & \cdots & 0 & I & 0 \end{bmatrix} \begin{bmatrix} W^n \\ \vdots \\ \vdots \\ \vdots \\ W^0 \end{bmatrix} + f^{0...n} \tag{4.2.6}$$

The goal with matrix stability analysis is to show that for a matrix problem Eq. 4.2.6, the spectral radius $\rho(L)$ of the difference matrix L, (i.e., the modulus of its largest eigenvalue) is bounded, usually by 1 [11]. This involves finding the maximum eigenvalue for L, for all timesteps n, and proving that its modulus is ≤ 1. The matrix L is not a straightforward or simple matrix and is not in the form of a special matrix where there are known analytical algorithms to find the eigenvalues. This makes the process of finding the eigenvalues very complicated and involved and we consider this to be outside the scope of this thesis, and conclude that despite the difficulty of finding the spectral radius, a major advantage of this method would be that it could be applied to any set of boundary conditions, or schemes with non-constant coefficients.

Von Neumann Analysis

As we saw in the previous section, matrix stability analysis for complex schemes with no simplifying assumptions, requires methods for finding eigenvalues, possibly including iterative numerical schemes. However, our finite difference scheme in Eq. 4.2.1 has the advantage of being linear with constant coefficients (i.e. α, β don't depend on t or $\vec{x}$) and with an additional simplifying assumption of periodic boundary conditions, we can make use of von Neumann stability analysis and Fourier series expansion. While von Neumann analysis doesn't take into account boundary conditions in the same way matrix stability analysis does, we make the reasonable assumption that stability problems are mostly affected by discretization of differential equations inside the domain, and minimally affected by boundary conditions [37]. In the majority of modeling and simulation problems in engineering, we consider this to be a valid and applicable simplifying assumption.

Given our linear scheme with constant coefficient and periodic boundary conditions, we assume the solution of scheme 4.2.1 at a particular location and time point, can be expressed as a discrete finite complex Fourier series expansion:

$$u^n_{j,l} = \sum_{v=0}^{N_y-1} \sum_{u=0}^{N_x-1} A^n_{uv} e^{ik_u x_j} e^{ik_v y_l} \tag{4.2.7}$$

where $k_u = \frac{2\pi u}{M}$, $x_j = j\Delta x$, $j = 0, 1, ..., N_x - 1$ and $k_v = \frac{2\pi v}{L}$, $y_l = l\Delta y$, $l = 0, 1, ..., N_y - 1$. k_u and k_v are the spatial wavenumbers in the x and y directions, respectively. The wavenumbers represent spatial frequency and are related to wavelength by $\lambda_x = \frac{2\pi\Delta x}{k_u}$, $\lambda = \frac{2\pi\Delta y}{k_v}$. In the context of Fourier modes and stability analysis, we are interested in suppressing (or as we shall see later in this section, limiting the amplification factor of) the high frequency modes (represented by wavelengths on the order of grid sizes Δx and Δy and k values away from 0 and near π) because these modes have a low probability of corresponding to any real features of the true solution, and usually represent noise arising from finite precision arithmetic, that is integral to any numerical calculations.

We substitute Eq. 4.2.7 into Eq. 4.2.1, noting that $x_{j+1} = x_j + \Delta x$, $x_{j-1} = x_j - \Delta x$, $y_{l+1} = y_l + \Delta y$ and $y_{l-1} = y_l - \Delta y$. After making use of Euler's formula and half angle trigonometric identities, we obtain

$$0 = \sum_{v=0}^{N_y-1} \sum_{u=0}^{N_x-1} e^{ik_u x_j} e^{ik_v y_l} \Big\{ A_{uv}^{n+1} - A_{uv}^{n} + \\ \Delta_t^{\gamma} \left(\frac{4\alpha}{\Delta x^2} sin^2 \left(\frac{k_u \Delta x}{2} \right) + \frac{4\beta}{\Delta y^2} sin^2 \left(\frac{k_v \Delta y}{2} \right) \right) \sum_{m=0}^{n} \psi(\gamma, m) A_{uv}^{n-m} \Big\} \tag{4.2.8}$$

Here we apply the principle of orthogonality of complex exponentials, where it is well known that

$$\sum_{j=0}^{N-1} e^{ik_n x_j} e^{-ik_m x_j} = \begin{cases} N & if\ n = m \\ 0 & otherwise \end{cases} \tag{4.2.9}$$

where $k_n = \frac{2\pi n}{L}$, $k_m = \frac{2\pi m}{L}$, $x_j = \frac{jL}{N} = j\Delta x$.

Let the value in brackets in Eq. 4.2.8 be denoted as K_{uv}^n

$$\sum_{v=0}^{N_y-1} \sum_{u=0}^{N_x-1} e^{ik_u x_j} e^{ik_v y_l} K_{uv}^n = 0 \tag{4.2.10}$$

We can multiply Eq. 4.2.10 by $\sum_{j=0}^{N_x-1} e^{-ik_g x_j}$ and reorder the summations to get

$$\sum_{v=0}^{N_y-1} \sum_{u=0}^{N_x-1} e^{ik_v y_l} K_{uv}^n \sum_{j=0}^{N_x-1} e^{ik_u x_j} e^{-ik_g x_j} = 0 \tag{4.2.11}$$

Using the principle of orthogonality as stated in Eq. 4.2.9 we can simplify this expression to

$$\sum_{v=0}^{N_y-1}\sum_{u=0}^{N_x-1} e^{ik_v y_l} K_{gv}^n N_x = 0 \tag{4.2.12}$$

Now we repeat the process by multiplying Eq. 4.2.12 by $\sum_{l=0}^{N_y-1} e^{-ik_h y_l}$ and applying Eq. 4.2.9 once again to get

$$K_{gh}^n N_x N_y = 0$$

Since N_x and N_y are nonzero, we conclude that K_{gh}^n must vanish. Since g and h are dummy variables that are defined by the same ranges as u and v, K_{uv}^n must also vanish for all values u, v:

$$A_{uv}^{n+1} - A_{uv}^n + \Delta_t^\gamma \left(\frac{4\alpha}{\Delta x^2} sin^2\left(\frac{k_u \Delta x}{2}\right) + \frac{4\beta}{\Delta y^2} sin^2\left(\frac{k_v \Delta y}{2}\right)\right) \sum_{m=0}^{n} \psi(\gamma, m) A_{uv}^{n-m} = 0$$

If we make the assumption that A_{uv} is independent of time step, we can write

$$A_{uv}^{n+1} = \sigma_{uv} A_{uv}^n \tag{4.2.13}$$

where σ_{uv} is known as the amplification factor that gives the growth or decay for Fourier mode (u, v). We require $|\sigma_{uv}| \leq 1$ for all Fourier modes, or else some modes will be amplified at every time step and dominate the solution. We divide Eq. 4.2.13 by A_{uv}^n and rearrange to get

$$\sigma_{uv} = 1 - \Delta_t^\gamma \left(\frac{4\alpha}{\Delta x^2} sin^2\left(\frac{k_u \Delta x}{2}\right) + \frac{4\beta}{\Delta y^2} sin^2\left(\frac{k_v \Delta y}{2}\right)\right) \sum_{m=0}^{n} \psi(\gamma, m) \sigma_{uv}^{-m}$$

We require $|\sigma_{uv}| \leq 1$ for all values of σ_{uv}, which are the roots of this polynomial. We can also see that the difficulty of solving for the roots increases as time progresses and n increases. Past a few timesteps, solving for the roots of the polynomial requires iterative matrix procedures and does not result in a tractable analytical solution from which we can find the bounds on the relationship between $\alpha, \Delta t, \Delta x, \text{and} \Delta y$. Instead of solving for roots, we will approach the problem by considering ‘worst-case’ scenarios to simplify our expression, and proceed in a similar manner as Yuste and Acedo’s treatment of the analogous one-dimensional case [57]. Consider the bound on the parameters resulting from the case $\sigma = 1$:

$$0 \geq \Delta_t^\gamma \left(\frac{4\alpha}{\Delta x^2} sin^2 \left(\frac{k_u \Delta x}{2} \right) + \frac{4\beta}{\Delta y^2} sin^2 \left(\frac{k_v \Delta y}{2} \right) \right) \sum_{m=0}^{n} \psi(\gamma, m) \tag{4.2.14}$$

The summation term and parameters are always greater than 0, and therefore Eq. 4.2.14 can only hold true if both wavenumbers are 0. Therefore, we consider the other ‘worst-case’ scenario, where $\sigma = -1$:

$$2 = \Delta_t^\gamma \left(\frac{4\alpha}{\Delta x^2} sin^2 \left(\frac{k_u \Delta x}{2} \right) + \frac{4\beta}{\Delta y^2} sin^2 \left(\frac{k_v \Delta y}{2} \right) \right) \sum_{m=0}^{n} \psi(\gamma, m) (-1)^m \tag{4.2.15}$$

From (4.2.15) we can infer the inequality bounding the parameters as

$$\Delta_t^\gamma \left(\frac{4\alpha}{\Delta x^2} sin^2 \left(\frac{k_u \Delta x}{2} \right) + \frac{4\beta}{\Delta y^2} sin^2 \left(\frac{k_v \Delta y}{2} \right) \right) \leq \frac{2}{\sum_{m=0}^{n} \psi(\gamma, m) (-1)^m} = B_1 (\gamma, n) \tag{4.2.16}$$

We can see that the bounded value is dependent on the time step n but the dependence is very weak. Yuste and Acedo demonstrate that because in the limit $n \to \infty$, the summation term in Eq. 4.2.16 involving the memory function, is an alternating converging series; as n increases, B_1 oscillates but clearly converges to an equilibrium value [57]. Therefore we can approximate $\sum_{m=0}^{n} \psi(\gamma, m) (-1)^m$ by taking the limit as $n \to \infty$. We originally derived the 2D FTCS full finite difference equation and adaptive memory algorithms by using the Grünwald-Letnikov definition to make a first-order approximation of the Riemann-Liouville fractional derivative operator [40]. And in the first-order approximation, the memory function $\psi(\gamma,m)$ is defined as $(-1)^m \left(\begin{array}{c} 1-\gamma \\ m \end{array} \right)$. Podlubny ([40]) shows that this expression can also be defined as the coefficients of the power series for the function $(1-z)^{1-\gamma}$:

$$(1-z)^{1-\gamma} = \sum_{m=0}^{\infty} (-1)^m \left(\begin{array}{c} 1-\gamma \\ m \end{array} \right) z^m = \sum_{m=0}^{\infty} \psi(\gamma,m) z^m \tag{4.2.17}$$

Applying this identity to $B_1(\gamma,n)$ and considering the limit $n \to \infty$, we can derive

$$B_{2,full}(\gamma) = 2^{\gamma} \tag{4.2.18}$$

As Yuste and Acedo describe in their considerations of stability of the one-dimensional case, we could consider the second-order approximation of the Riemann-Liouville definition by making use of a different set coefficients to define our memory function. This option would result in a slightly lower bound on our parameters, but for the purposes of estimating appropriate parameter values, we consider the first-order approximation to be sufficient.

From (4.2.16) we can see that we have three degrees of freedom in parameters that we are free to choose for our simulations $(\Delta_t,\ \Delta x^2,\ \Delta y^2)$. To determine the most conservative restraints on those degrees of freedom (minimum values for grid size and maximum value for time step), we assume the maximum possible value for the trigonometric terms, $sin^2\left(\frac{k_u \Delta x}{2}\right) = sin^2\left(\frac{k_v \Delta y}{2}\right) = 1$. The resulting inequality determining parameter bounds is

$$\Delta_t^{\gamma}\left(\frac{4\alpha}{\Delta x^2} + \frac{4\beta}{\Delta y^2}\right) \leq B_{2,full}(\gamma)$$

If we assume a simple case with a square grid $(\Delta x = \Delta y = \Delta)$ and equal diffusion coefficients in both spatial directions $(\alpha = \beta)$, the expression becomes

$$r = \frac{\alpha \Delta t^{\gamma}}{\Delta^2} \leq \frac{B_{2,full}(\gamma)}{8} = 2^{\gamma-3} = B_{full}(\gamma) \tag{4.2.19}$$

As a final check, when we set $\gamma = 1$ (classical diffusion), we recover the well known traditional bound $r \leq \frac{1}{4}$ that results from the ordinary 2D diffusion equation using the FTCS discretization.

Error Propagation

Alternatively we can interpret stability in terms of error propagation and find the resulting bounds expression to be the same as in Section 2.1.2. Let $u_{j,l}^n$ be the exact solution of Eq. 4.2.1 (assuming no roundoff error) and $U_{j,l}^n$ be the approximate solution due to roundoff error from floating point or finite precision arithmetic. Let the error at an arbitrary j, l, n be defined as

$$\varepsilon_{j,l}^n = u_{j,l}^n - U_{j,l}^n$$

If we substitute this into Eq. 4.2.1 we find that the equation for $\varepsilon_{j,l}^n$ is

$$\varepsilon_{j,l}^{n+1} - \varepsilon_{j,l}^n = \Delta_t^{\gamma} \sum_{m=0}^{n} \psi(\gamma, m) \left(\frac{\alpha}{\Delta x^2} \left(\varepsilon_{j+1,l}^{n-m} - 2\varepsilon_{j,l}^{n-m} + \varepsilon_{j-1,l}^{n-m} \right) + \frac{\beta}{\Delta y^2} \left(\varepsilon_{j,l+1}^{n-m} - 2\varepsilon_{j,l}^{n-m} + \varepsilon_{j,l-1}^{n-m} \right) \right) \tag{4.2.20}$$

We can take the same approach as in Section 2.1.2 and apply a Fourier expansion to express the error at a particular time and location as

$$\varepsilon_{j,l}^n = \sum_{v=0}^{N_y-1} \sum_{u=0}^{N_x-1} \zeta_{uv}^n e^{ik_u x_j} e^{ik_v y_l}$$

Here we interpret stability of an algorithm as ensuring that the overall error due to roundoff errors integral to any numerical calculation, does not propagate in time:

$$\frac{\left|\varepsilon_{j,l}^{n+1}\right|}{\left|\varepsilon_{j,l}^{n}\right|} = \frac{\left|\zeta_{j,l}^{n+1}\right|}{\left|\zeta_{j,l}^{n}\right|} = |G| \leq 1$$

where G is the amplification factor. The rest of the mathematical derivation progresses in much the same manner as in Section 2.1.2 and results in the same parameter boundaries.

4.2.2 Adaptive Memory Algorithm

As discussed in Chapter 3, while the full implementation represented by Eq. 4.2.1 is an accurate finite difference approximation to the solution of Eq. 4.1.1, using the Grünwald-Letnikov definition of the fractional derivative involves a summation that takes into account the entire past history of the simulation. As the simulation progresses, calculating the summation term becomes increasingly cumbersome, time consuming, and memory intensive. Therefore we have developed an adaptive memory method (introduced in Chapter 3) that improves on computational efficiency while maintaining an order of accuracy comparable to the original full implementation. This is achieved by recognizing that the further back a time point in the history of the simulation, the less it contributes to the calculation of the solution at the next time point. Therefore we sample the history of the system more frequently for recent time points (which contribute more to the solution at the next time step) and less often for time points further back in the history of the system, weighted by an appropriate amount to compensate for less frequent sampling; this approach significantly reduces actual computational time. For smooth functions, the convolution of $\psi(\gamma, m)$and the function in question is sufficiently slowly changing that the weighting of the sampled time points compensates for the less frequent sampling and maintains overall accuracy. The motivation and derivation of this algorithm was discussed in further detail in Chapter 3. Here we reproduce a modified version of the algorithm and recognize that analyzing stability requires only a few modifications from the process described in the last section:

$$
\begin{aligned}
\frac{u_{j,l}^{n+1} - u_{j,l}^{n}}{\Delta_t^{\gamma}} &= \sum_{m=0}^{a} \overbrace{\psi(\gamma, m)\left(\frac{\alpha}{\Delta x^2}\delta x_{j,l}^{n-m} + \frac{\beta}{\Delta y^2}\delta y_{j,l}^{n-m}\right)}^{R1} \qquad (4.2.21)\\
&+ \sum_{s=2}^{s_{max}} \left\{ \sum_{\eta=1}^{\eta_{max}(s,n)} \overbrace{(2s-1)\,\psi\left(\gamma, M(s,\eta)\right)\left(\frac{\alpha}{\Delta x^2}\delta x_{j,l}^{n-M} + \frac{\beta}{\Delta y^2}\delta y_{j,l}^{n-M}\right)}^{R2} \right. \\
&\left. + \sum_{p=M_{max}+s_{max}}^{min(a^s,n)} \overbrace{\psi(\gamma, p)\left(\frac{\alpha}{\Delta x^2}\delta x_{j,l}^{n-p} + \frac{\beta}{\Delta y^2}\delta y_{j,l}^{n-p}\right)}^{R3} \right\} \\
M(s,\eta) &= a^{s-1} + (2s-1)\eta - s + 1 \\
\eta_{max}(s,n) &= min\left(\left\lfloor \frac{a^s - a^{s-1}}{2s-1} \right\rfloor, \left\lfloor \frac{n - a^{s-1}}{2s-1} \right\rfloor\right) \\
M_{max} &= M(s_{max}, \eta_{max}) \\
\delta x_{j,l}^{n} &= u_{j+1,l}^{n} - 2u_{j,l}^{n} + u_{j-1,l}^{n} \\
\delta y_{j,l}^{n} &= u_{j,l-1}^{n} - 2u_{j,l}^{n} + u_{j,l+1}^{n} \qquad (4.2.22)
\end{aligned}
$$

where a is the base interval, $\lfloor\ \rfloor$ denotes the floor function, and s_{max} is determined by the current time step n such that $a^{s_{max}-1} + 1 \leq n \leq a^{s_{max}}$. The terms $R1, R2, R3$ are referred to in more detail during the complexity analysis in later sections.

There are various ways to interpret how a is translated into a numerical algorithm, depending on whether one assumes that it is with respect to m (mathematical series beginning at 0), time step n (which may need to be adjusted to begin from index 0 or 1, depending on the programming language), or time t. For example, if one assumes a is a time, we can define $a_{effective} = a/dt$ where a is converted to a time step. For $dt < 1$

this has the same effect as increasing a and interpreting it with respect to time step n. The larger the value of a, the more accurate the scheme presented in Eq. 4.2.21, when compared against the full algorithm in Eq. 4.2.1. Fig. 4.2.1 shows the error plots (where the adaptive step algorithm in Eq. 4.2.21 is compared against the reference case Eq. 4.2.1, for the cases $a = 4$ and $a = 20$.

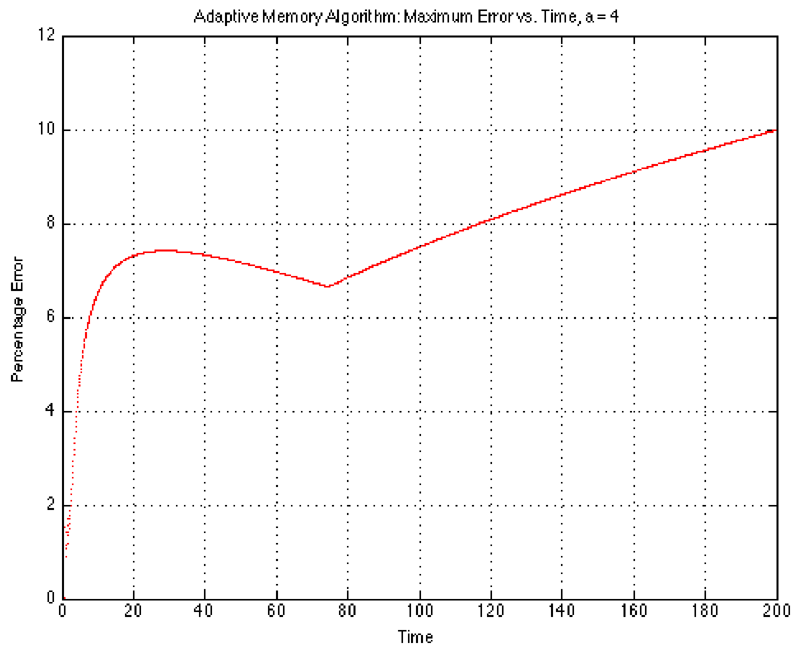
As with the full implementation, we apply a von Neumann stability analysis, assume the solution is separable in the form Eq. 4.2.7, substitute into Eq. 4.2.21 and simplify in the same fashion as described in Section 4.2.1 to yield

$$\begin{aligned} 0 \;=\; & \sum_{v=0}^{N_y-1}\sum_{u=0}^{N_x-1} e^{ik_u x_j} e^{ik_v y_l} \Big(A_{uv}^{n+1} - A_{uv}^{n} \\ & + \Delta_t^{\gamma}\left(\frac{4\alpha}{\Delta x^2} sin^2\left(\frac{k_u \Delta x}{2}\right) + \frac{4\beta}{\Delta y^2} sin^2\left(\frac{k_v \Delta y}{2}\right)\right)\left[\sum_{m=0}^{n}\psi(\gamma,m) A_{uv}^{n-m}\right. \\ & \left. + \sum_{s=2}^{s_{max}}\left\{\sum_{\eta=1}^{\eta_{max}(s,n)}(2s-1)\,\psi(\gamma,M(s,\eta))\,A_{uv}^{n-M} + \sum_{p=M_{max}+s_{max}}^{min(a^s,n)}\psi(\gamma,p)A_{uv}^{n-p}\right\}\right]\Big) \end{aligned}$$

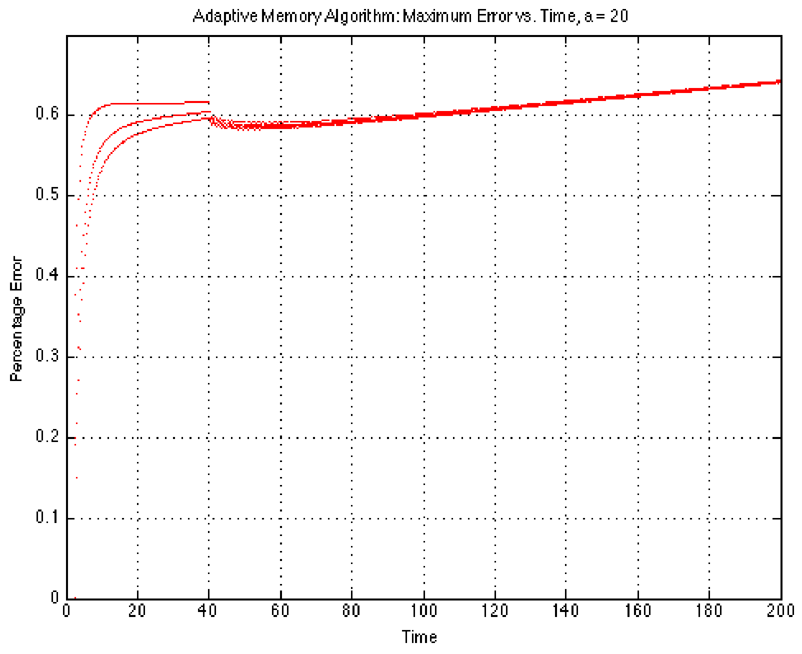
As before, we apply the principles of orthogonality of the complex exponentials, and once again assume $A_{uv}^{n+1} = \sigma_{uv} A_{uv}^{n}$ to get the following expression:

$$\begin{aligned} \sigma_{uv} \;=\; & 1 - \Delta_t^{\gamma}\left(\frac{4\alpha}{\Delta x^2} sin^2\left(\frac{k_1 \Delta x}{2}\right) + \frac{4\beta}{\Delta y^2} sin^2\left(\frac{k_2 \Delta y}{2}\right)\right)\left[\sum_{m=0}^{a}\psi(\gamma,m)\sigma_{uv}^{-m}\right. \\ & + \sum_{s=2}^{s_{max}}\left\{\sum_{\eta=1}^{\eta_{max}(s,n)}(2s-1)\,\psi(\gamma,M(s,\eta))\,\sigma_{uv}^{-M}\right. \\ & \left.\left. + \sum_{p=M_{max}+s_{max}}^{min(a^s,n)}\psi(\gamma,p)\sigma_{uv}^{-p}\right\}\right] \end{aligned} \qquad (4.2.23)$$

Again, for stability we require that $\mid \sigma_{uv} \mid < 1$ for all values of σ. If we assume the "worst-case" scenario ($\sigma = -1$), we can infer the following inequality to provide bounds



A)



B)

Figure 4.2.1: **Effect of** a **on Accuracy. A)** With adaptive step parameter $a = 4$, defined with respect to time step, we see the error (compared against the full 2D discretization defined in Eq. 4.2.1) grows significantly. **B)** For $a = 20$, the error remains consistently below 0.7%, even as long as 200 s into the simulation. The tradeoff for this increase in accuracy is a longer computation time. For both cases the following parameters are used: $\gamma = 0.6$, $\alpha = \beta = 50\ \frac{units^2}{s^\gamma}$, $dt = 0.1s$, $\Delta x = \Delta y = 10\ units$, $N_{x,} = N_y = 20$.

on the time step and spatial steps

$$B_1(\gamma,n,a) = \frac{2}{\Xi} \geq \Delta_t^{\gamma}\left(\frac{4\alpha}{\Delta x^2} sin^2\left(\frac{k_u \Delta x}{2}\right) + \frac{4\beta}{\Delta y^2} sin^2\left(\frac{k_v \Delta y}{2}\right)\right) \tag{4.2.24}$$

$$\begin{aligned}\Xi = & \sum_{m=0}^{a} \psi(\gamma,m)(-1)^m \\ & + \sum_{s=2}^{s_{max}}\left\{\sum_{\eta=1}^{\eta_{max}(s,n)} (2s-1)\psi(\gamma,M)(-1)^M \right. \\ & \left. + \sum_{p=M_{max}+s_{max}}^{min(a^s,n)} \psi(\gamma,p)(-1)^p\right\}\end{aligned} \tag{4.2.25}$$

If we assume a simple case with a square grid and equal diffusion coefficients, the most conservative restraint occurs when the trigonometric terms are equal to 1 and we get

$$r = \frac{\alpha \Delta t^{\gamma}}{\Delta^2} \leq \frac{B_1(\gamma,n,a)}{8} = \frac{1}{4\Xi} = B_{adap}(\gamma,n,a) \tag{4.2.26}$$

Unlike the full implementation in Section 4.2.1 where we could use the convergent nature of the series to approximate the bound B_{adap} with an analytical expression, the adaptive memory algorithm involves the parameter a which makes it difficult to take a limit approach to the expression Ξ. Fig. 4.2.2A shows the value of Ξ for a simulation with parameters $\gamma = 0.6$, and a varying from $a = 4$ to $a = 11$. We can see that for each value of s, the interval oscillates around some equilibrium value and if extended over infinite timesteps, would converge to said value. As would be expected, we also observe that as a increases, the value of Ξ approaches the the value of the summation in (4.2.16) in the analogous full 2D case.

From Ξ in Eq. 4.2.25, we can take the sum over η for a given value of s

$$\sum_{\eta=1}^{\eta_{max}(s,n)} (2s-1)\psi(\gamma,M(s,\eta))(-1)^{M(s,\eta)} \tag{4.2.27}$$

Table 4.1: Error values between Ξ and Ξ_{approx} for various values of a.

a	4	5	6	7	8	9	10	11
$\lvert\%Error\rvert$	2.043	1.845	1.717	1.424	1.293	1.097	1.010	0.878

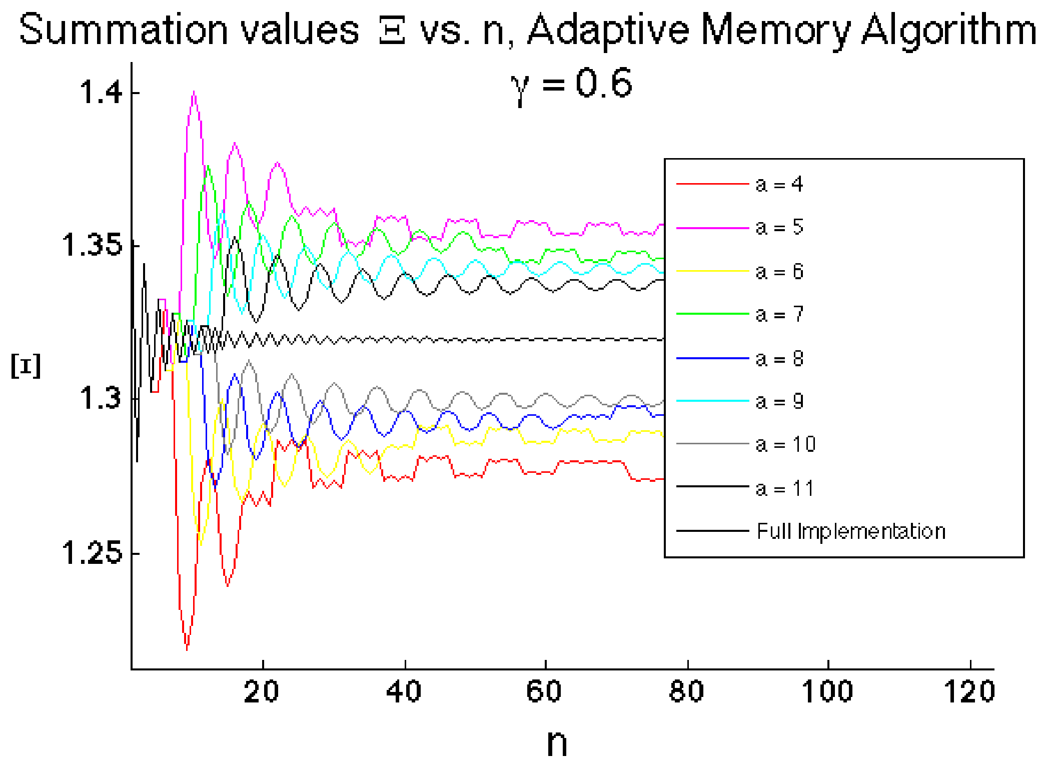
and by virtue of the nature of the memory function ψ,

$$\left|\psi(\gamma, M_{\eta+1})\right| \leq \left|\psi(\gamma, M_{\eta})\right|$$

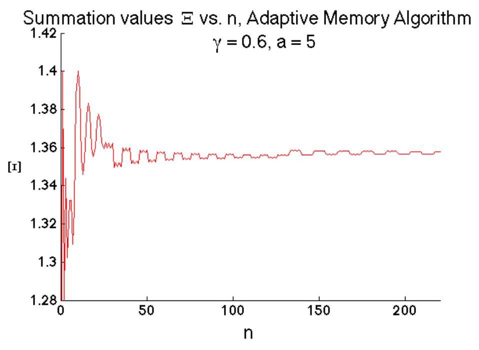
which makes Eq. 4.2.27 an alternating series that, in the limit $\eta_{max} \to \infty$, converges. However, η never approaches infinity in any of these sums, and depending on the parity of a and whether or not M indices repeat parity or switch parity between the end of one interval and the start of another, the next interval can see the sum Ξ oscillating around a different equilibrium value (see discontinuities in Fig.4.2.2B,C). In many cases it seems that as $n \to \infty$ and the amplitude of oscillations becomes smaller and smaller for each subsequent interval, we do approach a limit or at least a narrow range of values. However, all of these complications make it difficult to further simplify the expression for Ξ as an analytical expression.

On the other hand we do note that because the modulus of the memory function $|\psi|$ decreases quickly as n increases, especially for the first few terms, it is the value of the first summation term in Eq. 4.2.25 that largely determines the final value of Ξ (and in turn B_{adap}). Let $\Xi_{approx} = \sum_{m=0}^{a} \psi(\gamma, m)\,(-1)^m$. We see from Table 4.1 that for various values of a, the error between Ξ (after it approaches a clear equilibrium, which occurs at least by $n \sim 200$ steps in most cases) and Ξ_{approx} is within 2%, and decreases as a increases, as expected. Since we are unable to extract a simple analytical expression for Ξ, we will instead use Ξ_{approx} which, while numerically based, is accurate, simple, and quick to calculate for a reasonable range of a value.

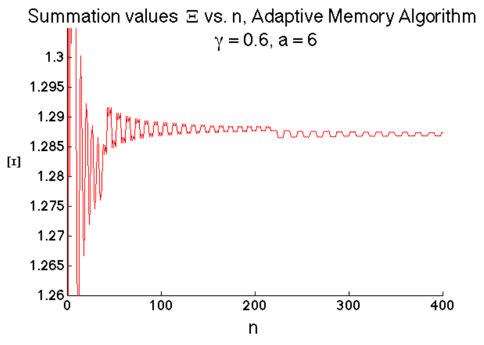
We will now show with numerical examples how B_{adap} varies with γ, n, a, and



A)



B)



C)

Figure 4.2.2: **A)**For a range of values for a, we plot Ξ as a function of time step n. **B,C)** Ξ for $a = 5$, $a = 6$. There are several obvious discontinuities in the pattern of the oscillations, which occur at the breaks between consecutive s intervals.

Table 4.2: $B(\gamma,n)_{full}$ and $B(\gamma,n,a)_{adap}$ values as parameters a,n are varied.

$\gamma = 0.6$		n			
		100	1000	10000	100000
	4	0.196072	0.195852	0.195848	0.195848
a	7	0.185494	0.185568	0.185584	0.185584
	15	0.187897	0.187877	0.187881	0.187880
	$B_1(\gamma,n)$	0.189495	0.189466	0.189465	0.189465
	$B(\gamma)_{full}$	0.189465			

Table 4.3: $B_{full}(\gamma,n)$ and $B_{adap}(\gamma,n,a)$values as parameters a,n are varied.

$\gamma = 1.2$		n			
		100	1000	10000	100000
	4	0.267001	0.271656	0.271833	0.272202
a	7	0.308339	0.309281	0.306723	0.306786
	15	0.300969	0.300604	0.300355	0.300456
	$B_1(\gamma,n)$	0.286279	0.287032	0.287152	0.287171
	$B(\gamma)_{full}$	0.287175			

compare to B_{full}, derived in Section 2.1.2. Tables 4.2 and 4.3 show that the values for the B functions change minutely as $n \to \infty$. It is also clear that for $B_{adap}(\gamma,n,a)$, the dependence on a is not very strong, but still noticeable. And as expected, the bounds depend significantly on the order of the fractional operator, γ. Still, for both the full implementation and the adaptive memory algorithms, it is easy to precompute B based on γ (and a in the adaptive case) and then decide how to define parameters $\Delta t, \Delta x$, and Δy as needed to maintain stability during the simulation. Since the stability dependence on n is weak in all cases, the length of simulation has little influence on the parameters. It is also clear from the table results that B_{adap} is pretty close to B_{full} in many cases, suggesting that the adaptive memory algorithm significantly saves on computational time and power, without adverse effect on the stability regime of the parameters.

4.2.3 Stability Results

Here we will show that our stability analyses agree well with numerical simulations. For all plots in the results section, we are using the adaptive memory algorithm defined in Eq. 4.2.21 and have set $\alpha = 50\frac{units^2}{s^\gamma}$, $\Delta x = \Delta y = 10$ *units*, and base interval $a = 8$ (which insures a reasonable level of accuracy with maximum error of the adaptive memory algorithm remaining under 1% for the duration of the simulation). Our initial condition is a narrow two-dimensional Gaussian: $u(\vec{x},0) = e^{-x^2/2\sigma_1^2}e^{-y^2/2\sigma_2^2}$, $\sigma_1 = 5\ units$, $\sigma_2 = 5\ units$.

We begin by setting $\gamma = 0.6$, which puts our simulations in the subdiffusion regime where the true solution to the fractional diffusion equation is composed of only decaying modes). With the parameters we have already set, we find that according to Eq. 4.2.26, $r \leq 0.1929$ using Ξ, and $r \leq 0.1905$ if we use Ξ_{approx}. Our only degree of freedom is time step Δt.

In Fig. 4.2.3A-B, we have chosen $\Delta t = 0.1s$ which puts r in the stable regime and relatively far from the bound. Both the intensity maps and surface plots confirm that the numerical solution is clearly stable at early and late time points in the simulation, as there is no oscillatory behavior.

In Fig. 4.2.3C-D, we have chosen $\Delta t = 0.2s$, which puts r in the stable regime, but close to the boundary. The plots show that there is an oscillatory component that is evident early in the simulation. However, at a later time in the same simulation, the oscillatory component has been suppressed and the solution has decayed in time and space as expected of subdiffusive behavior. It is reasonable that the oscillatory component was present in the beginning of the simulation since r is close to the boundary between the stable and unstable regimes. However r is still strictly in the stable regime and this is verified by the fact that the oscillations do decay with time. Recalling the discussion in Section 4.2 relating to high frequency Fourier modes, we can see clearly from these plots that the oscillations causing instability are indeed on the order of the grid size.

In Fig. 4.2.4A-B, we have chosen $\Delta t = 0.21s$, which puts r in the unstable region, but close to the boundary between stable and unstable regimes. As expected of the unstable regime, the numerical solution has oscillatory behavior that persists as the simulation continues. But because r is close to the boundary with the stable regime, the oscillations do not increase drastically overwhelm the decaying modes, indicating that the solution is only mildly unstable.

In Fig. 4.2.4C-D, we have chosen $\Delta t = 0.3s$, which puts r clearly in the unstable region and further from the boundary. The plots confirm this by showing that the oscillatory behavior of the solution grows drastically with time and completely overwhelms the decaying modes of the true solution, as observed by scale axes, which are orders of magnitude larger than that of the true solution.

Next, we will consider the superdiffusion cases with $\gamma = 1.2$. According to Eq. 4.2.26, for a stable numerical solutions, $r \leq 0.27105$ using Ξ, and $r \leq 0.2809$ if we use Ξ_{approx}. In addition, the differential equation is in the superdiffusion regime. It is less straightforward to characterize this regime because the true solution has oscillatory components (which is logical because as γ increases towards 2, we approach the classical wave equation), in addition to decaying modes characteristic of diffusion. However, we can still observe the difference in behavior between unstable and stable numerical solutions.

In Fig. 4.2.5A, we have chosen $\Delta t = 0.4s$, which puts r in the stable region. The plots show a numerical solution with low spatial frequency oscillatory components (oscillating on a much larger scale than the scale of the grid elements), and the presence of high frequency components characteristic of noise, is absent. In 4.2.5B, $\Delta t = .55s$, which puts r still in the stable region but close to the boundary between stable and unstable regions. The presence of the high frequency noise components is apparent, but even late into the simulation, they do not overwhelm the true solution and are clearly bounded, which verifies stability. Finally, in 4.2.5C-D, we have chosen $\Delta t = 0.7s$ so r is clearly in the unstable regime. The plots show that the high frequency oscillatory behavior of the solution grows drastically and unbounded with time, which, as noted in Fig. 4.2.4, is a good indicator of numerical instability.

All our results verify the conclusions made during our stability analysis in Sections 2.1.2 and 2.2. It's also clear from Figs. 4.2.3 and 4.2.4 that the boundary between stable and unstable regimes is quite a sharp one, as changing our time step from $\Delta t = 0.2s$ to $\Delta t = 0.21s$ was enough to change our simulations from being stable to unstable.

4.3 Complexity Analysis

In the following sections we analyze the complexity of the full two-dimensional implementation of the fractional diffusion equation, the adaptive memory algorithm, and the linked list alternative version defined in Chapter 3.

4.3.1 Full Implementation

We begin with the pseudocode for the full implementation defined in Eq. 4.2.1. In our pseudocode we generally omit instructions not essential to showing the logical structure of the algorithm and note that the most important aspect of time complexity analysis are data structures involving loops. Lines of code that are run a constant number of times within a loop and are independent of variables like timestep, can be ignored in the overall function in the final big-O result.

From the pseudocode in Algorithm 1 we can see that for each n, the total approximate number of instructions (ignoring things like constant number of instructions) is

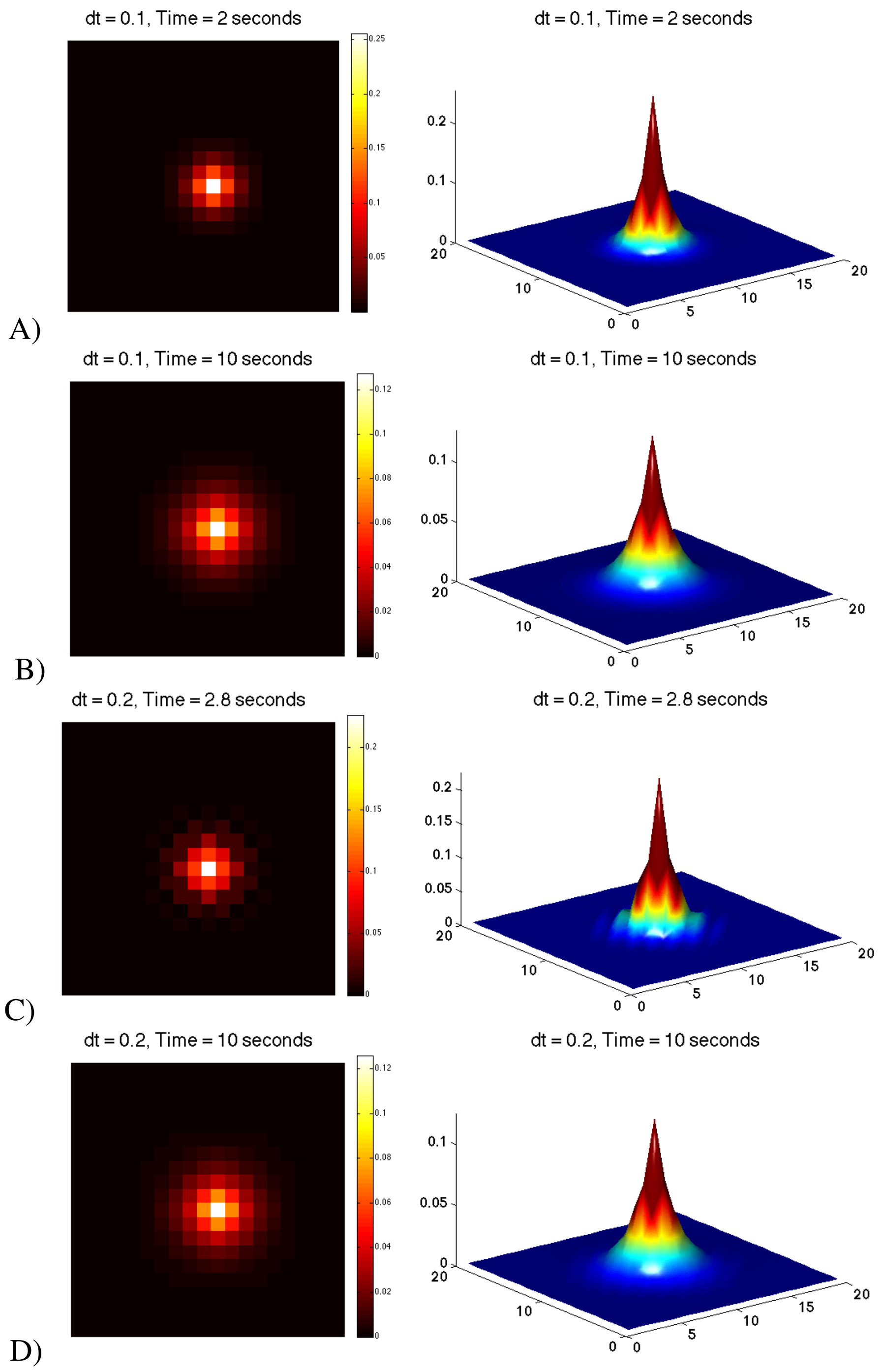
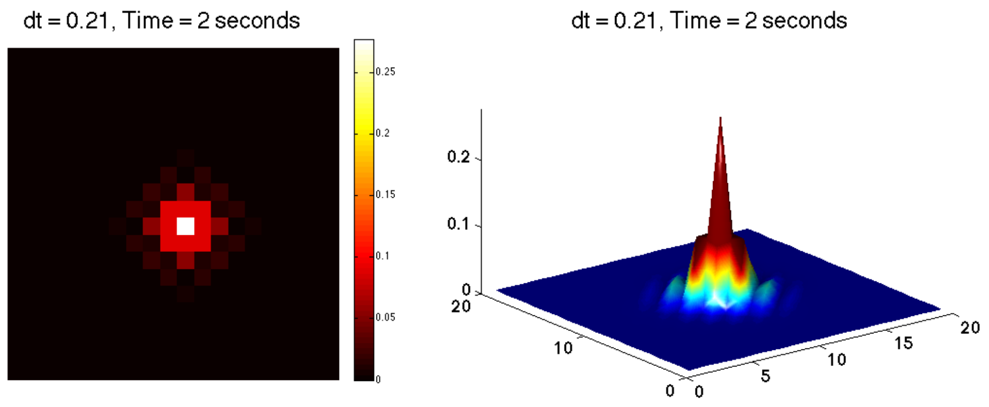
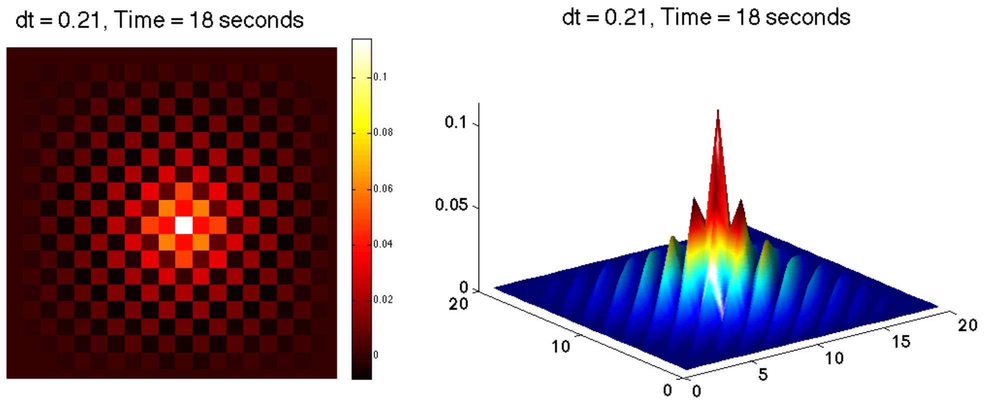


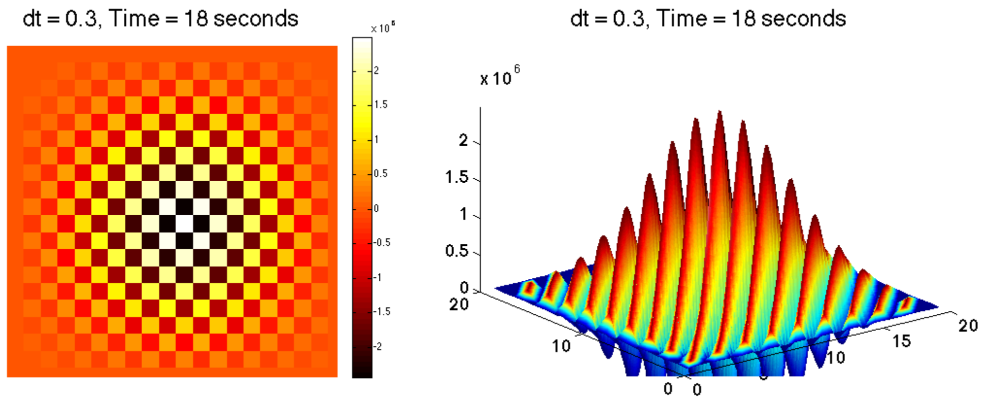
Figure 4.2.3: **Stable Simulations in the Subdiffusion Region**. $\alpha = 50\frac{units^2}{s^\gamma}$, $\Delta x = \Delta y = 10$ *units*, $\gamma = 0.6$,left: 2D intensity plots, right: 3D surface plots. **A-B)** $\Delta t = 0.1$ *s*, *r* is in the stable region and far from the bound, and it is clear that there is no oscillatory component at any point in the simulation. **C-D)** $\Delta t = 0.2s$, *r* in the stable regime, but very close to the boundary. It is clear during the snapshots, that there is a tiny oscillatory component that appears early in the simulation (away from the center of the grid), but decays as the simulation progresses, and so the solution remains bounded.



A)



B)



C)

Figure 4.2.4: **Unstable Simulations in the Subdiffusion Region**. $\alpha = 50 \frac{units^2}{s^\gamma}$, $\Delta x = \Delta y = 10$ *units*, $\gamma = 0.6$, left: 2D intensity plots, right: 3D surface plots. **A-B)** $\Delta t = 0.21s$, r is in the unstable region and close to the boundary. The plots reflect an oscillatory component that appears early in the simulation and is sustained through the duration of the simulation. Even though the oscillations do not increase, and do not overwhelm the decaying modes of the true solution, they do not disappear and would remain even as $n \to \infty$, which is an indication of instability. **C)** $\Delta t = 0.3s$, r is in the unstable region, and far from the bounds between the unstable and stable regions. The plot reflects oscillatory components that not only remain later in the simulation, if we note the scale axis of both plots, we see that those components have completely overwhelmed the decaying modes of the solution and increase unbounded as the simulation progresses. This is a hallmark of strong instability in numerical simulations.

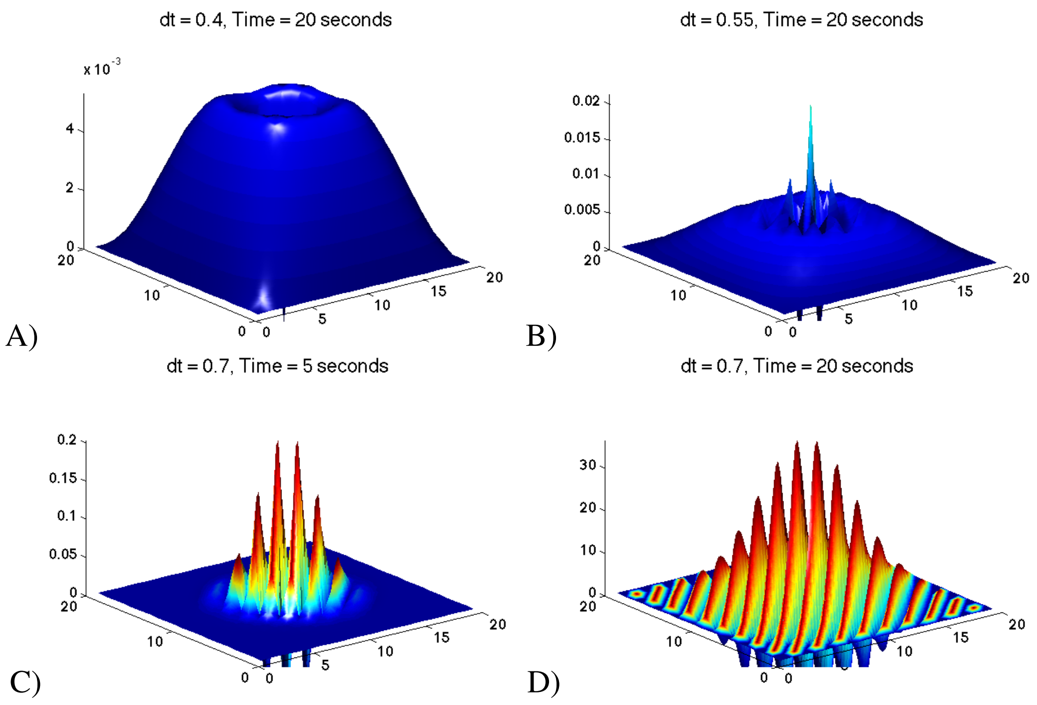


Figure 4.2.5:
Stable and Unstable Superdiffusion Simulations. $\alpha = 50\frac{units^2}{s^\gamma}$, $\Delta x = \Delta y = 10\ units$, $\gamma = 1.2$, left: 2D intensity plots, right: 3D surface plots. **A)** $\Delta t = 0.4s$, r is in the stable regime. Because we are in the superdiffusion regime, the true solution does have oscillatory components alongside decaying modes. However, the numerical solution is obviously bounded, even later in the simulation, and there is no sign of high frequency oscillations (visible on the plots C and D), so the plots seem to verify that the simulation is indeed stable. **B)** $\Delta t = 0.55s$, r is in the stable regime. We can clearly see high frequency components, especially contrasted the same plot at the same time in the simulation, in part A. However, as observed by the scale bar, the high frequency components don't overwhelm the decaying modes of the solution and cause it grow unbounded, so this verifies stability. **C-D)** $\Delta t = 0.7s$, r is in the unstable regime. It is clear from the plots that the high frequency oscillatory components is present early in the simulation, but the decaying modes still dominate the overall solution. However, it's clear that later in the simulation the high frequency oscillations continue to grow unbounded in time, which verifies the unstable nature of the numerical solution.

N_xN_yn where n varies from $1:N$. Therefore:

$$\begin{aligned} total\ instructions &= N_xN_y\left(1+...+N\right) \\ &= N_xN_y\left(\frac{N\left(N+1\right)}{2}\right) \\ &\Rightarrow O\left(N_xN_yN^2\right) \end{aligned}$$

This gives us the growth in execution time as a function of the grid dimension N_x and N_y, as well as the timestep N. While time grows linearly with the size of each spatial dimension of the grid we are using, it grows as the square of the number of timesteps in the simulation, which is not the most efficient result.

Algorithm 1 Full 2D Implementation

```
1: for $n = 1 : N$ do                       ▷ $N$ is the total # of timesteps the algorithm will run
2:     for $j = 1 : N_x$ do                  ▷ iterate over all grid points in x direction
3:         for $l = 1 : N_y$ do              ▷ iterate over all grid points in y direction
4:             for $m = 0 : 1 : n$ do
5:                 Calculate $\psi(\gamma, m)\left(\frac{\alpha}{\Delta x^2}\delta x_{j,l}^{n-m} + \frac{\beta}{\Delta y^2}\delta y_{j,l}^{n-m}\right)$
6:             end for
7:             Calculate $u_{j,l}^{n+1}$
8:         end for
9:     end for
10: end for
```

Here we note that in the one-dimensional case, we can easily vectorize the loop iterating over grid points, increasing efficiency with regards to grid size. However, the time complexity still grows as $O\left(N^2\right)$.

4.3.2 Adaptive Memory Algorithm

In Algorithm 2 we provide the pseudocode for the implementation of the adaptive memory algorithm described in equation Eq. 4.2.21. As before, we omit the instructions detailing the calculations themselves because counting these instructions involve multiplying constants against the number of times the loops are run, and do not affect the form of the big-O expressions in terms of the input variable of interest. As with the full implementation, we note that in the one-dimensional analogue, we can vectorize the loop iterating over spatial gridpoints.

Analysis of the total number of instructions in Part 1 of Algorithm 2 proceeds in exactly the same way as in Section 4.3.1. The total instruction count is approximately

$$\begin{aligned} N_xN_y\left(1+2+...+a\right) &= N_xN_y\frac{a\left(a+1\right)}{2} \\ &\Rightarrow O\left(N_xN_ya^2\right) \end{aligned}$$

Analysis of Part 2 of Algorithm 2 is more complicated because of the additional parameters s_{max}, η_{max}, *and* M_{max}. For loop $\{\text{L1}\}$ (lines 21-23), η_{max} depends on the current s interval, but the maximum value it can be is always $\frac{a^s-a^{s-1}}{2s-1}$. For loop $\{\text{1b}\}$ (lines 24-26), the loop serves to sample timesteps when the current interval $[a^{s-1} : a^s]$, is not evenly divisible by the weight for that interval, $(2s-1)$, and therefore these few timesteps aren't taken into account with loop $\{\text{L2}\}$. The maximum number of times this loop is run is always $\leq 2s-1$. Therefore in the large for loop in Algorithm 2 encompassing lines 13-31, for a given value of n, the number of instructions run is:

$$N_xN_y\left\{a+\sum_{s=2}^{s_{max}}\left(\underbrace{\frac{a^s-a^{s-1}}{2s-1}}_{\{1a\}}+\underbrace{2s-1}_{\{1b\}}\right)\right\} \tag{4.3.1}$$

To simplify this expression further we need to make some assumptions about worst case scenarios. We can simplify the first term $\{\text{1a}\}$ of the summation in Eq. 4.3.1 using telescoping series:

Algorithm 2 Adaptive Memory Algorithm

```
                                                            ▷ Part 1
for $n = 1 : a$ do
    for $j = 1 : N_x$ do                        ▷ iterate over all grid points in x direction
        for $l = 1 : N_y$ do                    ▷ iterate over all grid points in y direction
            for $m = 0 : 1 : n$ do
                Calculate $\psi(\gamma, m)\left(\frac{\alpha}{\Delta x^2}\delta x_{j,l}^{n-m} + \frac{\beta}{\Delta y^2}\delta y_{j,l}^{n-m}\right)$
            end for
            At this location, calculate $u_{j,l}^{n+1}$
        end for
    end for
end for
                                                            ▷ Part 2
for $n = a+1 : N$ do           ▷ $N$ is the total # of timesteps the algorithm will run
    Determine $s_{max}$ as a function $n$
    for $j = 1 : N_x$ do                        ▷ iterate over all grid points in x direction
        for $l = 1 : N_y$ do                    ▷ iterate over all grid points in y direction
            for $m = 0 : 1 : a$ do                      ▷ Calculate Sum 1 (base interval)
                Calculate R1: $\psi(\gamma, m)\left(\frac{\alpha}{\Delta x^2}\delta x_{j,l}^{n-m} + \frac{\beta}{\Delta y^2}\delta y_{j,l}^{n-m}\right)$ ▷ R1 from Eq. 4.2.21
            end for
            for $s = 2 : s_{max}$ do
                for $\eta = 1 : \eta_{max}$ do              ⎫
                    Calculate R2 in Eq. 4.2.21              ⎬ $\{L1\}$
                end for                                     ⎭
                for $p = M_{max} : min\,(a^s, n)$ do        ⎫
                    Calculate R3 in Eq. 4.2.21              ⎬ $\{L2\}$
                end for                                     ⎭
            end for
            Use R1-R3 to calculate $u_{j,l}^{n+1}$
        end for
    end for
end for
```

$$\begin{aligned}\sum_{s=2}^{s_{max}} \frac{a^s - a^{s-1}}{2s-1} &= \frac{a^2-a}{3} + \frac{a^3-a^2}{5} + \ldots + \frac{a^{s_{max}} - a^{s_{max}-1}}{2s_{max}-1} \\ &< \frac{a^2-a}{3} + \frac{a^3-a^2}{3} + \ldots + \frac{a^{s_{max}} - a^{s_{max}-1}}{3} \\ &= \frac{1}{3}\left\{\left(a^2-a\right) + \left(a^3-a^2\right) + \ldots + \left(a^{s_{max}-1} - a^{s_{max}-2}\right) + \left(a^{s_{max}} - a^{s_{max}-1}\right)\right\} \\ &= \frac{1}{3}\left(a^{s_{max}} - a\right)\end{aligned}$$

The second term $\{1b\}$ of the summation can be reduced to

$$\begin{aligned}\sum_{s=2}^{s_{max}} 2s-1 &= 2\sum_{s=2}^{s_{max}} s - \sum_{s=2}^{s_{max}} 1 \\ &= 2\left(\frac{s_{max}\left(s_{max}+1\right)}{2} - 1\right) - \left(s_{max}-1\right) \\ &= s_{max}^2 - 1\end{aligned}$$

The expression in Eq. 4.3.1 can now be reduced to

$$N_x N_y \left(a + \frac{1}{3}a^{s_{max}} - \frac{1}{3}a + s_{max}^2 - 1\right) \tag{4.3.2}$$

s_{max} for a given timestep n is determined by $a^{s_{max}-1} + 1 \leq n \leq a^{s_{max}}$, or $\frac{ln(n)}{ln(a)} \leq s_{max} \leq \frac{ln(n)}{ln(a)} + 1$. Since we are concerned with worst case scnearios, we use the maximum value of this interval, $s_{max} = \frac{ln(n)}{ln(a)} + 1$ and substitute into Eq. 4.3.2 . Now we must consider the for loop encompassing lines 13-31 in Algorithm 2 and how the total number of instructions relates to the total number of timesteps N:

$$N_x N_y \sum_{n=a+1}^{N} \left(\underbrace{\frac{2}{3}a}_{\{2a\}} + \underbrace{\frac{1}{3}a^{ln(n)/ln(a)}a}_{\{2b\}} + \underbrace{\left(\frac{ln(n)}{ln(a)}\right)^2}_{\{2c\}} + \underbrace{2\frac{ln(n)}{ln(a)}}_{\{2d\}} \right) \tag{4.3.3}$$

We now simplify the four components of the summation in Eq. 4.3.3. For $\{2a\}$,

$\sum_{n=a+1}^{N} \frac{2}{3}a = (N-(a+1))\frac{2}{3}a$. Since a is a constant parameter chosen by the user prior to the simulation, the dominant term in this expression is clearly N.

For {2b}:

$$\begin{aligned}
\frac{a}{3}\sum_{n=a+1}^{N} a^{\frac{ln(n)}{ln(a)}} &= \frac{a}{3}\sum_{n=a+1}^{N} a^{\frac{log_a n}{log_a e}\frac{1}{ln a}} \\
&= \frac{a}{3}\sum_{n=a+1}^{N} n \\
&= \frac{a}{3}\left(\frac{N^2+N}{2} - (1+2+...+a)\right)
\end{aligned}$$

The dominant term here is clearly N^2.

For {2c} in equation Eq. 4.3.3:

$$\begin{aligned}
\frac{1}{(ln(a))^2}\sum_{n=a+1}^{N}(ln(n))^2 &= \frac{1}{(ln(a))^2}\left[(ln(a+1))^2 + (ln(a+2)^2 + ... + (ln(N))^2\right] \\
&\leq \frac{1}{(ln(a))^2}\left[(ln(N))^2 + (ln(N))^2 + ... + (ln(N))^2\right] \\
&= \frac{1}{(ln(a))^2}\left[(N-a)(ln(N))^2\right]
\end{aligned}$$

This expression in big-O notation is $O\left(N\left(ln(N)\right)^2\right)$. Alternatively, we can simplify {2c} using integration by parts, to get the same result.

For {2d}:

$$\begin{aligned}
\frac{2}{ln(a)}\sum_{n=a+1}^{N} ln(n) &= \frac{2}{ln(a)} ln\left[(a+1)(a+2)...(N)\right] \\
&= \frac{2}{ln(a)} ln\left(\frac{N!}{a!}\right) \\
&= \frac{2}{ln(a)}\left(ln(N!) - ln(a!)\right)
\end{aligned}$$

Using Stirling's approximation our final expression is

$$\frac{2}{ln(a)}\left(Nln(N) - N + O\left(ln\left(N\right)\right) - ln(a!)\right) \tag{4.3.4}$$

which is dominated by the $Nln\left(N\right)$term.

Combining the simplified expressions for {2a}-{2d} in Eq. 4.3.3, it is clear that the N^2 term dominates the behavior of the overall expression as N becomes large, and so we conclude that the worst case behavior of the adaptive step algorithm, is bounded by $O\left(N_xN_yN^2\right)$. While the adaptive time step algorithm improves raw execution time in relation to the full memory implementation, the complexity analysis shows that when it comes to scaling with large N, the algorithm is as inefficient as the full implementation.

4.3.3 Linked List Adaptive Timestep Algorithm

In Chapter 3 we describe an alternative version of the adaptive memory algorithm which is based on a power law that enables us to eliminate past history timepoints that will never again need to be referenced, thus saving us execution time and memory. Here we summarize the algorithm and refer to Chapter 3 for additional background technical details and derivation.

$$\frac{u_{j,l}^{n+1} - u_{j,l}^{n}}{\Delta_t^{\gamma}} = \sum_{\left\{u_{j,l}^{i} \in U^n\right\}} \psi(\gamma, n-i) w^i \left(\frac{\alpha}{\Delta x^2}\delta x_{j,l}^{i} + \frac{\beta}{\Delta y^2}\delta y_{j,l}^{i}\right) \tag{4.3.5}$$

$$w^{n+1} = 1$$

$$W^{n+1} := \left\{w^i \in W^n\right\} + \left\{w^{n+1}\right\}$$

$$U^{n+1} := \left\{u^i \in U^n\right\} + \left\{u^{n+1}\right\}$$

Table 4.4: Linked list data for timestep $n = 25$

Timestep	0	4	8	12	14	16	18	20	22	23	24	25
Weight	$2^2 = 4$	4	4	$2^1 = 2$	2	2	2	2	$2^0 = 1$	1	1	1

where the data associated with each timestep n is stored via the elements constituting a doubly linked list. Additionally, when there are more than η points in the set W^i, of a given weight, the elements of the set are condensed according to Algorithm 4.1 in Chapter 3. As in the previous sections, we present the pseudocode for the linked list implementation, shown in Algorithm 3.

Algorithm 3 Linked List Implementation

```
1: for $n = 1 : N$ do                                  ▷ $N$ is the total # of timesteps the algorithm will run
2:     for $j = 1 : N_x$ do                             ▷ iterate over all grid points in x direction
3:         for $l = 1 : N_y$ do                         ▷ iterate over all grid points in y direction
4:             for $node$ = 1:length(linked list) do                                              ⎫
5:                 From each node calculate $\psi(\gamma, n-i)w^i\left(\frac{\alpha}{\Delta x^2}\delta x^i_{j,l} + \frac{\beta}{\Delta y^2}\delta y^i_{j,l}\right)$  ⎬ $\{L1\}$
6:             end for                                                                           ⎭
7:             Calculate $u^{n+1}_{j,l}$
8:         end for
9:     end for
10:    If-else statements to determine whether we need to condense the linked list
       according to Algorithm 3.4.1 in Chapter 3. If so:
11:    while it continues to be necessary to condense the linked list do   ⎫
12:        Constant number of instructions                                 ⎬ $\{L2\}$
13:    end while                                                           ⎭
14: end for
```

To determine how many times loop $\{L1\}$ runs in relation to timestep n, we can consider the following.

For a given timestep n, we are interested in a summation of terms involving weights (powers of two), each set of which never exceeds η nodes. For example, for timestep $n = 25$ and $\eta = 5$, Table 4.4 shows the timestep and weight for all nodes currently in the linked list at this point in the simulation.

Since the dynamically changing weight is directly related to how often the history of the simulation is sampled, we can construct the following relationship between time step and weight based on the data in Table 4.4:

$$2^0 \cdot 5 + 2^1 \cdot 5 \le n+1 = 26 \le 2^0 \cdot 5 + 2^1 \cdot 5 + 2^2 \cdot 5$$

The inequality arrives from the fact that a particular weight category has η or fewer nodes.

For a general timestep n we can write the inequalities in terms of geometric series:

$$\sum_{l=0}^{x-1} 2^l \le \frac{n+1}{\eta} \le \sum_{l=0}^{x} 2^l \tag{4.3.6}$$

where $x+1$ is the number of weight categories represented in the linked list at timestep n. In the example in Table 4.4, the number of weight categories is 3, with weights 2^0, 2^1, and 2^2. Substituting the geometric series with their total sums, we can rewrite equation Eq. 4.3.6 as

$$\begin{gathered} x \le log_2\left(\frac{n+1}{\eta}+1\right) \le x+1 \Rightarrow \\ log_2\left(\frac{n+1}{\eta}+1\right) \le x+1 \le log_2\left(\frac{n+1}{\eta}+1\right)+1 \end{gathered} \tag{4.3.7}$$

Where $x+1$ is the number of weight categories.

We now return to Algorithm 3 and refer to loop {L1}. The maximum possible number of times this loop runs for any given timestep n, is η times the maximum number of weight categories at that time, which, using Eq. 4.3.7, is

$$\eta\left(log_2\left(\frac{n+1}{\eta}+1\right)+1\right)$$

Therefore the total number of instructions in lines 2-9 of Algorithm 3 is $N_xN_y\eta\left(log_2\left(\frac{n+1}{\eta}+1\right)+1\right)$. We now look at loop {L2} (lines 11-13). This while loop runs whenever we need to condense the linked list based on the number of nodes belonging to a particular weight category, exceeding the user-set parameter η. The maximum number of times this may run within a single timestep, is the number of weight categories in the linked list. For example, if the number of nodes with weight 2^0 exceeds η, we delete the second "least" node in the set of nodes with this weight, and double the weight of the "least" element (node) in this same set, to 2^1. (See the derivation of Algorithm 3.4.1 in Chapter 3 for details about the ordering of these sets). For some particular timestep this may mean that the number of nodes with weight 2^1 now exceeds η and we continue to condense in the same fashion until we have addressed all weight categories currently represented in the linked list. The maximum number of weight categories is given by Eq. 4.3.7, and this is the maximum number of times loop {L2} is run for a given time step n. Now we can combine all loops in Algorithm 3 and write that the maximum number of instructions (ignoring constants) is given by

$$\sum_{n=1}^{N}\left[N_xN_y\eta\left(log_2\left(\frac{n+1}{\eta}+1\right)+1\right)+log_2\left(\frac{n+1}{\eta}+1\right)+1\right]$$
$$=(N_xN_y\eta+1)\left[\underbrace{\sum_{n=1}^{N}log_2\left(\frac{n+1}{\eta}+1\right)}_{\{1\}}+N\right] \tag{4.3.8}$$

We can simplify the summation term {1}:

$$\sum_{n=1}^{N}log_2\left(\frac{n+1+\eta}{\eta}\right)=\sum_{n=1}^{N}\left(log_2\left(n+1+\eta\right)-log_2\left(\eta\right)\right)$$

With a change of variable $d = n+1+\eta$ we can rewrite the summation as

$$\begin{aligned}\sum_{d=2+\eta}^{N+1+\eta=N'} log_2(d) - Nlog_2(\eta) &\leq \sum_{d=1}^{N'} log_2(d) - Nlog_2(\eta)\\ &= log_2(1) + log_2(2) + ... + log_2(N') - Nlog_2(\eta)\\ &= log_2(N'!) - Nlog_2(\eta)\\ &= N'log_2(N') - N' + O(log_2(N')) - Nlog_2(\eta)\end{aligned}$$

We can now write Eq. 4.3.8 as

$$\text{total instructions} \leq (N_xN_y\eta + 1)\left[N'(log_2(N')) - N' + O(log_2(N')) - Nlog_2(\eta) + N\right]$$

It is clear that the dominant term here is $N'(log_2(N'))$and since N' is simply equal to N offset by a constant, we can conclude the overall complexity is

$$O(N_xN_yNlog_2(N))$$

Compared to the full and adaptive step algorithms analyzed in this section, the linked list implementation is clearly the most efficient in terms of how execution time scales as number of timesteps N, grows large. An added benefit (as noted in Chapter 3, Section 4) would be the lowered memory requirements from $O(N)$ (data stored for all N time steps) to $O(log_2N)$, since we no longer require keeping all time points in the history of the simulation.

As before, we also note that in the one-dimensional analogous case of the linked list pseudocode, the loop iterating over spatial grid points can be vectorized.

4.3.4 Empirical Results

We verify our theoretical complexity results with simulated data. In Fig. 4.3.1 we show computation run times for simulations of various number of steps N and demonstrate the complexity of the three numerical methods explored in this section. For each algorithm we used a one-dimensional analog of the two-dimensional verison described in the pseudocode in this section. For all algorithms we set $\gamma = 0.8$, $D = \frac{units^2}{s^\gamma}$, $dt = 0.1s$, $dx = 0.69\ units$. We used adaptive step parameter and η values of 20.

Data was fitted to trendlines using Matlab's curve fitting toolbox. For the data corresponding to the full and adaptive memory implementations, the best fit lines were polynomials of second order (modeled with three coefficients). For the linked list implementation the trendline with minimum number of coefficients was a $Nlog_2N$ function.

- full implementation simulation time = $p_1N^2 + p_2N + p_3$ with $p_1 = 1.106e-5$, $p_2 = 5.5e-4$, $p_3 = -0.0423$ and goodness of fit measure $R-squared = 0.9999$

- adaptive memory simulation time = $p_1N^2 + p_2N + p_3$ with $p_1 = 2.671e-5$, $p_2 = 1.441e-3$, $p_3 = -0.05873$ and goodness of fit measure $R-squared = 0.9999$

- linked list simulation time = $p_1Nlog_2(N)$ with $p_1 = 7.047e-4$ and goodness of fit measure $R-squared = 0.9992$. This is the simplest trendline that was found to be a good fit. There were other functions that fit the data for the linked list implementation but required additional coefficients.

The data fitting and empirical data in Fig. 4.3.1 thus supports the theoretical analysis done in this section.

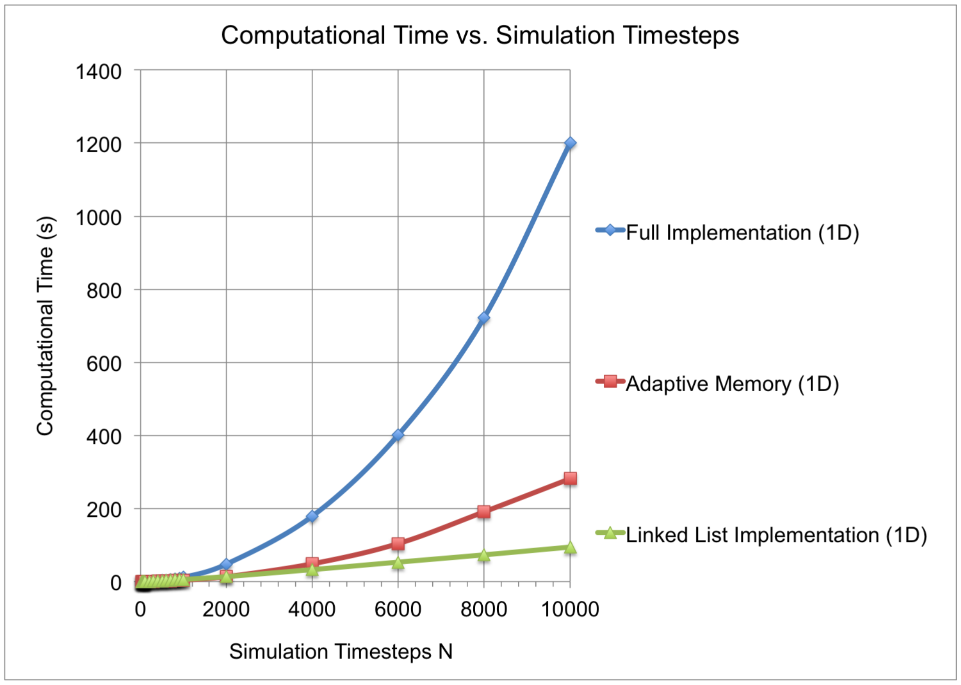


Figure 4.3.1: Empirical data verifying theoretical complexity results. For all simulations we set $\gamma = 0.8$, $D = \frac{units^2}{s^\gamma}$, $dt = 0.1s$, $dx = 0.69\ units$. We used adaptive step adaptive step parameter and η values of 20.

4.4 Error

In the last section we found that the algorithmic complexity for the linked list implementation is considerably more efficient than the full and adaptive memory implementations. However, with numerical algorithms, considerable advantages in one area usually come with some drawback or cost, in another. In this case, the accuracy of the linked list implementation is the drawback; in Fig. 4.4.1 we compare the error of all three explicit algorithms discussed in this chapter (full implementation, adaptive memory, linked list implementation based on power law) and see that while the error for the first two converge over time, the error of the linked list implementation jumps in discrete intervals. This corresponds to the instances when the a new weight category is introduced and the linked list is condensed according to Eq. 4.3.5. The rate at which these jumps occur, is given by parameter η (the larger this value is, the slower the error increases). We leave as an open problem the analysis of the error resulting from the linked implementation (whether it converges over time and its exact dependence on η). Even in the case that error does not converge, one can set the value of η depending on the required length of the simulation and how much percentage error is willing to be tolerated during the simulation.

4.5 Conclusion

We have successfully characterized the stability of two finite difference algorithms used in solving the time-fractional diffusion equation. We provide our rationale for selecting a von Neumann analysis while also exploring a few alternative interpretations and approaches to analyzing stability. Using a von Neumann analysis we have developed bounding expressions for parameters like time step, spatial discretization and grid size that can be used to appropriately set parameters to ensure accurate simulations that reflect

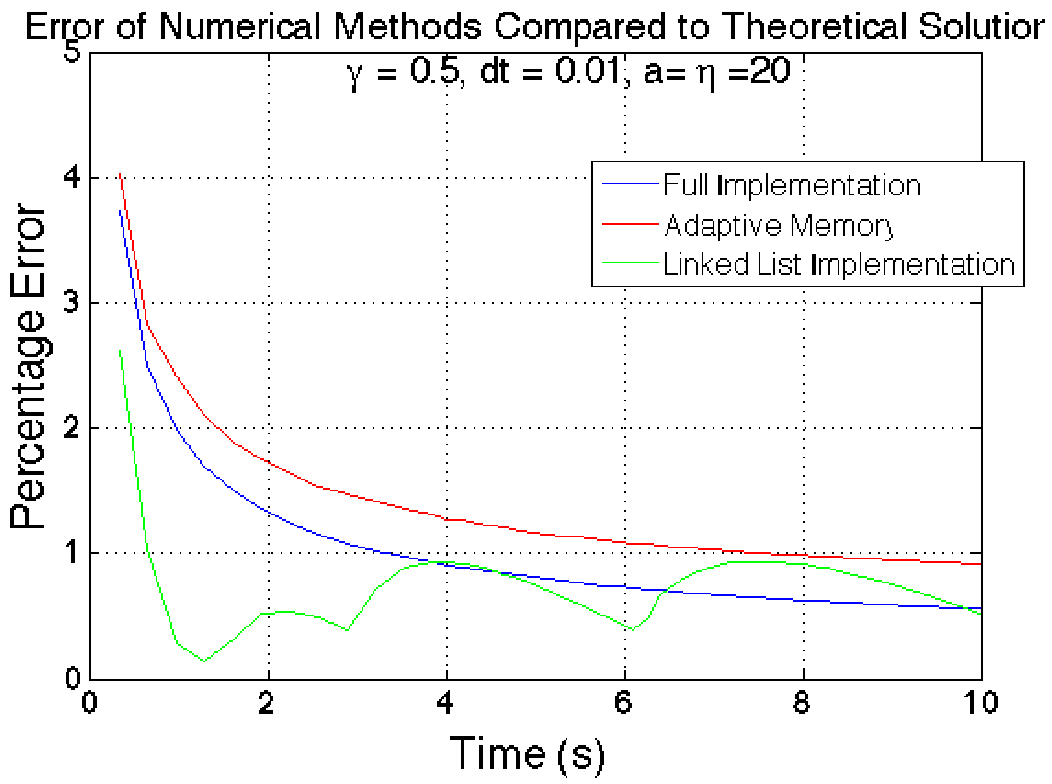


Figure 4.4.1: Error comparison between one-dimensional versions of all the explicit algorithms discussed in this chapter. Error is compared to the fundamental solution described in Chapter 2, at the location of the initial condition (center of the spatial axis), which is the most quickly changing point on the spatial domain. It is clear that the error for the full implementation and adaptive memory converge over time, but the error for the linked list implementation is seen to decay and then jump in discrete intervals.

the true solution of the fractional diffusion to be solved. Our simulation results verify that our bounding expressions resulting from our stability analysis are accurate and valid. We also note here that the stability analysis of the linked list implementation should also be explored, and we leave this as an open problem for further consideration.

We have also successfully characterized the algorithmic complexity of all three finite difference algorithms introduced in Chapter 3. We find that the full and adaptive step algorithms have a big-O complexity of $O\left(N^2\right)$ while the linked list scheme performs much better with $O\left(Nlog_2N\right)$ complexity. We compare our analytical results with empirical data and find they are in good agreement.

Chapter 4, in part, is currently being prepared for submission for publication of the material. Nirupama Bhattacharya and Gabriel A. Silva. The dissertation author was the primary investigator and author of this paper.

Chapter 5

An Efficient Finite Difference Approach to Solving the Time-fractional Diffusion Equation

5.1 Introduction

As discussed in Chapter 3, there has been experimental evidence suggesting that diffusion of ATP from retinal astrocytes, has a diffusion profile with peaked center [38]. One potential mechanism explaining such such a profile is anomalous subdiffusion, which, as we have explored in earlier chapters, can be modeled with a time-fractional diffusion equation. Our interest was in applying the computationally efficient approaches developed in the last chapter, to model the diffusion of material in a cellular network with biophysically accurate parameters, including a high diffusion coefficient. However, we soon found that a drawback to these methods is that those approaches are based on explicit schemes that are only conditionally stable and first order accurate in time. Using parameters such as large diffusion coefficients, required us to use extremely small time steps in order to maintain a stable simulation. Therefore, we developed a more stable numerical scheme that can still take advantage of the speed and efficiency increases of the methods developed in Chapter 3.

In this chapter we apply the implicit Crank-Nicholson (CN) numerical scheme, in combination with an operator-splitting method, to develop an efficient numerical method for solving the two-dimensional time-fractional diffusion equation; this scheme has an expanded stability regime that enables us to simulate problems with biologically accurate parameters such as high diffusion coefficients, without requiring a prohibitively small simulation time step.

5.2 Methods

As established in Chapter 2, anomalous diffusion is characterized by a nonlinear relation between the mean square displacement of a particle, and time:

$$< x^2(t) > \sim \alpha t^{\gamma}, \gamma \neq 1 \qquad (5.2.1)$$

$0 < \gamma < 1$ denotes the subdiffusion regime and $1 < \gamma < 2$ denotes superdiffusion. If $\gamma = 1$, we recover the linear relationship that is the basis of classical diffusion. Equation 5.2.1 is related to the breakdown of the Central Limit Theorem which shows that the Brownian motion random walk at the continuity limit, is the classical diffusion equation. At the microscopic level, equation 5.2.1 represents random walk processes where a particle's jumps at each timestep, and the wait time between jumps, are drawn from probability distribution functions with long tails, which are distinct from standard Gaussian distributions that are the basis of classical diffusion. These long-tailed probability distribution functions are integral to continuous time random walk (CTRW) models, from which we can derive (see [36, 57] for detailed derivations) the homogenous fractional diffusion equation given by:

$$\frac{\partial^\gamma u(\vec{x},t)}{\partial t^\gamma} = \alpha \nabla^2 u(\vec{x},t) \tag{5.2.2}$$

where α is the diffusion coefficient (in $\frac{spatial\ unit^2}{time\ unit^\gamma}$), and u is the concentration as a function of space and time. We will begin with the homogenous equation without additional terms, but will later consider the addition of $f(u(\vec{x},t))$, a general source or sink term, to form the general inhomogenous fractional diffusion equation

$$\frac{\partial^\gamma u(\vec{x},t)}{\partial t^\gamma} = \alpha \nabla^2 u(\vec{x},t) + f(u(\vec{x},t)) \tag{5.2.3}$$

As in Chapter 3, we rearrange equation 5.2.2 in terms of the continuous time Riemann-Liouville fractional derivative operator, as defined in [40, 39, 57].

$$\frac{\partial u}{\partial t} = \alpha D_t^{1-\gamma} \nabla^2 u \tag{5.2.4}$$

However, the Riemann-Liouville definition of the fractional derivative operator is given in terms of an integral, and is not in a form that can be easily manipulated for computational purposes without the aid of quadrature algorithms. We therefore make use of the Grünwald-Letnikov definition of the fractional derivative operator

$$\begin{aligned} D_t^p u(\vec{x},t) &= \lim_{\tau \to 0} \tau^{-p} \sum_{m=0}^{n=t/\tau} (-1)^m \left(\begin{array}{c} p \\ m \end{array} \right) u(\vec{x}, t - m\tau) \Rightarrow && (5.2.5) \\ &= \tau^{-p} \sum_{m=0}^{n=t/\tau} (-1)^m \left(\begin{array}{c} p \\ m \end{array} \right) u(\vec{x}, t - m\tau) + O(\tau) && (5.2.6) \end{aligned}$$

where τ is the discretization step in time. The Riemann-Liouville and Grünwald-Letnikov definitions approach the same value if $u(\vec{x},t)$ is continuous and its first derivative is integrable on the interval $[0,t]$ [57]. Since we are assuming that the solution to a fractional diffusion equation used to model many physical processes, is sufficiently smooth and therefore meets the requirements of continuity and integrability of the derivative, and so we consider it appropriate to use the Grünwald-Letnikov definition of the fractional derivative operator.

Within this premise we can define the binomial term in equation 5.2.5 as a 'memory' function $\psi(\gamma, m) = (-1)^m \begin{pmatrix} 1-\gamma \\ m \end{pmatrix}$. This memory function can also be written as a recursive relation in terms of the Γ function, so that it can be fully computed before the time course simulation, saving valuable computational time. See Chapter 3 for a more detailed derivation of $\psi(\gamma, m)$. Our diffusion equation can now be expressed in terms of the 'memory' function $\psi(\gamma, m)$:

$$\frac{\partial u}{\partial t} = \lim_{\tau \to 0} \tau^{\gamma-1} \alpha \sum_{m=0}^{t/\tau} \psi(\gamma, m) \nabla^2 u(\vec{x}, t - m\tau) \Rightarrow \tag{5.2.7}$$

$$= \tau^{\gamma-1} \alpha \sum_{m=0}^{t/\tau} \psi(\gamma, m) \nabla^2 u(\vec{x}, t - m\tau) + O(\tau) \tag{5.2.8}$$

5.2.1 Crank-Nicholson in Time

Progressing from this point involves approximating the time derivative and the spatial Laplacian, and here we consider the implicit Crank-Nicholson scheme as an alternative to an explicit scheme, in order to yield a more stable and more accurate numerical solution.

The general Crank-Nicholson scheme utilizes the second order accuracy given by a centered difference scheme. The derivative with respect to time

$$\frac{\partial u}{\partial t} \approx \frac{u(\vec{x}, t + \Delta t) - u(\vec{x}, t)}{\Delta t} \tag{5.2.9}$$

is a difference centered around $t + \frac{\Delta t}{2}$. This approximation gives us an error proportional to the square of Δt, as shown by the Taylor series expansion of function u around $t+\frac{\Delta t}{2}$:

$$\begin{aligned} u\left(x,y,t+\frac{\Delta t}{2}+\frac{\Delta t}{2}\right) &= u\left(x,y,t+\frac{\Delta t}{2}\right)+\frac{\Delta t}{2}u^{(1)}\left(x,y,t+\frac{\Delta t}{2}\right) \\ &\quad +\frac{\left(\frac{\Delta t}{2}\right)^2}{2!}u^{(2)}\left(x,y,t+\frac{\Delta t}{2}\right) \\ &\quad +\frac{\left(\frac{\Delta t}{2}\right)^3}{3!}u^{(3)}\left(x,y,t+\frac{\Delta t}{2}\right)+\dots \qquad (5.2.10) \\ u\left(x,y,t+\frac{\Delta t}{2}-\frac{\Delta t}{2}\right) &= u\left(x,y,t+\frac{\Delta t}{2}\right)-\frac{\Delta t}{2}u^{(1)}\left(x,y,t+\frac{\Delta t}{2}\right) \\ &\quad +\frac{\left(\frac{\Delta t}{2}\right)^2}{2!}u^{(2)}\left(x,y,t+\frac{\Delta t}{2}\right) \\ &\quad -\frac{\left(\frac{\Delta t}{2}\right)^3}{3!}u^{(3)}\left(x,y,t+\frac{\Delta t}{2}\right)+\cdots \qquad (5.2.11) \end{aligned}$$

where $u^{(n)}$ refers to the nth derivative of u with respect to time. Subtracting equation 5.2.11 from equation 5.2.10 and solving for the first derivative of u gives us

$$u^{(1)}\left(x,y,t+\frac{\Delta t}{2}\right)=\frac{u(x,y,t+\Delta t)-u(x,y,t)}{\Delta t}+R_n$$

where R_n is the Taylor series remainder, or truncation, that is $O\left(\Delta t^2\right)$.

We also use centered difference approximations for the second order derivatives in space, which are well known to give second order truncation errors in each spatial direction [Haberman]

$$\begin{aligned} \frac{\partial^2 u}{\partial x^2} &= \frac{u(x+\Delta x,y,t)-2u(x,y,t)+u(x-\Delta x,y,t)}{\Delta x^2}+O\left(\Delta x^2\right) \\ \frac{\partial^2 u}{\partial y^2} &= \frac{u(x,y+\Delta y,t)-2u(x,y,t)+u(x,y-\Delta y,t)}{\Delta y^2}+O\left(\Delta y^2\right) \end{aligned}$$

In the scope of this chapter we are interested in solving the fractional diffusion equation on an equally spaced grid in both directions and therefore we will set $\Delta x=\Delta y=\Delta$ for the spatial discretization symbol.

Looking at equation 5.2.9, our time derivative is evaluated at $t+\frac{\Delta t}{2}$, so we need to adjust our second order spatial derivatives by taking the averages at t and $t+\Delta t$:

$$\begin{aligned}
\frac{\partial^2 u}{\partial x^2} &= \frac{1}{2}\left(\frac{u(x+\Delta,y,t+\Delta t)-2u(x,y,t+\Delta t)+u(x-\Delta,y,t+\Delta t)}{\Delta^2}\right. \\
&+ \left.\frac{u(x+\Delta,y,t)-2u(x,y,t)+u(x-\Delta,y,t)}{\Delta^2}\right)+O(\Delta^2) \\
\frac{\partial^2 u}{\partial y^2} &= \frac{1}{2}\left(\frac{u(x,y+\Delta,t+\Delta t)-2u(x,y,t+\Delta t)+u(x,y-\Delta,t+\Delta t)}{\Delta^2}\right. \\
&+ \left.\frac{u(x,y+\Delta,t)-2u(x,y,t)+u(x,y-\Delta,t)}{\Delta^2}\right)+O(\Delta^2)
\end{aligned}$$

We can see that the finite difference approximations for the spatial derivatives, still retain a second order error in the spatial step size.

We can now write equation 5.2.7 using grid notation. Let us define our grid in two spatial dimensions, with the timecourse as the third dimension. Let $n=\frac{t}{\Delta t}$, $j=\frac{x}{\Delta}$, $l=\frac{y}{\Delta}$. Taking the limit of $\tau \to \Delta t$, we can approximate equation 5.2.7 with a partial difference equation

$$\begin{aligned}
\frac{u_{l,j}^{n+1}-u_{l,j}^{n}}{\Delta t} &= \frac{\alpha \Delta t^{\gamma-1}}{2\Delta^2}\sum_{m=0}^{n}\psi(\gamma,m)\left(\delta x_{j,l}^{n-m+1}+\delta x_{j,l}^{n-m}+\delta x_{j,l}^{n-m+1}+\delta y_{j,l}^{n-m}\right) \\
&+O\left(\Delta^2\right)+O\left(\Delta t\right) \qquad (5.2.12)
\end{aligned}$$

where $\delta x_{j,l}^{n}=u_{l+1,j}^{n}-2u_{l,j}^{n}+u_{l-1,j}^{n}$, $\delta y_{j,l}^{n}=u_{l,j+1}^{n}-2u_{l,j}^{n}+u_{l,j-1}^{n}$.

At this point we can add in the generic source/sink term via superposition. For example, if our source or sink term is something like an exponential decay term in time described by $\frac{du}{dt}=-\beta u$, we can incorporate this into equation 5.2.4:$\frac{du}{dt}=\alpha D_t^{1-\gamma}\nabla^2 u-\beta u$. The full equation can be easily discretized in a similar manner as 5.2.12. We choose not to consider the complicacies of a generic source/sink term for the rest of the analysis in

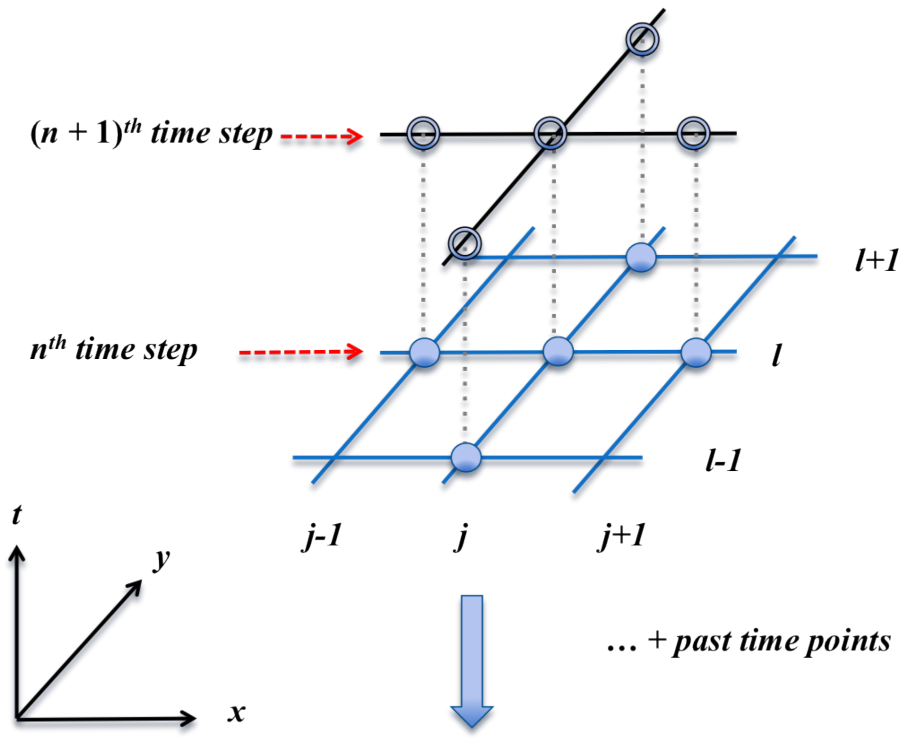


Figure 5.2.1: The schematic shows the classic five-point stencil used in many two-dimensional problems. The Crank-Nicholson algorithm applied to the fractional diffusion equation results in solving for multiple grid points at the $n+1$ timestep, simultaneously, based on current and past values; this requires a system of equations to be solved.

this chapter.

Now, when we step through a time evolution simulation of equation 5.2.12, at each timestep we are interested in solving for all u values for the $n+1$ timestep $\left(u_{j,l}^{n+1}, u_{j-1,l}^{n+1}, u_{j+1,l}^{n+1}, u_{j,l-1}^{n+1}, u_{j,l+1}^{n+1}\right)$, based on past values. See Fig. 5.2.1 for a schematic of this setup.

As shown in Fig. 5.2.1, in addition to the complicacies that the summation introduces, an implicit method requires a system of equations be solved simultaneously. Rearranging equation 5.2.12 yields

$$\begin{aligned}
-ru_{l+1,j}^{n+1} - ru_{l-1,j}^{n+1} + (1+4r)u_{l,j}^{n+1} - ru_{l,j+1}^{n+1} - ru_{l,j-1}^{n+1} \quad = \quad & r\sum_{m=1}^{n} \psi(\gamma,m)\left(\delta x_{j,l}^{n-m+1} + \delta x_{j,l}^{n-m}\right. \\
& \left. + \delta y_{j,l}^{n-m+1} + \delta y_{j,l}^{n-m}\right) \\
& + r\left(\delta x_{j,l}^{n} + \delta y_{j,l}^{n}\right) + u_{l,j}^{n} \\
& + \Delta t\left[O\left(\Delta^2\right) + O\left(\Delta t\right)\right] \qquad (5.2.13)
\end{aligned}$$

where we set $r = \frac{\alpha\Delta t^{\gamma}}{2\Delta^2}$ and make use of the fact that $\psi(\gamma,0) = 1$. Note that the right hand side of equation 5.2.13 is a function of past and present time steps. We can write equation 5.2.13 in matrix form $MU^{n+1} = F$, where F is a matrix that is a function of current and past events. U^{n+1} holds the values that we are solving for, and contains all nodes in our grid u, rearranged in a single column format. M is a large, sparse matrix structured as follows:

$$M = \begin{bmatrix} T & -rI & 0 & \cdots & 0 \\ -rI & T & -rI & 0 & \vdots \\ 0 & -rI & T & -rI & 0 \\ \vdots & 0 & -rI & T & -rI \\ 0 & \cdots & 0 & -rI & T \end{bmatrix},$$

$$T = \begin{bmatrix} (1+4r) & -r & 0 & \cdots & 0 \\ -r & (1+4r) & -r & 0 & \vdots \\ 0 & -r & \ddots & \ddots & 0 \\ \vdots & 0 & \ddots & \ddots & -r \\ 0 & \cdots & 0 & -r & (1+4r) \end{bmatrix}$$

Matrix M is built of block matrices T and $-rI$. T is a tridiagonal matrix, I is the identity matrix, and $-rI$ is a diagonal matrix, and together they make M a very sparse, banded matrix. If our grid u has N_xN_y total nodes, matrix U^{n+1} is of size N_xN_y by 1, and matrix M is of size (N_xN_y) by (N_xN_y). While directly solving this matrix problem will give us our solution, it is extremely computationally inefficient because of the size of matrix M and the fact that it is very sparse. While there are methods to solve sparse matrices efficiently, we have chosen to combine the Crank-Nicholson algorithm with the alternate direction implicit (ADI) method, which splits the problem into two half steps. To ensure that our method is consistent with Eq. 5.2.13, we will first write our formulation in factored form, from which we easily apply time-splitting.

5.2.2 Time-splitting into Two One-dimensional Steps

We write Eq. 5.2.13 using Λ_x and Λ_y as central difference operators in the x and y directions, respectively.

$$\begin{aligned}
\Lambda_x u_{j,l} &= u_{j-1,l} - 2u_{j,l} + u_{j+1,l} \\
\Lambda_y u_{j,l} &= u_{j,l-1} - 2u_{j,l} - u_{j,l+1}
\end{aligned}$$

$$\begin{aligned}
(I - r\Lambda_x - r\Lambda_y)\, u_{l,j}^{n+1} &= (I + r\Lambda_x + r\Lambda_y)\, u_{l,j}^{n} \\
&\quad + r\sum_{m=1}^{n} \psi(\gamma, m)\left[\delta x_{j,l}^{n-m+1} + \delta x_{j,l}^{n-m} + \delta y_{j,l}^{n-m+1} + \delta y_{j,l}^{n-m}\right] \\
&\quad + \Delta t\left[O\left(\Delta^2\right) + O\left(\Delta t\right)\right] \Rightarrow \\
(I - r\Lambda_x)(I - r\Lambda_y)\, u_{l,j}^{n+1} &= (I + r\Lambda_x)(I + r\Lambda_y)\, u_{l,j}^{n} \\
&\quad + r\sum_{m=1}^{n} \psi(\gamma, m)\left[\delta x_{j,l}^{n-m+1} + \delta x_{j,l}^{n-m} + \delta y_{j,l}^{n-m+1} + \delta y_{j,l}^{n-m}\right] \\
&\quad + \frac{\alpha^2 \Delta t^{2\gamma}}{4\Delta^2}\Lambda_x\Lambda_y\left(u_{l,j}^{n+1} - u_{l,j}^{n}\right) \\
&\quad + \Delta t\left[O\left(\Delta^2\right) + O\left(\Delta t\right)\right] \qquad (5.2.14)
\end{aligned}$$

A Taylor expansion of the $\left(u^{k+1}-u^{k}\right)$ term in Eq. 5.2.14 results in an order of accuracy $O(\Delta t)$. Therefore, as long as the cross terms have an order of accuracy greater than or equal to

$O\left(\Delta t^{2}\right)$, we can neglect them without loss of order of accuracy in our scheme. This is true when $2\gamma+1 \geq 2$ or when $\gamma \geq 1/2$. We can still simulate fractional diffusion with an anomalous exponent $\gamma < 1/2$, while accepting the decrease in the order of accuracy of our formulation.

Our formulation in factored form is now

$$\begin{aligned}
\left(I-r\Lambda_x\right)\left(I-r\Lambda_y\right)u_{l,j}^{n+1} = {} & \left(I+r\Lambda_x\right)\left(I+r\Lambda_y\right)u_{l,j}^{n} \\
& +r\sum_{m=1}^{n}\psi(\gamma,m)\left[\delta x_{j,l}^{n-m+1}+\delta x_{j,l}^{n-m}\right. \\
& \left.+\,\delta y_{j,l}^{n-m+1}+\delta y_{j,l}^{n-m}\right] \qquad (5.2.15)
\end{aligned}$$

From Eq. 5.2.15 we can easily make use of time-splitting. At each timestep in our simulation, we split our problem into two half-steps. In each half step, we apply the Crank-Nicholson algorithm only in one of our two spatial directions. This simplification has the effect of reducing our two-dimensional problem into two one-dimensional problems, thus eliminating the need for dealing with large and sparse matrices that are usually associated with two-dimensional problems. See Fig. 5.2.2 for a basic schematic of the time-splitting algorithm described in this section.

We also note that when $\gamma=1$, $\psi(\gamma,0)=1$ and $\psi(\gamma,m)=0$ for all $m\neq 0$. This reduces our numerical scheme to

$$\left(I-r\Lambda_x\right)\left(I-r\Lambda_y\right)u_{l,j}^{n+1}=\left(I+r\Lambda_x\right)\left(I+r\Lambda_y\right)u_{l,j}^{n}$$

which is the approximate factorization of the classical diffusion equation using the Crank-

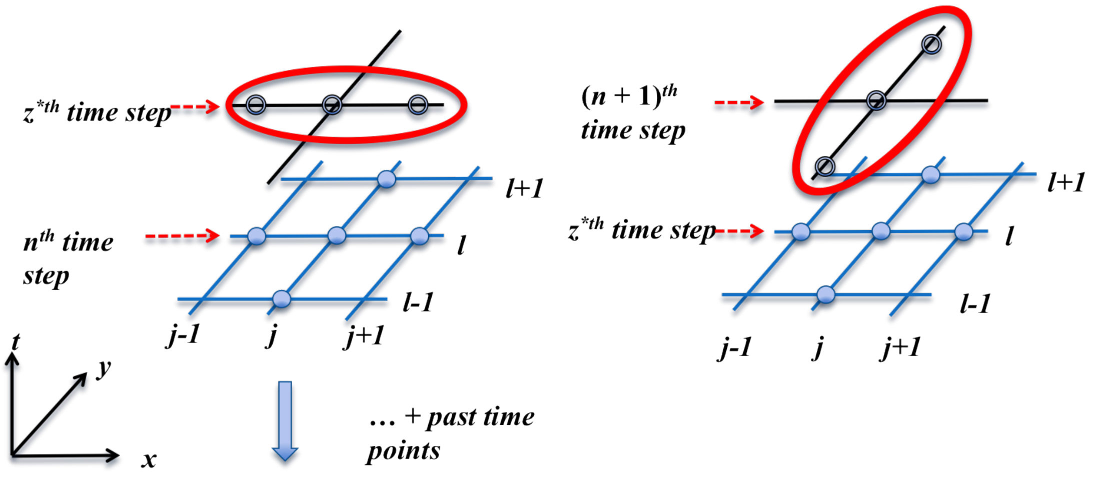


Figure 5.2.2: The schematic shows how the original Crank-Nicholson setup based on the five-point stencil, is split into two one-dimensional problems where at each step, we are solving a system of equations in one spatial direction only.

Nicholson method, which is well established to be second order accurate in time and space [Parvez].

Step 1: We define $(I - r\Lambda_y)\, u_{j,l}^{n+1} = z$, so equation 5.2.15 now reads

$$\begin{aligned}(I - r\Lambda_x)\, z_{l,j} &= (I + r\Lambda_x)(I + r\Lambda_y)\, u_{l,j}^{n} \\ &\quad + r\sum_{m=1}^{n} \psi(\gamma, m)\left[\delta x_{j,l}^{n-m+1} + \delta x_{j,l}^{n-m} + \delta y_{j,l}^{n-m+1} + \delta y_{j,l}^{n-m}\right] \quad (5.2.16)\end{aligned}$$

where z can be thought of as a 'virtual' timestep which we solve for as an intermediate value but don't use as part of our final solution (see Fig. 5.2.3). We can write Eq. 5.2.16 in matrix form if we carefully define our grid of interest on which we will do our simulation. If we are interested in solving the problem with Dirichlet boundary conditions. Let us

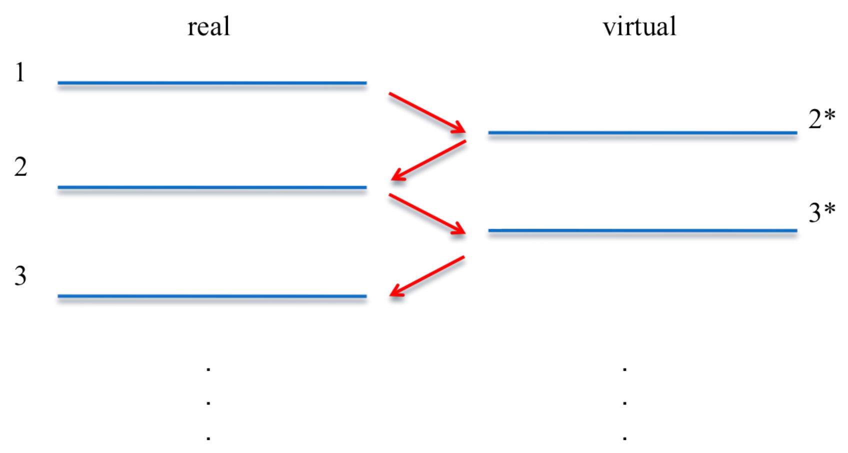


Figure 5.2.3: The values at the current and past real time steps are used to calculate the values at the next virtual timestep. This is in turn used to calculate the values at the next real time step.

identify our grid as the following:

$$N_y\text{nodes}\left\{\underbrace{\begin{array}{ccccc} B_{1,1} & \cdots & B_{j,1} & \cdots & B_{Nx,1} \\ \vdots & u_{2,2} & \cdots & u_{N_x-1,2} & \vdots \\ B_{1,l} & \vdots & \ddots & \vdots & B_{Nx,l} \\ \vdots & u_{2,N_y-1} & \cdots & u_{N_x-1,N_y-1} & \vdots \\ B_{1,Ny} & \cdots & B_{l,Ny} & \cdots & B_{Nx,Ny} \end{array}}_{N_x\text{nodes}}\right. \tag{5.2.17}$$

where the size of the grid is N_y total gridpoints in the y direction, and N_x total gridpoints in the x direction. If B is a function of time, it must be known at all times. Then we can prescribe boundary conditions for intermediate variable z, from its definition. We can also easily modify our matrices to incorporate other types of boundary conditions, including Neumann and mixed conditions. For the purposes of analysis later in this paper, we will consider a simple case with periodic Dirichlet boundary conditions that can be a function of time.

With this formulation, we are interested in finding the z values for the internal (non-boundary) gridpoints of layout 5.2.17. This problem can be represented in matrix form $AZ = R$:

$$\overbrace{\begin{bmatrix} 1+2r & -r & 0 & \cdots & 0 \\ -r & 1+2r & \ddots & & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & 1+2r & -r \\ 0 & \cdots & 0 & -r & 1+2r \end{bmatrix}}^{A} \overbrace{\begin{bmatrix} z_{1,1} & \cdots & u_{N_x-2,1} \\ & & \\ \vdots & & \vdots \\ & & \\ z_{1,N_y-2} & \cdots & z_{N_x-2,N_y-2} \end{bmatrix}}^{Z} = R \quad (5.2.18)$$

where R represents the right hand side of equation 5.2.16.

We see that because we are solving an implicit algorithm in only one direction, matrix A is a tridiagonal matrix, and this matrix equation can be very efficiently solved using the Thomas algorithm, which uses a simplified Gaussian elimination.

Step 2: Now that we have solved for the values of z, we can solve the entire equation in 5.2.15

$$(I - r\Lambda_y)\, u_{j,l}^{n+1} = z \quad (5.2.19)$$

As before, equation 5.2.19 can be formulated as a matrix equation

$$\overbrace{\begin{bmatrix} 1+2r & -r & 0 & \cdots & 0 \\ -r & 1+2r & \ddots & & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & 1+2r & -r \\ 0 & \cdots & 0 & -r & 1+2r \end{bmatrix}}^{A'} \overbrace{\begin{bmatrix} u_{1,1}^{k+1} & \cdots & u_{N_x-2,1}^{k+1} \\ & & \\ \vdots & & \vdots \\ & & \\ u_{1,N_y-2}^{k+1} & \cdots & u_{N_x-2,N_y-2}^{k+1} \end{bmatrix}^{T}}^{U'} = Z \quad (5.2.20)$$

In comparing equations 5.2.18 and 5.2.20, we may note that the U matrices are transposed: this is due to the fact that in Step 1, we applied the Crank-Nicholson scheme to one spatial dimension, and in Step 2, we applied it to the other spatial direction.

In Step 2 we are solving for the $n+1$ timestep efficiently using another tridiagonal matrix. However, in this step we don't need to consider memory contributions of past timesteps as that was accounted for in Step 1. This saves us from having to keep track of the history of virtual time steps, as the real timesteps are the ones we are interested in and visualize as our solution.

5.2.3 Adaptive Timestep

The ADI time-splitting method applied to the Crank-Nicholson differencing scheme described in the previous sections, increases the accuracy and stability, and decreases the computational time of our numerical scheme, compared to the full implementation described in Chapter 2. In addition, we can now apply the adaptive timestep algorithm to further save on computational time, while still maintaining a reasonable order of accuracy. Let a be the base interval used in the algorithm (adaptive time step parameter).

For timestep $n \leq a$, the formulation is exactly the same as in equations 5.2.16 and 5.2.19.

For $n > a$

$$
\begin{aligned}
(I - r\Lambda_x) z_{j,l} &= (I + r\Lambda_x)(I + r\Lambda_y) u_{l,j}^{n} \qquad (5.2.21) \\
&+ r \left[\sum_{m=1}^{a} \psi(\gamma, m) \left(\delta x_{j,l}^{n-m+1} + \delta x_{j,l}^{n-m} + \delta y_{j,l}^{n-m+1} + \delta y_{j,l}^{n-m} \right) \right. \\
&+ \sum_{s=2}^{s_{max}} \left\{ \sum_{\eta=1}^{\eta_{max}(s,n)} (2s-1)\, \psi(\gamma, M(s,\eta)) \left(\delta x_{j,l}^{n-M(s,\eta)+1} \right. \right. \\
&\left. + \delta x_{j,l}^{n-M(s,\eta)} + \delta y_{j,l}^{n-M(s,\eta)+1} + \delta y_{j,l}^{n-M(s,\eta)} \right) \\
&\left. \left. + \sum_{p=M_{max}+s_{max}}^{min(a^s,n)} \psi(\gamma, p) \left(\delta x_{j,l}^{n-p+1} + \delta x_{j,l}^{n-p} + \delta y_{j,l}^{n-p+1} + \delta y_{j,l}^{n-p} \right) \right\} \right] \\
(I - r\Lambda_y) u_{l,j}^{n+1} &= z_{j,l} \qquad (5.2.22) \\
M(s,\eta) &= a^{s-1} + (2s-1)\eta - s + 1 \\
\eta_{max}(s,n) &= min \left(\left\lfloor \frac{a^s - a^{s-1}}{2s-1} \right\rfloor, \left\lfloor \frac{n - a^{s-1}}{2s-1} \right\rfloor \right) \\
M_{max} &= M(s_{max}, \eta_{max}) \\
\delta x_{j,l}^{n} &= u_{j+1,l}^{n} - 2u_{j,l}^{n} + u_{j-1,l}^{n}
\end{aligned}
$$

where $r = \frac{\alpha \Delta t^{\gamma}}{\Delta^2}$, $\lfloor \, \rfloor$ denotes the floor function, and s_{max} is determined by the current timestep n such that $a^{s_{max}-1} + 1 \leq n \leq a^{s_{max}}$.

5.3 Analysis

5.3.1 Consistency

We begin with a consistency analysis of Eq. 5.2.15. Consider the original fractional diffusion equation using the Grunwald-Letnikov definition given in Eq. 5.2.5. Let $\tilde{u}$ be the exact solution of that differential equation and u be the approximate solution that is satisfied by our discrete numerical method given in Eq. 5.2.15. Let $L[u]$ be the difference operator

$$\begin{aligned} L\left[u_{l,j}^{n}\right] &= (I - r\Lambda_x)(I - r\Lambda_y)\, u_{l,j}^{n+1} - (I + r\Lambda_x)(I + r\Lambda_y)\, u_{l,j}^{n} \\ &\quad - r\sum_{m=1}^{n} \psi(\gamma, m)\left[\delta_x^{n-m+1} + \delta_x^{n-m} + \delta_y^{n-m+1} + \delta_y^{n-m}\right] \end{aligned} \tag{5.3.1}$$

For terms outside the summation, we apply a Taylor Series expansion around location x_j, y_l and the current timepoint n, but within the summation, we expand around the local timepoint in the past history, $n - m$. After simplifying, our end result is

$$L\left[u_{j,l}^{n}\right] = \Delta_t \frac{\partial u_{j,l}^{n}}{\partial t} + \frac{\Delta_t^2}{2}\frac{\partial^2 u_{j,l}^{n}}{\partial t^2} - \alpha\Delta_t^{\gamma}\sum_{m=0}^{n}\psi(\gamma, m)\nabla^2 u_{j,l}^{n-m} + \ldots\text{(higher order terms)} \tag{5.3.2}$$

If we use the notation $u_{j,l}^{n} = u\left(x_j, y_l, t_n\right)$ with $x_j = j\Delta$, $y_l = l\Delta$, $t_n = \Delta_t n$, and consider that indices j, l, n are generic, equation 5.3.2 can apply to any point in space and time. Therefore we can write

$$L[u] = \Delta_t \frac{\partial u(x, y, t)}{\partial t} + \frac{\Delta_t^2}{2}\frac{\partial^2 u(x, y, t)}{\partial t^2} - \alpha\Delta_t^{\gamma}\sum_{m=0}^{n}\psi(\gamma, m)\nabla^2 u(x, y, t - m\Delta_t)$$

Since u is the solution of our finite difference equation 5.2.15, $L[u] = 0$ and we can divide through by Δ_t to get

$$\frac{\partial u(x, y, t)}{\partial t} = -\frac{\Delta_t}{2}\frac{\partial^2 u(x, y, t)}{\partial t^2} + \alpha\Delta_t^{\gamma-1}\sum_{m=0}^{n}\psi(\gamma, m)\nabla^2 u(x, y, t - m\Delta_t)$$

If we take the limit as $\Delta, \Delta_t \to 0$ and replace Δ_t with τ we get

$$\frac{\partial u(x, y, t)}{\partial t} = \lim_{\tau\to 0} \tau^{\gamma-1}\alpha\sum_{m=0}^{t/\tau}\psi(\gamma, m)\nabla^2 u(x, y, t - m\tau)$$

Therefore, the difference equation in 5.2.15 is consistent with the original equation we began with (Eq. 5.2.7).

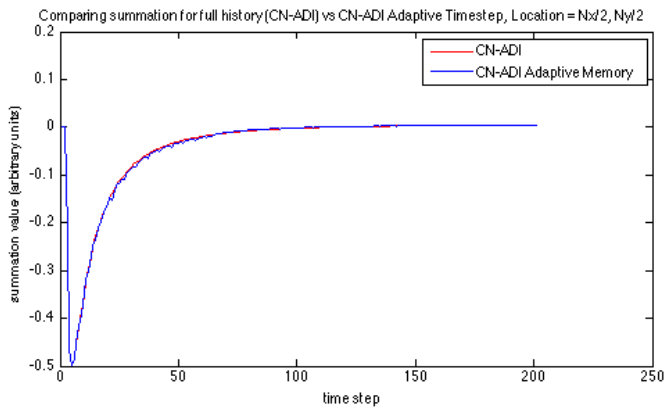
In the same manner we apply a consistency analysis to the adaptive memory version of our scheme, using equations 5.2.21 and 5.2.22. As expected, the final equation that we recover is not the original homogenous equation that we begin with, but the following, adaptive memory version:

$$\begin{aligned}\frac{\partial u(x,y,t)}{\partial t} &= \lim_{\tau\to 0}\tau^{\gamma-1}\alpha\left\{\sum_{m=0}^{t_a/\tau}\psi(\gamma,m)\nabla^2 u(x,y,t_a-m\tau)\right.\\ &+\sum_{s=2}^{s_{max}}\left\{\sum_{\eta=1}^{\eta_{max}(s,n)}(2s-1)\,\psi(\gamma,M(s,\eta))\,\nabla^2 u(x,y,t_\eta-M(s,\eta)\,\tau)\right\}\\ &+\left.\sum_{p=M_{max}+s_{max}}^{t/\tau}\psi(\gamma,p)\nabla^2 u(x,y,t-p\tau)\right\} \qquad (5.3.3)\end{aligned}$$

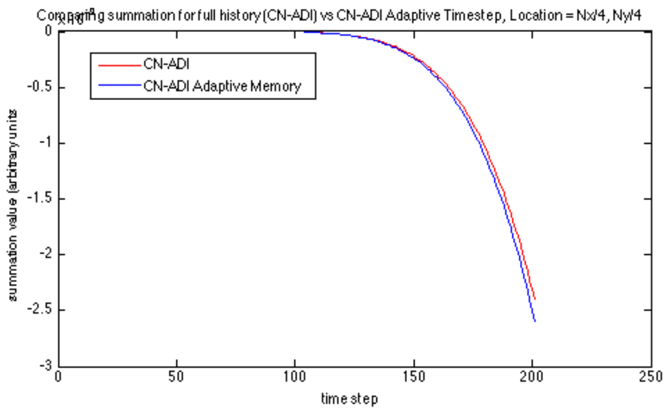
where t_a, t_η are the exact times when timesteps a and η occur, respectively. We consider Eq. 5.3.3 to be a good approximation of Eq. 5.2.7 based on the fact that the total contribution of past timepoints as described by all the discrete summations in the adaptive memory algorithm (Section 5.2.3) very closely approaches the value of the summation in Eq. 5.2.15. Figure 5.3.1 compares the value of the summations in equations 5.2.21 and 5.2.15 at three different locations on our spatial grid. We find that the way the adaptive memory algorithm describes a weighted sampling of the history of the simulation is a very good approximation of the full history when sampled at every past timestep.

5.3.2 Stability

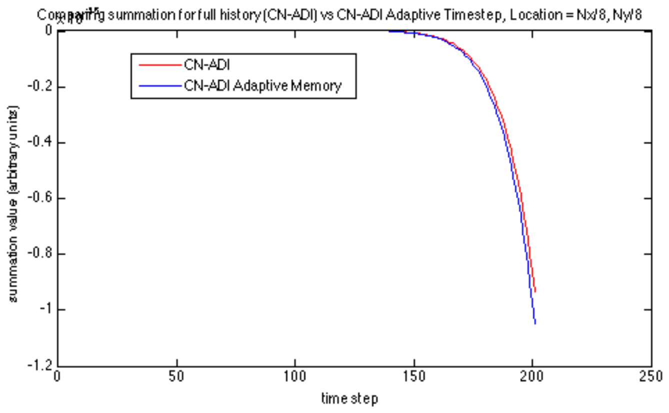
Because our adaptive step algorithm is a linear scheme with constant coefficients on a uniformly spaced grid, and for the scope of this analysis we are assuming simple periodic Dirichlet boundary conditions, we are able to use a von Neumann type stability analysis, as we have done in Chapter 4. For more complex boundary conditions or



A)



B)



C)

Figure 5.3.1: The value of the history summation values in Eq. 5.2.15 and Eq. 5.2.21 is compared at **A)** $u_{Nx/2,Ny/2}$, **B)** $u_{Nx/4,Ny/4}$, and **C)** $u_{Nx/8,Ny/8}$

non-constant parameters and coefficients, we will need more advanced approaches to stability including matrix stability analysis. Using separation of variables we assume a special product form of the solution to determine the convergence or divergence of spatial wave oscillations:

$$u_{j,l}^{n} = \sum_{v=0}^{N_y-1} \sum_{u=0}^{N_x-1} A_{uv}^{n} e^{ik_u x_j} e^{ik_v y_l} \tag{5.3.4}$$

Where A_{uv}^{n} can be thought of as the amplification factor dependent on time, and k_u and k_v are the wavenumbers in the x and y directions, respectively. See [Haberman, Chapter 6, Section 3] for a thorough derivation.

Substituting Eq. 5.3.4 into equations 5.2.21 and 5.2.22 and simplifying in the same manner as in Chapter 4, we get

$$\begin{aligned} A_{uv}^{n+1}(1-rB)(1-rD) = {} & A_{uv}^{n}(1+rB)(1+rD) + r\Bigg[\sum_{m=1}^{a} \psi(\gamma,m)\left(BA_{uv}^{n-m+1} + BA_{uv}^{n-m}\right. \\ & \left. + DA_{uv}^{n-m+1} + DA_{uv}^{n-m}\right) \\ & + \sum_{s=2}^{s_{max}} \left\{ \sum_{\eta=1}^{\eta_{max}(s,n)} (2s-1)\,\psi(\gamma, M(s,\eta))\left(BA_{uv}^{n-M+1}\right.\right. \\ & \left. + BA_{uv}^{n-M} + DA_{uv}^{n-M+1} + DA_{uv}^{n-M}\right) \\ & + \sum_{p=M_{max}+s_{max}}^{min(a^s,n)} \psi(\gamma,p)\left(BA_{uv}^{n-p+1} + BA_{uv}^{n-p}\right. \\ & \left.\left. + \; DA_{uv}^{n-p+1} + DA_{uv}^{n-p}\right)\right\}\Bigg] \end{aligned} \tag{5.3.5}$$

where $B = -4sin^2\left(\frac{k_u \Delta x}{2}\right)$ and $D = -4sin^2\left(\frac{k_v \Delta y}{2}\right)$. We would like to show the amplification factor $A_{uv}^{n}$is bounded in time; that is,

$$|A_{uv}^{n}| \leq C\left|A_{uv}^{0}\right| \text{ for all timesteps } n$$

where C is some constant that is not dependent on any parameters like time or spatial discretization steps. Solving for A^n_{uv} is very complicated, so we will instead proceed with a proof by induction. Our base case is for $n = 0$:

$$\begin{aligned}
A^1_{uv}(1-rB)(1-rD) &= \phi_0(1+rB)(1+rD) \Rightarrow \\
\left|\frac{A^1_{uv}}{A^0_{uv}}\right| &= \left|\frac{(1+rB)(1+rD)}{(1-rB)(1-rD)}\right| \qquad (5.3.6)
\end{aligned}$$

Since $r > 0$ and $B, D \le 0$, the numerator in Eq. 5.3.6 is always less than or equal to the denominator, and so we have $\left|A^1_{uv}\right| \le \left|A^0_{uv}\right|$

For $n = 1$ we have

$$\begin{aligned}
A^2_{uv}(1-rB)(1-rD) &= A^1_{uv}(1+rB)(1+rD) + r\left(BA^1_{uv} + BA^0_{uv} + DA^1_{uv} + DA^0_{uv}\right) \\
&= A^1_{uv}\left[(1+rB)(1+rD) + rB + rD\right] + A^0_{uv}(rB+rD) \\
&\Rightarrow \\
\left|A^2_{uv}\right| &= \frac{1}{(1-rB)(1-rD)}\left|A^1_{uv}\{(1+rB)(1+rD) + rB + rD\}\right. \\
&\quad \left. + A^0_{uv}(rB+rD)\right| \\
&\le \frac{1}{(1-rB)(1-rD)}\left\{\left|A^1_{uv}\right|\left|(1+rB)(1+rD) + rB + rD\right|\right. \\
&\quad \left. + \left|A^0_{uv}\right|\left|rB+rD\right|\right\} \\
&\le \frac{\left|A^0_{uv}\right|}{(1-rB)(1-rD)}\left\{\left|(1+rB)(1+rD)\right| + \left|rB+rD\right|\right. \\
&\quad \left. + \left|rB+rD\right|\right\} \\
&= \frac{\left|A^0_{uv}\right|}{(1-rB)(1-rD)}\left\{\left|1+rB+rD+r^2BD\right| - 2rB - 2rD\right\} \\
&\le \left|A^0_{uv}\right|\left(\frac{1+r^2BD-3r(B+D)}{1+r^2BD-r(B+D)}\right) \\
&= \left|A^0_{uv}\right| f(r,B,D) \qquad (5.3.7)
\end{aligned}$$

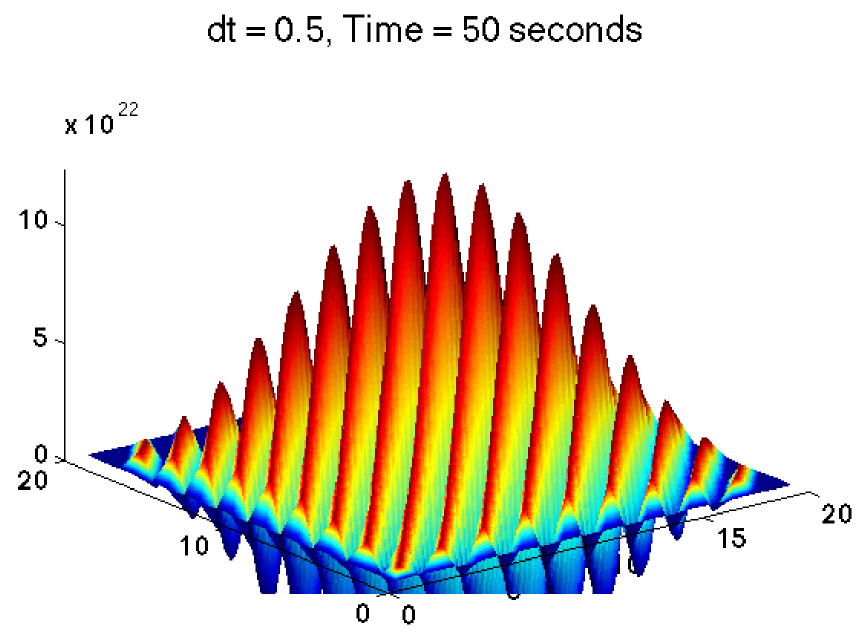
The value of function f is dependent on B and D, and so indirectly depend on the wavenumbers k_u and k_v. The exact expression of $f(r,B,D)$ will determine the conditions under which this method is stable, and we leave this as an open problem to be solved. However, it can be empirically shown that the Crank-Nicholson ADI method with adaptive timestep implementation, does have an expanded stability regime compared to the explicit version of the adaptive timestep implementation defined in Chapter 3. For example, in Fig. 5.3.2 we see a simulation with a set of parameters that results in an unstable solution with the full implementation, and a stable one with the CN-ADI Adaptive Memory algorithm, demonstrating an expanded stability regime for the latter numerical method.

5.3.3 Order of Accuracy

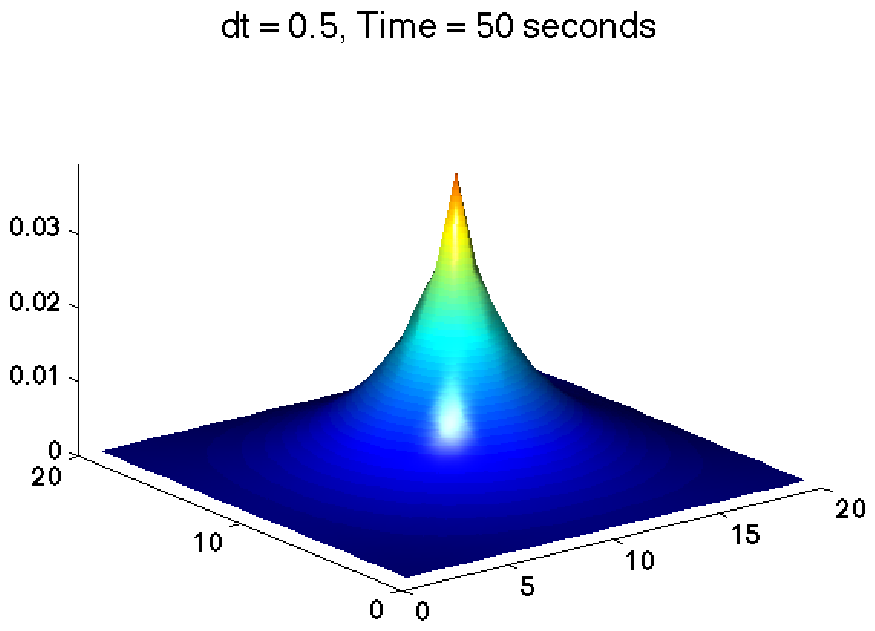
As shown in Section 5.2, we are $O\left(\Delta^2\right)$ in the spatial variable, but only $O\left(\Delta_t\right)$ in the time variable, due to the Grünwald-Letnikov approximation and the way in which we have defined our binomial coefficients. As discussed in [40], the memory function $\psi(\gamma, m) = (-1)^m \begin{pmatrix} 1-\gamma \\ m \end{pmatrix}$ can be considered the coefficients of the power series for the function $(1-z)^{1-\gamma}$:

$$(1-z)^{1-\gamma} = \sum_{m=0}^{\infty} (-1)^m \begin{pmatrix} 1-\gamma \\ m \end{pmatrix} z^m$$

Using this definition with the Grünwald-Letnikov approximation gives us an expression for the $(1-\gamma)$ th order fractional derivative that is $O(\Delta_t)$. Lubich shows that we can obtain higher order approximations by using memory functions that are the coefficients of the Taylor series expansions of the appropriate ‘generating functions’ [26, 40]. For a second order approximation, we use the generating function $\chi(\gamma, z) = \left(\frac{3}{2} - 2z + \frac{z^2}{2}\right)^{1-\gamma}$. In this way we obtain a numerical scheme that recovers $O\left(\Delta t^2\right)$, an advantage of the Crank Nicholson formulation that was lost.



A)



B)

Figure 5.3.2: Stability comparison for the full implementation and the CN-ADI Adaptive Memory method. The set of parameters used are $\alpha = 50 units^2/s^\gamma$, $\Delta t = 0.5$ s,$\Delta = 10$ *units*, $\gamma = 0.7$. A) These parameters clearly result in an instable solution with the full implementation. B) However, with the same set of parameters we remain in the stable regime for the CN-ADI Adaptive Memory method. This clearly demonstrates an expanded stability regime for the latter method.

A disadvantage to recalculating the memory function, however, is that we can no longer use an ADI algorithm. With a recalculated memory function, in Eq. 5.2.14, we would only be able to ignore the cross terms if $\gamma > 1$, which precludes all equations in the sub diffusive realm. We could still use the unfactored version of the Crank Nicholson scheme in Eq. 5.2.13 and combine it with the adaptive memory algorithm, but we will have lost some of the advantage that factoring and time-splitting provides in terms of efficiency.

However, another potential approach to recovering a $O\left(\Delta t^2\right)$ scheme, is to use Richardson extrapolation in the time variable. Richardson extrapolation obtains accurate numerical schemes by combining two or more less accurate solutions [37]. For example, with a set timestep Δt, we derived the exptression 5.2.6 that is $O(\Delta_t)$. We can now set time step to $\Delta t/2$ and rederive an expression that is also $O(\Delta_t)$ but with different coefficients. We can then create a linear combination of the two expressions to eliminate the $O(\Delta_t)$ term, leaving us with higher order truncated terms and a scheme that is of higher order accuracy in time. We leave this as an open problem to be considered in the future.

5.4 Results

5.4.1 Accuracy

We compare the results of our CN-ADI Adaptive Memory algorithm given in Section 5.2.3, with the Full Implementation, described in Chapter 3. We simulated an initial value problem with the following conditions: $\Delta = 10\mu m$, $N_x = N_y = 20$, $\Delta t = .5s$, $\alpha = 20\mu m^2/s^\gamma$, $\gamma = 0.7$, with an initial condition of $u(\vec{x},0) = e^{-x^2/2\sigma_1^2}e^{-y^2/2\sigma_2^2}$, $\sigma_1 = 5\,units$, $\sigma_2 = 5\,units$. Fig. 5.4.1 shows the percentage error of the CN-ADI Adaptive Memory implementation at grid point $u^n_{N_x/2,N_y/2}$, compared to the Full Implementation, for the duration of the simulation. For initial timesteps, there is a large error compared to

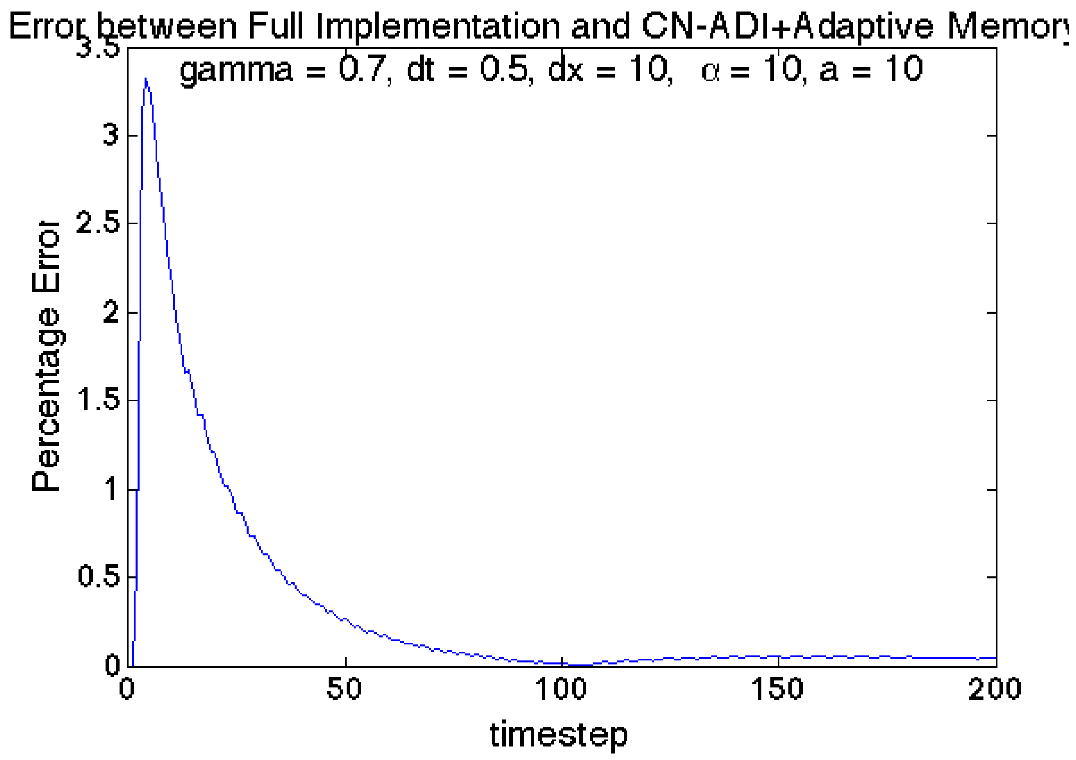


Figure 5.4.1: The figure shows the percentage error of the solution using the CN-ADI Adaptive Memory algorithm, compared to the full finite difference algorithm, at the center of the 2D grid, which is the most quickly changing part of the simulation.

the full algorithm, but this can be improved by reducing Δt. After the initial timesteps, the error quickly converges to less than 0.5%. We note here that the full implementation itself is a numerical algorithm that has an error compared to the true solution; however, lacking an exact analytical solution for this problem, we consider it acceptable to compare our algorithm to the full implementation, based on the assumption that the full implementation error decays or is bounded in time. Therefore in comparing our algorithm to the full implementation, a converging error would mean that the error associated with our new method, is also bounded in time.

5.4.2 Computational Time

In Fig. 5.5.1 we compare the actual computational time required to simulate a set number of seconds, for the full fractional diffusion implementation as given by Podluby([40]) and explored fully in Chapter 3 (Full Implementation), the adaptive memory algorithm based on an explicit time scheme as described in Chapter 3 (Adaptive Memory), the Crank-Nicholson scheme combined with the ADI method given by equation 5.2.15 (CN-ADI), and the Crank-Nicholson ADI method that includes an adaptive memory algorithm, as given in Section 5.2.3 (CN-ADI Adaptive Memory). The plots demonstrate that the Full Implementation algorithm is the least efficient, as computational time has a near exponential dependence on simulation time, due to the costs of taking into account the entire past memory of the simulation at every timestep and having to loop over all grid location in both spatial directions. Both the original Adaptive Memory and CN-ADI algorithms present a significant increase in efficiency compared to the Full Implementation, and the decrease in computational time becomes more significant as simulation length increases. Compared to all three of the other algorithms, the CN-ADI Adaptive Memory is shown to have the greatest decrease in computational time.

However, we also note that in Chapter 4 we have done a complete algorithmic complexity analysis on any scheme that is based on the full sampling of past timesteps or integrates the adaptive memory algorithm based on an arithmetic sequences. Both were found to have complexity $O\left(N^2\right)$ where N is the number of timesteps (proportional to simulation time). Therefore, all four methods referred to in this section and for which data is shown in Fig. 5.4.2, have complexity $O\left(N^2\right)$, despite having drastically different raw execution times for a given simulation.

In Chapter 4 we also analyzed the complexity of the linked list implementation (which essentially applies an adaptive time step algorithm based on a power law) and found it to be $O\left(Nlog_2N\right)$, which scales considerably more efficiently with number of timesteps N. Therefore a natural consideration for future development of this work would be to adapt the Crank-Nicholson -ADI scheme developed in this chapter, with the linked list implementation. The complexity would be $O(Nlog_2N)$ but additional stability and consistency analyses would be necessary.

5.5 Conclusion

With the CN-ADI Adaptive Memory scheme we have developed an implicit, efficient numerical method for solving the two-dimensional time-fractional diffusion equation. With an expanded stability regime it can be applied to problems with biophysically accurate parameters (such as large diffusion coefficients) without requiring an unfeasibly small timestep. With regards to computational time it performs well compared to the full implementation and adaptive memory schemes described in Chapter 3. However, as alluded to in the last section, there is room to further improve on its algorithmic complexity by combining it with the linked list approach.

In addition, we also note that although implicit methods have many advantages over explicit methods (stability, accuracy, efficiency), it is also far less straightforward to incorporate nonlinear or complex source terms, when applying these schemes to real physical problems. In this regards, explicit schemes have an advantage of ease of applicability. Depending on the problem being modeled, the advantages and disadvantages of all options must be considered; the most appropriate algorithm will dependon the context of the problem being solved.

Chapter 5, in part, is currently being prepared for submission for publication of the material. Nirupama Bhattacharya and Gabriel A. Silva. The dissertation author was the primary investigator and author of this paper.

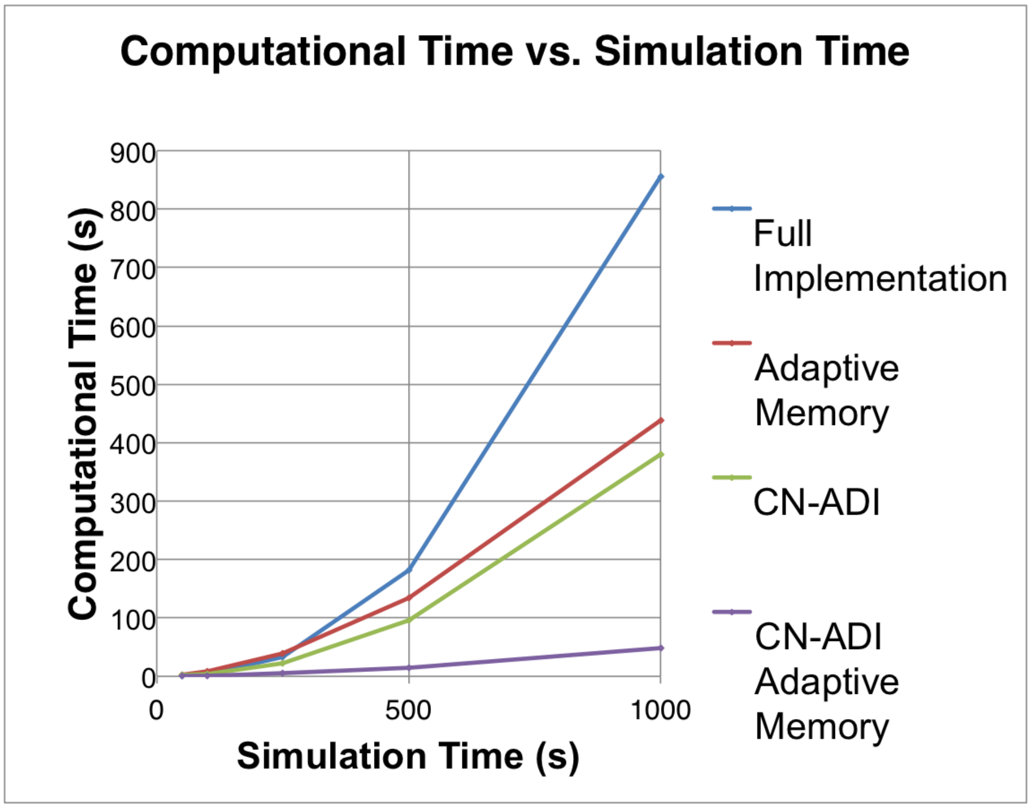


Figure 5.5.1: Comparison of computational times as a function of simulation time, for various numerical algorithms for the fractional diffusion equation.

Chapter 6

Fractional Diffusion of IP_3 in the Induction of LTD

6.1 Introduction

Over the last three chapters we have discussed in depth numerical methods used to solve the time-fractional diffusion equation, and now turn our focus to an application of fractional diffusion in neuroscience, specifically regarding the anomalous diffusion of ubiquitous second messenger signaling molecule inosital-1,4,5-trisphosphate, or IP_3.

It has been experimentally and computationally demonstrated that anomalous diffusion of various molecules occurs along the spiny dendrites of Purkinje neurons, particularly for the second messenger IP_3 [43]. Detailed computational analysis explains that the anomalous nature of the diffusion is caused by the complex geometry of spines along the dendrites, which act as localized traps for particles diffusing along the axial direction of the dendrite. Santamaria et. al. also explored the diffusion of calcium; however they found that because excess calcium concentration was so quickly removed by buffers that it did not have a chance to be trapped, it did not undergo anomalous diffusion. This result is verified by Biess et. al [5], who find that calcium dynamics are most strongly regulated by buffers and not by cytoplasmic crowding or the geometry of the dendritic spines.

In the context of Chapter 2 and the discussion of certain long-tailed distributions giving rise to the fractional diffusion equation, we can interpret the findings in [43] in the following way. If diffusing particles travel along the dendrite as they normally would, they are taking a finite step size after a finite time step. However, if some of these particles are locally trapped within a nearby spine, from the axes of the dendrite, it appears that these particles are not moving, and thus waiting for a long period of time before making the next jump (although they may be diffusing within the spine itself). This behavior corresponds to a long tailed probability distribution for waiting time, which, as we have seen in Chapter 2, is the foundation for the derivation of the time-fractional diffusion equation. Thus, anomalous diffusion in the complex geometry of spiny dendrites, can be accurately reflected by the time-fractional diffusion equation.

Although IP_3 plays an important role in many biological contexts, in this chapter we focus on its role in the signaling events leading up to the induction of long-term depression, or LTD, in the spines of Purkinje dendrites. Long-term depression is an important form of synaptic plasticity that results in sustained depression of the activity level of a neuron, caused by decrease in synaptic strength between the neuron and its inputs. LTD is observed in cerebellar Purkinje neurons, and is involved in learning and memory [9], motor learning tasks and coordination, as well as posture and locomotion adaptation [16].

One of the main mechanisms of LTD induction, is coincidental activation of a neuron by climbing fiber (CF) and parallel fiber (PF) inputs. Parallel fiber inputs are from the axons of neighboring granule cells synapsing onto the spines of a Purkinje cell dendrite. While each spine generally has a weak synaptic connection with a single

PF [17], a single Purkinje neuron may have connections with a total of hundreds of thousands of parallel fibers. The biochemical details of the parallel fiber input involves a cascade of signaling events that begins when glutamate released from the granule cell, binds to its receptors on the spine of the Purkinje neuron. This sets off a sequence of events that results in production of IP_3 that then diffuses through the cytosol of the spine and neighboring dendrite [16], become degraded, or bind to IP_3R (IP3 receptors), the last of which can result in calcium release from internal stores. In contrast to the multiple connections with parallel fibers, a single Purkinje neuron has a total input from a single climbing fiber which has a strong synaptic connection with the Purkinje neuron [17]. CF activation results in a nonlocal depolarization of the neuronal membrane (in contrast to a very localized activation of a single spine area by PF inputs); this allows calcium influx into the cytosol through voltage-gated calcium channels. LTD has been found to be initiated by a supralinear calcium response from coincidental PF and CF activation of the Purkinje neuron; this response is greater in magnitude than the linear summation of the individual calcium responses to individual PF or CF activation [17, 16].

Hernjak et. al. built a one-dimensional model to simulate the signaling events that precedes LTD in a cerebellar Purkinje dendrite. Their model simulates the concentration of several species, including second messengers calcium and IP_3, as well as calcium buffers, which strongly regulate the calcium levels outside the ER. For each species, the concentration in the dendrite and spines are modeled as two separate compartments, with a flux term describing the flow between the compartments, that takes into account the complex geometry of the spine neck (see schematic in Fig. 6.1.1). This model was able to accurately reflect parallel-fiber (PF) and climbing-fiber (CF) stimulation of a Purkinje dendrite, both independently of each other, and coincidentally with each other. The model also captures the supralinear calcium influx that results from coincidental PF and CF stimulation, which, as stated, is a key component in the induction of LTD.

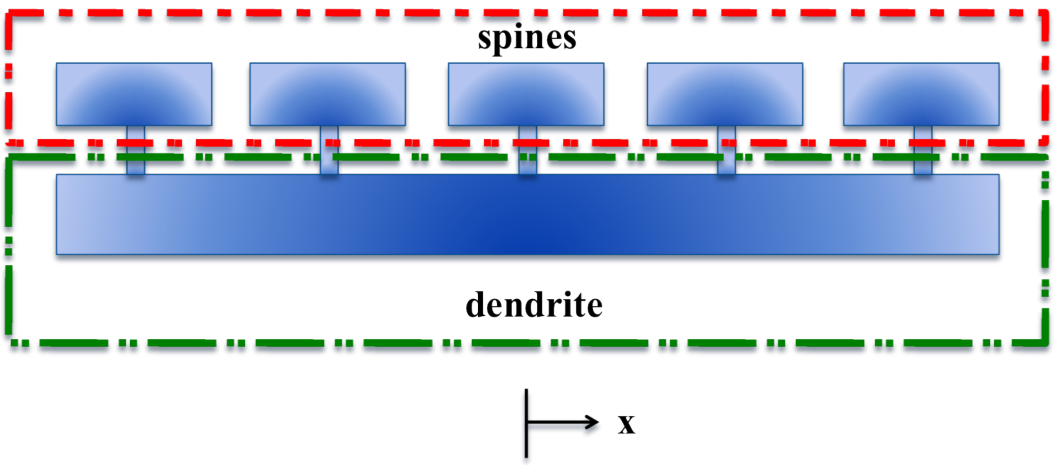


Figure 6.1.1: Schematic of the model developed by Hernjak et. al [16]. The model is simulated on a one-dimensional domain, but takes into account separate compartments for concentrations in the spines, and concentration in the dendrite. A flux term is included to simulate the transport of material between the compartments, through the narrow spines necks.

In this chapter, our goal is to model the diffusion of IP_3 using fractional diffusion, to reflect the geometric complexities of spines along the dendrite. A fractional diffusion component is also used to simplify Hernjak's multi compartmental model, while still retaining the functionality that captures the dynamics of PF and CF activation, and the very important supralinear calcium response that is a vital component in the induction of LTD. Instead of simulating species concentrations for both dendritic and spine compartments, our modified model only has a single one-dimensional domain and does not explicitly calculate flux through the spine neck or take into account the complex geometry of the spines (see schematic in Fig. 6.1.2); those details are reflected in the fractional diffusion components, as have been demonstrated in previous chapters.

6.2 Modified Model

The model we develop here is a simplified version of the one-dimensional model developed in [16], and adapted to include fractional diffusion of second messenger

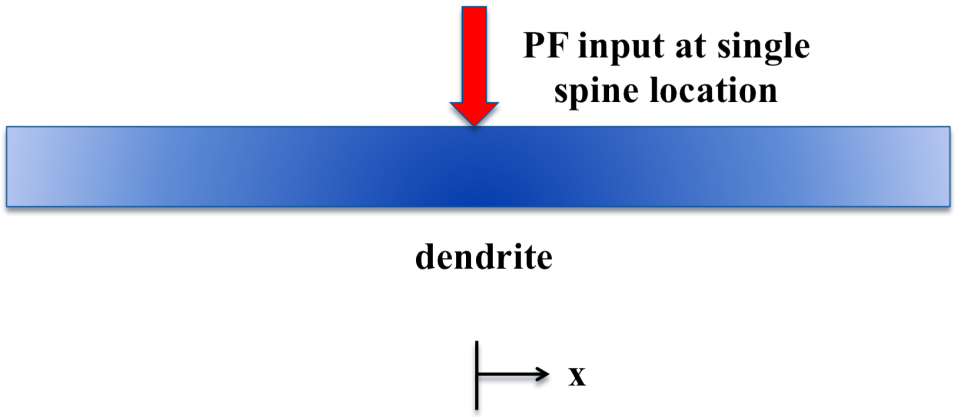


Figure 6.1.2: In the modified model presented in this chapter, we collapse everything into a single compartment that represents the dendrite along a one-dimensional domain. PF input, which occurs at a single spine, is applied to a single location along the one-dimensional axis.

IP_3. We model the concentration of second messengers Ca^{2+}, IP_3, and calcium buffers parvalbumin (PV), and calbindin ($CD28k$). PV has binding sites for both Ca^{2+} and Mg^{2+}, and $CD28k$ has medium and high-affinity binding sites. For both PV and $CD28k$, multiple binding sites are accounted for by treating each site as independent of each other. Therefore the model explicitly solves for the concentrations for the unbounded and bound forms of both buffers: PV, PVB_c, PVB_m, $CD28k$, $CD28kB_1$, $CD28kB_2$, and $CD28kB_{12}$. All variables are solved on a one-dimensional domain of length 250 μm centered around $x = 0$ which is the location of the spine where PF and CF activation occur. $\Delta t = 0.1$ ms and spatial grid size is determined by the numerical algorithm used and order of the fractional operator.

6.2.1 Calcium Dynamics

For Ca^{2+}, the governing equation is given by

$$\frac{\partial \left[Ca^{2+}\right]}{\partial t} = R_{diffusion,Ca} + R_{buffering} + R_{channel} + R_{entry} - R_{extrusion} \qquad (6.2.1)$$

where each term is explained in further detail.

$R_{diffusion}$ describes the diffusion process acting on calcium. Santamaria et. al. conclude that although IP_3 undergoes anomalous diffusion due to localized trapping caused by geometrical complexities of spiny dendrites, the same does not apply to calcium in the cytosol because it is removed so quickly by calcium buffers that it does not travel far enough to be trapped. Therefore we model the diffusion of calcium with classical Gaussian diffusion:

$$R_{diffusion,Ca} = D_{Ca}\left(\nabla^2\left[Ca^{2+}\right]\right) \tag{6.2.2}$$

where D_{Ca} is the diffusion coefficient.

$R_{buffering}$ describes the rate of binding of calcium to buffers calbindin and parvalbumin. It is given by

$$\begin{aligned}
R_{buffering} &= -R_{PVBc} + R_{CD28k} \\
R_{PVBc} &= k_{on,PV,Ca}\left[Ca^{2+}\right][PV] - k_{off,PV,Ca}\left[PVB_c\right] \\
R_{CD28k} &= -k_{on,CD28k,h}\left[CD28k\right]\left[Ca^{2+}\right] - k_{on,CD28k,m}\left[CD28k\right]\left[Ca^{2+}\right] \\
&\quad -k_{on,CD28k,h}\left[CD28kB_2\right]\left[Ca^{2+}\right] - k_{on,CD28k,m}\left[CD28kB_1\right]\left[Ca^{2+}\right] \\
&\quad +k_{off,CD28k,m}\left[CD28kB_2\right] + k_{off,CD28k,h}\left[CD28kB_1\right] \\
&\quad +\left(k_{off,CD28k,h} + k_{off,CD28k,m}\right)\left[CD28kB_{12}\right]
\end{aligned}$$

where the on and off binding rates are given in Table 6.1, and whose values are taken from [16].

$R_{channel}$ describes the behavior of calcium flux across the ER membrane. It reflects the behavior of the following components: IP_3R, SERCA pumps, and leak current from the ER. IP_3R has a nonlinear response based on binding of either or both IP_3 and Ca^{2+}; it is responsible for releasing calcium from the ER (where it is stored in high concentration) through calcium-induced-calcium-release (CICR). SERCA pumps are responsible for resequestering calcium from the cytosol, back in the ER. And lastly, there is a general leak term representing calcium that is released from the ER through a leak current that, due to the small leak constant L, has a negligible contribution to the overall calcium dynamics. $R_{channel}$ is based on the Li-Rinzel minimal model of calcium dynamics [20] and is given by

$$\begin{aligned} R_{channel} &= a\left(1-\frac{[Ca^{2+}]}{[Ca^{2+}]_{ER}}\right)\left(\frac{h\,[Ca^{2+}]\,[IP_3]}{([Ca^{2+}]+d_{Ca})\,([IP_3]+d_{IP3})}\right)^3 \\ &\quad -\frac{V_{max}\,[Ca^{2+}]^2}{[Ca^{2+}]^2+k_{er}^2}+L\left(1-\frac{[Ca^{2+}]}{[Ca^{2+}]_{ER}}\right) \\ \frac{\partial h}{\partial t} &= \left(K_1-\left([Ca^{2+}]+K_1\right)h\right)K_2 \end{aligned} \tag{6.2.3}$$

The variable h is the probability of an inhibition site on the IP_3R, being unoccupied. Three other parameters of importance in the context of this section, are a, d_{IP3}, and d_{Ca}. a relates directly to the density of IP_3R in the system. IP_3R density is found to be larger in Purkinje cells compared to other types of neurons, and therefore the a value used in this model is the same as in [16], which is 10x the typical value used with the Li-Rinzel model applied to other types of neurons. d_{IP3} relates inversely with the sensitivity of the IP_3R to the concentration of IP_3 (meaning the higher the sensitivity, the lower the value of d_{IP3}). IP_3R in Purkinje cells have found to have decreased sensitivity, and therefore the value of d_{IP3} used in this model is also the same as in [16], which is 10x the typical value found in other studies dealing with other types of neurons. Details about parameter values are found in Table 6.1.

$R_{channel}$ is a key part of the calcium dynamics that is largely responsible for the supralinear calcium response during coincidental PF and CF stimulation, as we shall see in the results section. In Fig.6.2.1 we plot $R_{channel}$ as a function of IP_3 and calcium, for various h values. Notice that for low IP_3 concentrations, $R_{channel}$ varies slowly as calcium concentration increases. But for higher IP_3 concentrations, we observe a sharper increase in $R_{channel}$ values as calcium concentration increases, as seen by the distinct contour lines. This essentially means that due to coincidental detection (or signal integration) and CICR mechanisms of IP_3Rs, it is easier to go from low to high calcium concentrations, for a given high concentration of IP_3. At high $[IP_3]$ concentrations, more IP_3 is bound to its receptors. If a slight increase in calcium concentration is initiated by some other external signal (in this model it is caused by CF activation), enough calcium becomes bound to IP_3R to cause a small release of calcium from internal stores, which through the positive feedback mechanism that is calcium-induced-calcium-release, causes more calcium to be bound to the receptors, thus allowing more calcium influx into the cytosol. In addition, the higher the value of h, the greater the range of values that $R_{channel}$ encompasses, and the greater $R_{channel}$ contributes to the change in overall calcium dynamics, for given concentrations of IP_3 and calcium. Overall, we see that the $R_{channel}$ is greatest in value when both IP_3 and calcium concentrations are high, (due to the power term in Eq. 6.2.3 which represents behavior of IP_3R); this clearly coincides with the idea that the receptor together with the ER function as a signal integrator for second messengers like calcium and IP_3 [4].

R_{entry} is the term that simulates the depolarization of the Purkinje cell by the CF stimulus. In the original version given by [16], to represent a delocalized depolarization of the entire Purkinje cell, there is an influx of calcium into both the spine that the CF stimulus originates from, and the adjacent dendrite. In this modified model, we reflect this behavior by using J_s to determine the magnitude of calcium entry in the spine, only at the spine location (whose physical width is given by L and is inversely proportional to the linear spine density along the dendrite), while J_d is used as the magnitude of calcium

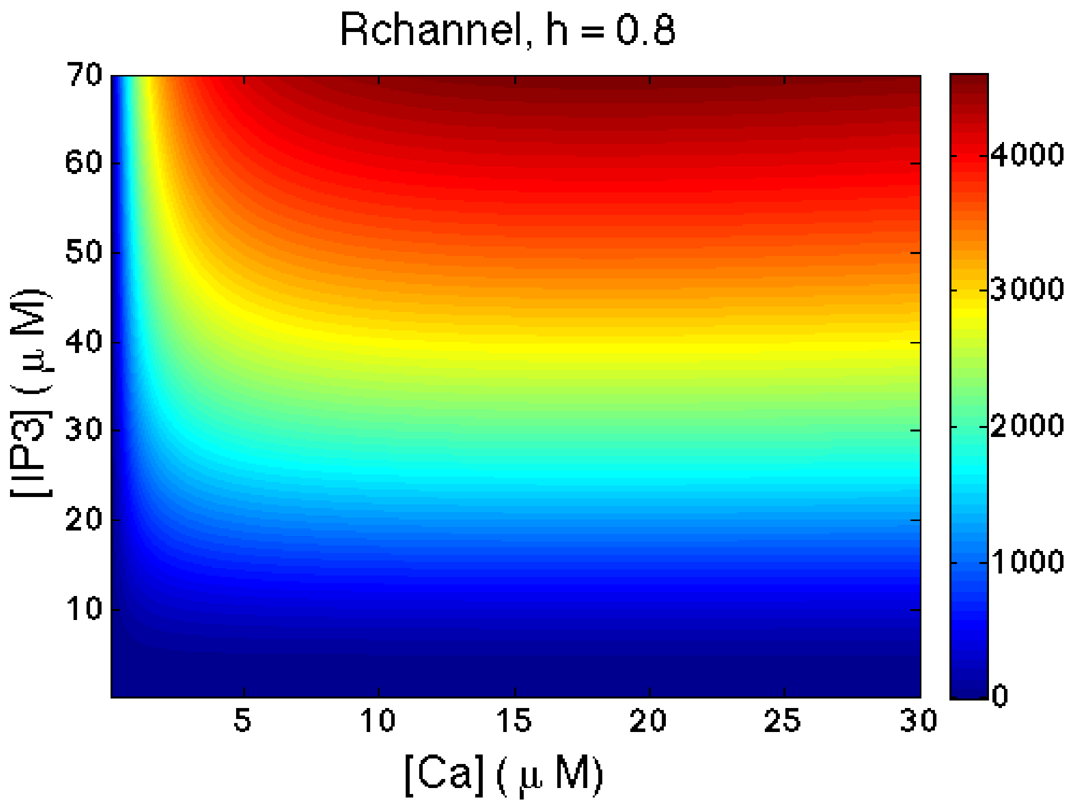


Figure 6.2.1: The behavior of $R_{channel}$ as a function of IP_3 and calcium concentrations. Notice that for low IP_3 concentrations, $R_{channel}$ varies slowly as calcium concentration increases. But for for higher IP_3 concentrations, we observe a sharp increase in $R_{channel}$ value when the calcium concentration is around 1 μM, as seen by the distinct contour lines. In addition, the higher the value of h, the greater the range of values that $R_{channel}$ encompasses, and the great $R_{channel}$ contributes to the change in overall calcium dynamics, for given concentrations of IP_3 and calcium.

entry everywhere else in the domain, which represents the dendrite.

$$\begin{aligned}
R_{entry,s} &= \left(|x| < \frac{L}{2}\right) J_s\,(t > \tau_1)\,(t < \tau_2)\left(\left[Ca^{2+}\right]_{ex} - \left[Ca^{2+}\right]\right) && (6.2.4)\\
R_{entry,d} &= \left(|x| > \frac{L}{2}\right) J_d\,(t > \tau_1)\,(t < \tau_2)\left(\left[Ca^{2+}\right]_{ex} - \left[Ca^{2+}\right]\right) && (6.2.5)\\
R_{entry} &= R_{entry,s} + R_{entry,d} && (6.2.6)
\end{aligned}$$

This entry term represents an influx of calcium proportional to the difference in calcium concentration between the extracellular space and the ER, that only occurs during the time period $\tau_1 < t < \tau_2$.

And finally, $R_{extrusion}$ represents the removal of calcium through the plasma membrane and is proportional to the amount of calcium above a threshold level (below which the membrane pumps providing the mechanisms for extrusion are inactive):

$$R_{extrusion} = \sigma P\left(\left[Ca^{2+}\right] - \left[Ca^{2+}\right]_T\left(\left[Ca^{2+}\right] > \left[Ca^{2+}\right]_T\right)\right)$$

P is the rate of pumping, and σ denotes the surface to volume ratio of the area of extrusion, and this value is different for the dendrite and spine compartments. In the modified model, this translates to a different extrusion value at the location of the spine, compared to everywhere else on the domain.

A general schematic of the components contributing to calcium dynamics and where they are occuring, is given in Fig. 6.2.2.

6.2.2 IP_3 Dynamics

The governing equation for IP_3 dynamics is given by

$$\frac{\partial\,[IP_3]}{\partial t} = R_{diffusion,IP3} + R_{IP3} \qquad (6.2.7)$$

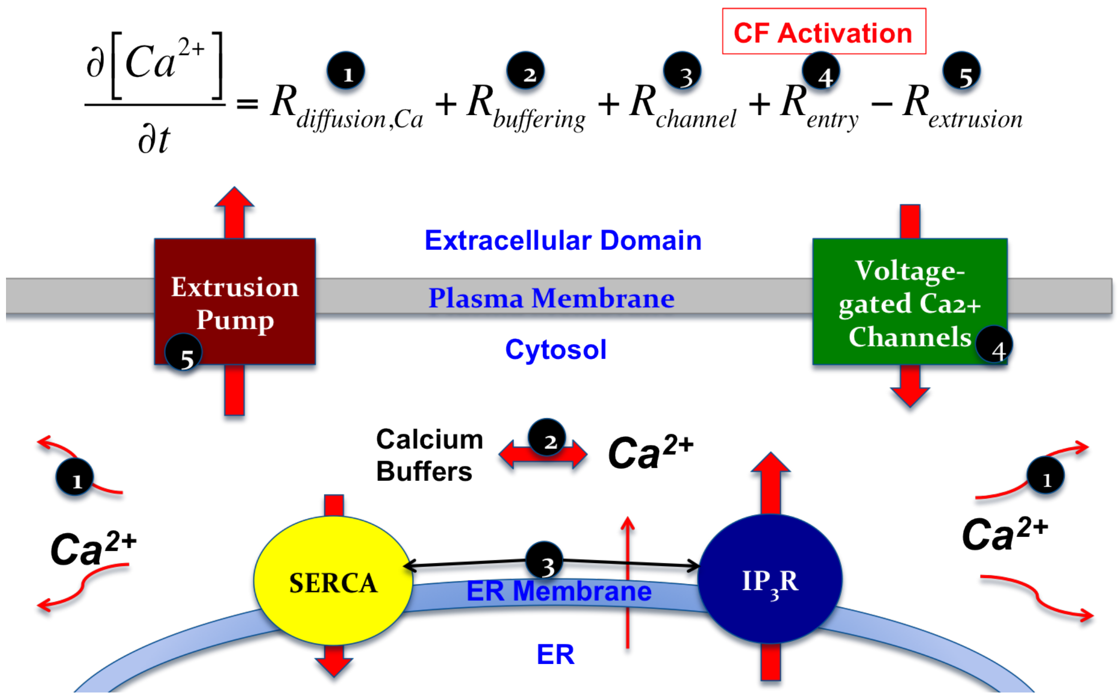


Figure 6.2.2: Schematic of the components contributing to calcium dynamics, including diffusion, calcium buffers, transport across the ER membrane, and transport across the plasma membrane.

Santamaria et. al. concludes that IP_3 undergoes anomalous diffusion due to localized trapping caused by geometrical complexities of spiny dendrites; therefore we choose to describe IP_3 diffusion with fractional diffusion:

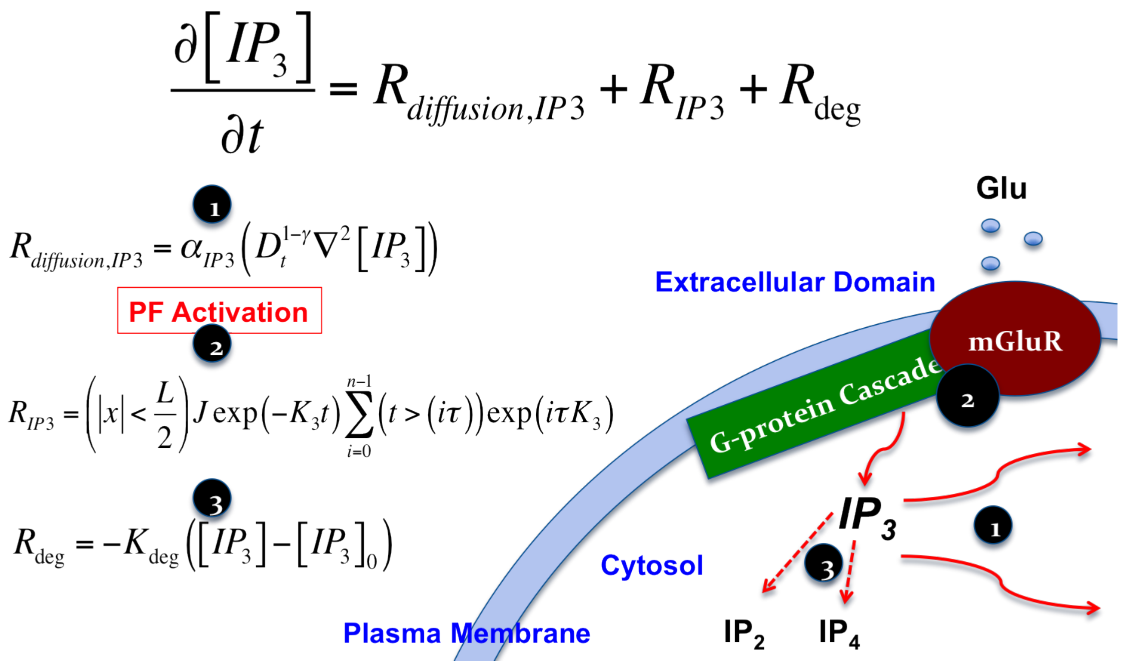
$$R_{diffusion,IP3} = D_{IP3}\left(D_t^{1-\gamma}\nabla^2\left[IP_3\right]\right)$$

where D_{IP3} is the diffusion coefficient and $D_t^{1-\gamma}$ is the fractional derivative operator. However, in this model, IP_3 influx is initiated at a location on the one-dimensional model, that corresponds to a single spine. That is, IP_3 influx caused by PF activation, occurs in a spine before spreading to the adjacent dendritic segment. In Hernjak's original model, the spine and dendrite are simulated as distinct compartments, and there is a flux term describing the flow of material from the location of the spine, to the adjacent section of the dendrite. There is no other diffusion term applied to the IP_3 located in the spine; in this way the signal there remains localized and only travels to the adjacent dendritic segment through the geometric bottleneck that is the spine neck. The resulting effect is elevated IP_3 concentration at the point of influx (at the location of the spine), which approximates the localized IP_3 signal that is sustained in a separate spine compartment (that undergoes no diffusion except as flux through the spine neck) as modeled by Hernjak et. al.

The source term R_{IP3} describes the production of IP_3 due to PF activation, as well as degradation of IP_3 to IP_2 and IP_4, and is given by

$$\begin{aligned} R_{IP3} &= \left(|x| < \frac{L}{2}\right) J_p exp\left(-K_3 t\right) \sum_{i=0}^{n-1} \left(t > \left(i\tau_3\right)\right) exp\left(i\tau_3 K_3\right) \\ &\quad -K_{deg}\left(\left[IP_3\right] - \left[IP_3\right]_0\right) \end{aligned} \tag{6.2.8}$$

PF activation instigates IP_3 production only in a single spine, and there is then flux of material between the spine and neighboring dendritic segment, through the spine neck.



$$\frac{\partial[IP_3]}{\partial t} = R_{diffusion,IP3} + R_{IP3} + R_{\deg}$$

$$R_{diffusion,IP3} = \alpha_{IP3}\left(D_t^{1-\gamma}\nabla^2[IP_3]\right)$$

$$R_{IP3} = \left(|x| < \frac{L}{2}\right) J \exp(-K_3 t) \sum_{i=0}^{n-1} \left(t > (i\tau)\right) \exp(i\tau K_3)$$

$$R_{\deg} = -K_{\deg}\left([IP_3] - [IP_3]_0\right)$$

Figure 6.2.3: Schematic of the components contributing to IP_3 dynamics, including a source term that represents the production of IP_3 resulting from PF activation, degradation, and diffusion.

The increase in $[IP_3]$ is modeled as a series of pulses that are spaced out by time τ_3, and whose magnitude and decay rate are controlled by parameters J_p and K_3.

A general schematic of the components contributing to IP_3 dynamics and where they are occuring, is given in Fig. 6.2.3

6.2.3 Calcium Buffers

The behavior of calcium buffer dynamics are given by the common governing equation

$$\frac{\partial[X]}{\partial t} = R_{diffusion,X} + R_X$$

where species X represents the bound and unbound forms of the buffers parvalbumin and calmodin. $R_{diffusion,X}$ is given by classical diffusion, $D_X \nabla^2 [X]$. While it is possible that compared to individual ions, the relatively large size of calcium buffer molecules causes them to undergo anomalous diffusion due to molecular crowding or localized traps, there is currently a lack of experimental or computational data to reflect this and therefore we do not consider the fractional diffusion of the buffer species, in the modified model.

R_X gives the expression of binding rates, and the expressions for all buffer species are the same as in the Appendix of [16], where instead of dendritic and spine compartments we have a single one-dimensional domain: $[X]_d = [X]_s = [X]$

6.2.4 Parameters

While the majority of parameters used in the modified model are taken from Table 1 in [16], there were several initial conditions that needed to be calculated, and several parameters that needed to be adjusted to recreate signals comparable to existing experimental results in literature.

To begin, initial buffer concentrations were determined by removing both PF and CF inputs, and letting all concentration values reach steady state. The steady state buffer concentrations under these conditions were then used as initial values. All initial values of species are reported in Table 6.1.

In addition, when running the modified model with only PF stimulation, the parameters relating to IP_3 influx, needed to be adjusted from the source values in [16]. Even in attempting to replicate the results using the original model by Hernjak et. al, the IP_3 dynamics during only PF stimulus were not able to be captured given the same parameter values as in Table 1 of [16]. The amplitude of the fast rise of IP_3 seen in the activated spine (Fig. 4 in [16]), were only replicated when parameters J_{IP3} and K_3 were adjusted. The model by Hernjak et. al., was solved by finite volume methods in the Virtual Cell program, while I reproduced their model in Matlab using finite difference methods; the discrepancy in parameter usage may possibly lie in this difference in numerical implementation.

Table 6.1: Parameter values and initial conditions used in the modified model.

Parameter	Description	Value	Source, Notes
D_{Ca}	Calcium diffusion coefficient	$223\ \mu m^2 s^{-1}$	[16]
D_{IP3}	IP_3 diffusion coefficient	$283\ \mu m^2 s^{-1}$	[16]
D_{PV}	PV diffusion coefficient	$43\ \mu m^2 s^{-1}$	[16]
D_{CD28k}	$CD28k$ diffusion coefficient	$28\ \mu m^2 s^{-1}$	[16]
$k_{on,CD28k,high}$	Forward rate coefficient for $CD28k$	$5.5\ \mu m^2 s^{-1}$	[16]
$k_{on,CD28k,med}$	Forward rate coefficient for $CD28k$	$43.5\ \mu m^2 s^{-1}$	[16]
$k_{off,CD28k,high}$	Reverse rate coefficient for $CD28k$	$2.6\ s^{-1}$	[16]
$k_{off,CD28k,med}$	Reverse rate coefficient for $CD28k$	$35.8\ s^{-1}$	[16]
$k_{on,PV,Ca}$	Forward rate coefficient for PV, calcium binding site	$107.0\ \mu M^{-1} s^{-1}$	[16]
$k_{on,PV,Mg}$	Forward rate coefficient for PV, magnesium binding site	$0.8\ \mu M^{-1} s^{-1}$	[16]
$k_{off,PV,Ca}$	Reverse rate coefficient for PV, calcium binding site	$0.95\ s^{-1}$	[16]
$k_{off,PV,Mg}$	Reverse rate coefficient for PV, magnesium binding site	$25.0\ s^{-1}$	[16]
a	IP_3R calcium release amplitude, proportionate to receptor density/abundance	$21,000.0\ \mu M s^{-1}$	[16]
$[Ca^{2+}]_{er}$	ER $[Ca^{2+}]$	$400\ \mu M$	[16]
d_{Ca}	IP_3R binding constant for calcium, inversely proportionate to sensitivity	$0.3\ \mu M$	[16]
d_{IP3}	IP_3R binding constant for IP_3, inversely proportionate to sensitivity	$20.0\ \mu M$	[16]
V_{max}	Amplitude of SERCA pump intake	$3.75\ \mu M s^{-1}$	[16]
k_{er}	Pump binding constant	$0.27\ \mu M s^{-1}$	[16]
L	Leak constant	$0.12\ \mu M$	[16]
K_1	Dissociation constant for IP_3R	$0.2\ \mu M$	[16]
K_2	Forward rate coefficient, calcium binding to inactivating IP_3R site	$2.7\ \mu M^{-1} s^{-1}$	[16]
h_0	Initial h value, probability of inhibition site on IP_3R being unoccupied	0.6	Fit
τ_1	Start time of CF calcium influx	$0.18\ s$	Fit
τ_2	End time of CF calcium influx	$0.185\ s$	Fit
J_s	Magnitude of CF calcium entry signal at the spine location	$6\ s^{-1}$	Fit
J_d	Magnitude of CF calcium entry signal along the rest of the dendrite	$3\ s^{-1}$	Fit
$[Ca^{2+}]_{ex}$	Extracellular calcium concentration	$1000\ \mu M$	[16]
J_{IP3}	IP_3 pulse magnitude	$1200 \mu M\ s^{-1}$	Fit
τ_3	Time between IP_3 pulses	$0.012\ s$	[16]
K_3	IP_3 pulse decay factor	$10 * 1.188\ s^{-1}$	[16]
K_{deg}	IP_3 degradation rate	$0.14\ s^{-1}$	[16]
P	Calcium extrusion pumping rate	$8.0\ \mu m\ s^{-1}$	[16]
$[Ca^{2+}]_T$	Threshold $[Ca^{2+}]$	$0.2\ \mu M$	[16]
$[Mg^{2+}]$	$[Mg^{2+}]$	$590\ \mu M$	[16]
$[PV]_t$	Total $[PV]$	$40\ \mu M$	[16]
$[CD28k]_t$	Total $[CD28k]$	$40\ \mu M$	[16]
$[PV]_0$	Initial $[PV]$	$1.63\ \mu M$	Calc.
$[PVBc]_0$	Initial $[PVBc]$	$8.14\ \mu M$	Calc.
$[PVBm]_0$	Initial $[PVBm]$	$30\ \mu M$	Calc.
$[CD28k]_0$	Initial $[CD28k]$	$34.615\ \mu M$	Calc.
$[CD28kB1]_0$	Initial $[CD28kB1]$	$3.304\ \mu M$	Calc.
$[CD28kB2]_0$	Initial $[CD28kB2]$	$1.9\ \mu M$	Calc.
$[CD28kB12]_0$	Initial $[CD28kB12]$	$0.181\ \mu M$	Calc.
$[Ca^{2+}]_0$	Initial $[Ca^{2+}]$	$0.045\ \mu M$	[16]
$[IP_3]_0$	Initial $[IP_3]$	$0.16\ \mu M$	[16]

6.2.5 A Note on Numerical Methodology

There are several details to consider when deciding what numerical methods to apply to the set of equations constituting this adapted model. The nonlinear aspect of several source terms (including $R_{channel}$) that contribute to the governing equation for calcium dynamics, create stiff equations that require a small timestep for stability, especially when using explicit methods. In addition, the diffusion coefficients for calcium and IP_3 are large, and also require small timesteps to maintain stability. Using implicit methods will generally relax the stringent stability requirements of small timesteps associated with explicit numerical methods, but developing an implicit framework with such nonlinear source terms as in Eq.6.2.1, is quite complicated.

There have been methods developed to deal with fractional diffusion equations with nonlinear source terms [21]; however their numerical methods proceed from the Riemann-Liouville definition of the fractional derivative operator, and further work is needed to integrate their approach with the finite summation methods developed in the last few chapters of this thesis.

Therefore, explicit methods have been chosen to implement this modified model. However despite the ease with which explicit methods for fractional diffusion can be integrated with this modified model, the general nature of explicit numerical methods results in a strict relationship between fractional order γ, and simulation parameters Δx and Δt, provided diffusion coefficients are fixed, as they are in this context. As demonstrated in [57], for a one-dimensional system, the required relationship between parameters to ensure a stable simulation is

$$\frac{D\Delta t^{\gamma}}{\Delta x^{2}} \leq \frac{1}{2^{2-\gamma}} \tag{6.2.9}$$

One of the major challenges in developing this modified model, was in choosing appropriate timestep and grid size parameters in order to balance the execution time with inaccuracies that might result from too large a grid size, especially since localized signals are important in this model, and localization is lost once Δx becomes too large.

It should also be noted that for very small step sizes Δt, there is also an error introduced in the numerical algorithms for fractional diffusion, due to the Δt^{γ} term that is present in both explicit and implicit numerical methods to solve the fractional diffusion equation. Depending on the value of γ, when Δt is extremely small, this term can diverge greatly from the analogous term (simply timestep taken to the first power) for classical diffusion, and the result is a skewed solution in time where the fractional diffusion profile diffuses more quickly than the classical diffusion profile (all other conditions held constant), which is clearly an incorrect result. To bypass this numerical artifact, we would require implicit numerical time marching schemes for all equations that contribute to the modified model. As we have already stated, this is not straightforward because of highly nonlinear source terms that make certain governing equations very stiff in nature, but this is an open problem to consider in future endeavors and in refining the modified model to make it more robust to parameter changes.

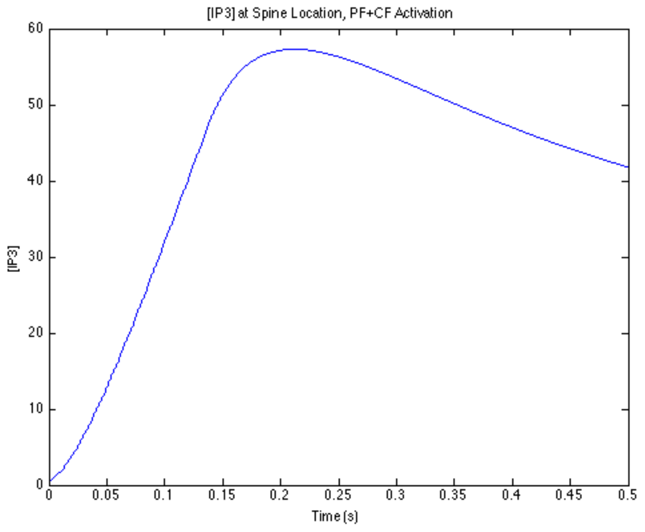
6.3 Preliminary Results and Discussion

6.3.1 Modified Model With Fractional Diffusion

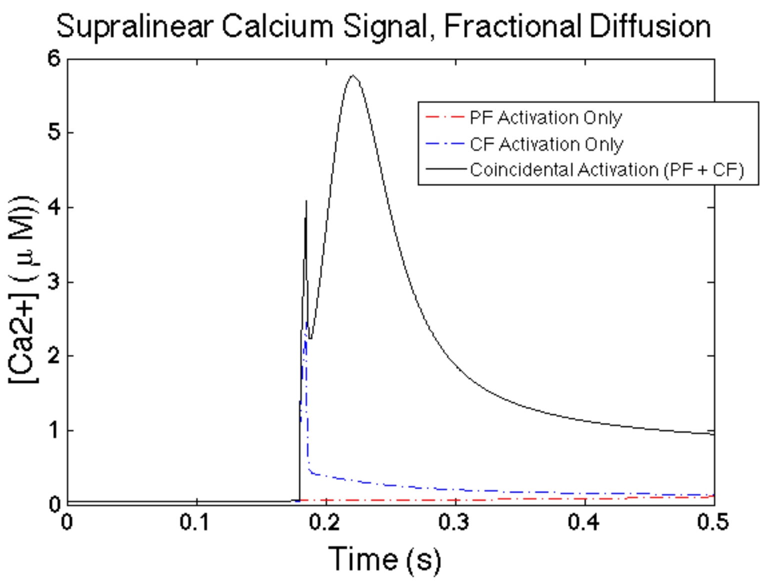
We present in Fig. 6.3.1 the results of our modified model with fractional diffusion of IP_3, with $\gamma = 0.7$, which is within the range of γ values that Santamaria et. al. found corresponded to spiny dendrites on Purkinje neurons [43]. The figure shows the IP_3 and calcium dynamics with PF activation, CF activation, and coincidental activation. In part A of Fig. 6.3.1, we see the IP_3 signal resulting from combined CF and PF inputs; this signal is the same as for PF activation alone, and remains at equilibrium concentration for CF activation alone. In 6.3.1 B, we see that under PF activation alone, the rise in calcium concentration is very small. Under CF activation alone, we see an expected spike in calcium concentration due to the dynamics presented in 6.2.4. The amplitude of this spike is approximately 2.5 μM. However, under coincidental activation (PF + CF activation), the resulting calcium dynamics clearly represent supralinear behavior: that is, the result is greater than a linear sum of the calcium response under PF activation alone, and CF activation alone. The key to this supralinear response lies in the dynamics representing the IP_3R (as mentioned in the discussion about $R_{channel}$ in Section 6.2.1) and high concentration of both IP_3 and calcium. The calcium response has two components: a fast spike, followed by a slower rise and decay. The fast spike has an amplitude that is greater than the sum of the calcium spike under CF activation alone, and the small calcium rise under PF alone; this represents supralinear behavior. In addition, the following slower rise and decay is due to $R_{channel}$ which represents the activity of the IP_3R. The dip in calcium concentration immediately after the fast spike, is due to fast acting calcium buffers trying to counter the rise in cytosolic calcium; however, they are completing with the behavior of IP_3R (represented by $R_{channel}$), which continues to have a large response due to simultaneous high concentration levels of calcium and IP_3 which is decaying slowly due to the decay factor K_3 in Eq. 6.2.8, as well as the fact that we are modeling diffusion with fractional diffusion (which as established in previous chapters, results in sustained higher concentrations for longer periods of time). As you can see in Fig. 6.3.1, the peak IP_3 concentration coincides with the time of CF stimulus; this time, set by τ_1 in Table 6.1, was selected to give the maximum supralinear response.

6.3.2 Modified Model With Regular Diffusion

Figure 6.3.2 shows the results of our modified model when the fractional diffusion of IP_3 is changed to regular diffusion (a simple switch by setting $\gamma = 1$). In A, it is clear



A)

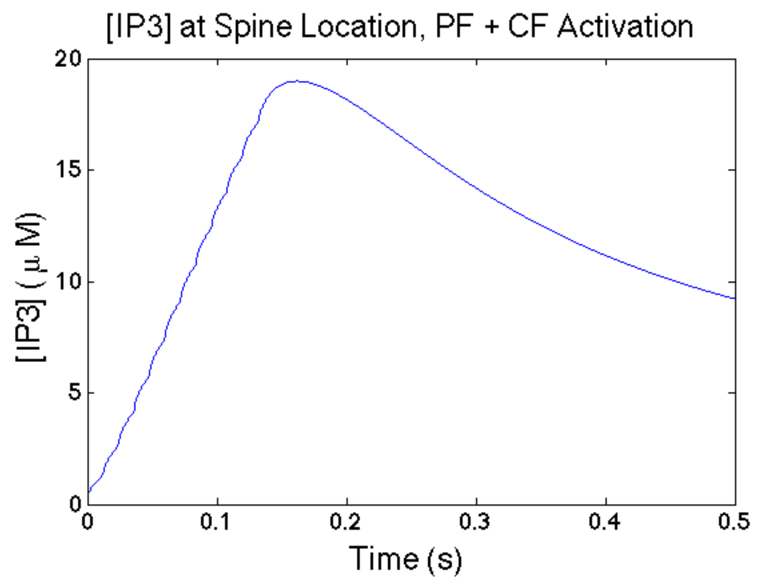


B)

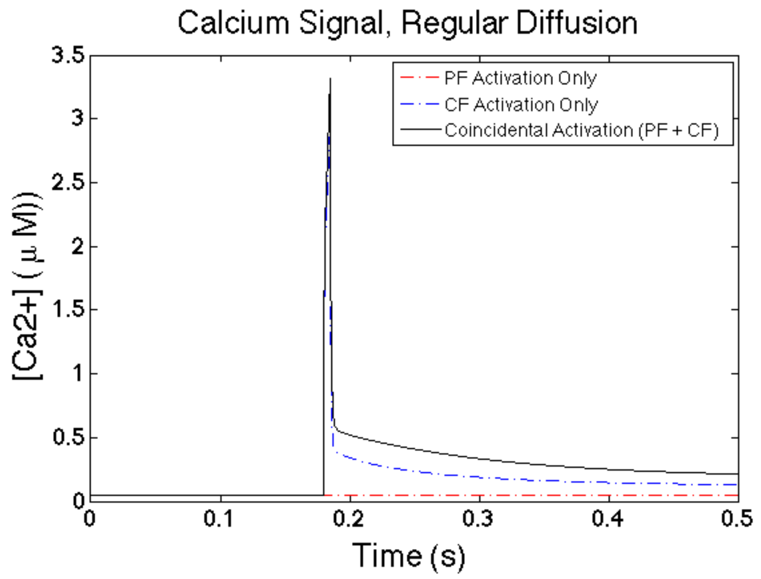
Figure 6.3.1: Preliminary results for the modified model with fractional diffusion, $\gamma = 0.7$: demonstration of PF activation, CF activation, and supralinear calcium dynamics during coincidental activation. **A)** $[IP_3]$ at the location of the spine, during PF + CF coincidental activation. Since the governing expression for IP_3 does not depend at all on calcium concentrations, the dynamics seen in this figure are the same as during solely PF activation. **B)** The graph shows the calcium dynamics during only PF activation (red), only CF activation (blue), and coincidental activation (PF + CF). It is clear that under coincidental activation the model produces supralinear calcium dynamics.

that IP_3 reaches much lower concentration levels over the same time period of simulation, and decays at a faster rate (compared to Fig. 6.3.1). In B, we see that the calcium response for PF or CF activation alone, is similar to Fig. 6.3.1B; however, the response during coincidental activation, fails to reproduce the supralinear dynamics that we see in Fig. 6.3.2. This suggests that the fractional diffusion component of our modified model is required to ensure that IP_3 concentration is high enough at the location of spine of interest, to induce a proper supralinear calcium response.

It should be noted that the grid size for the results in this section, are not the same as in Section 6.3.1; with regular diffusion, we find that we can make our grid more refined and still maintain stability. With fractional diffusion, to achieve the same refinement in grid size, we must decrease our time step further, which, because of increases in computational time and memory requirements, we found to be unfeasible to simulate with the use of a single personal computer; a cluster is likely required to calculate those results efficiently. However, we also refer to the discussion on numerical considerations in Section 6.2.5 and emphasize that in order to concretely compare the results of our model using fractional diffusion, and using regular diffusion, we would need to use implicit numerical methods for all governing equations (particularly for IP_3 and calcium), so that the time step can be increased, perhaps by a factor of 5 or 10. This would also allow us to decrease the spatial grid size and better localize some of our signals (especially related to PF activation). The result would be a more accurate model that is more robust to parameter changes. Nevertheless, the preliminary results given in Fig. 6.3.1 and 6.3.2 are logical given what we know about the nature of anomalous diffusion and are qualitatively in line with what we would expect from fractional diffusion of IP_3 in comparison to normal diffusion.



A)



B)

Figure 6.3.2: Preliminary results for the modified model with regular diffusion: demonstration of PF activation, CF activation, and lack of supralinear calcium dynamics during coincidental activation. **A)** $[IP_3]$ at the location of the spine, during PF + CF coincidental activation. Since the governing expression for IP_3 does not depend at all on calcium concentrations, the dynamics seen in this figure are the same as during solely PF activation. **B)** The graph shows the calcium dynamics during only PF activation (red), only CF activation (blue), and coincidental activation (PF + CF). It is clear that under coincidental activation the modified model with regular diffusion of IP_3, fails to produce supralinear calcium dynamics.

6.4 Conclusion

In this chapter, we have explored a particular application of the time-fractional diffusion equation, in modeling the anomalous diffusion of IP_3 in spiny dendrites of Purkinje neurons. We have shown that we can use fractional diffusion in predictive modeling, to simplify an existing, more complicated geometrical model developed by Hernjak et. al., which explicitly takes into account spine and dendritic compartments, as well as the flux between the two through a narrow spine length. After removing all references to explicit geometry of the protruding spines and replacing these geometric details with a fractional diffusion element in the governing equation for IP_3 dynamics, the modified model was still able to capture the most important functionality of the original model related to the induction of LTD, as shown in Section 6.3. There are still some open ended numerical considerations to pursue in order to make the modified model more accurate and robust to parameter changes, but in this chapter we have laid out the motivation for our modified model, the details of our model, and preliminary results demonstrating our initial predictions which were based on well known and observed characteristics of anomalous diffusion (sustained, slowly diffusing peaked profile).

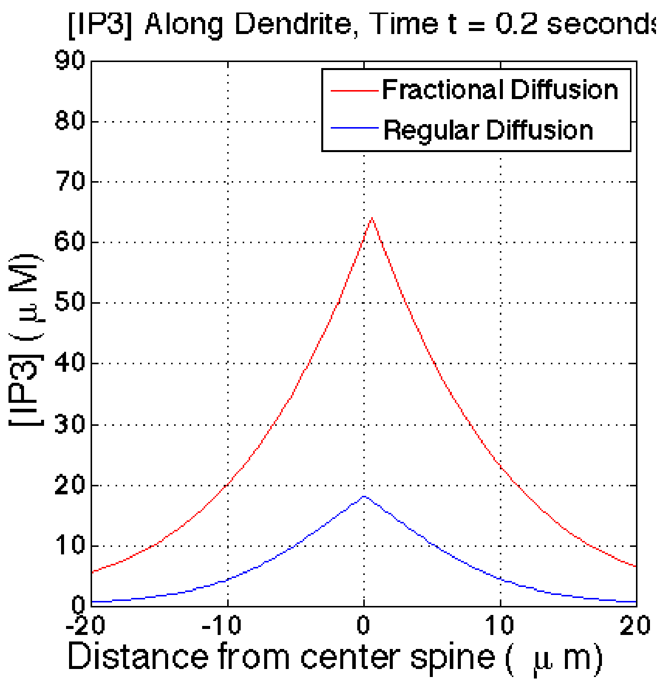
Chapter 7

Conclusion and Future Directions

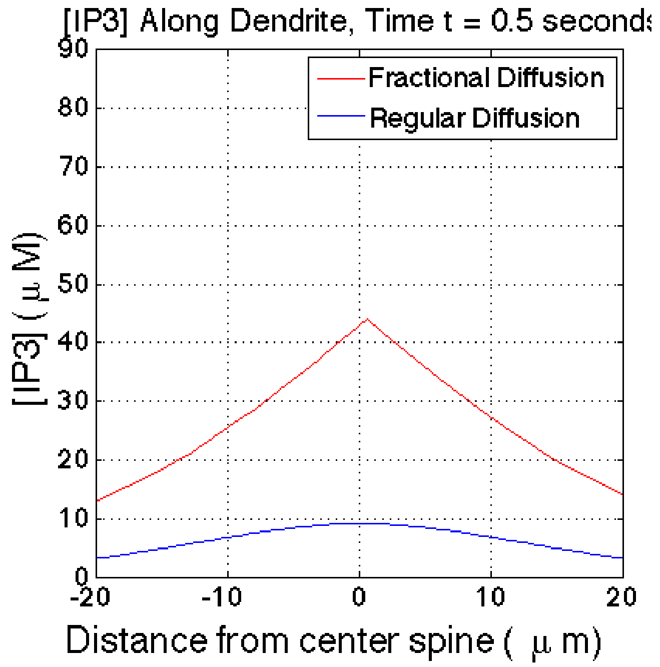
This thesis represents a multifaceted exploration of the fractional diffusion equation. In Chapter 2 we focused on its theoretical derivations originating from the random walk scenario and the trajectories of diffusing particles; Chapters 3-5 then explored the practical considerations of numerically simulating this equation, including questions related to stability, efficiency, and algorithmic complexity, and the development of a new computational algorithm. In Chapter 6 we delved into a specific neuroscience application in which the time-fractional diffusion equation was used to simulate the dynamics of anomalously diffusing IP_3, and integrated into a larger model of the signaling events leading up to the induction of LTD. Finally, in this chapter, we take a brief look at several future directions where there are interesting questions to be asked about how fractional diffusion can accurately reflect the true behavior of diffusing particles, and what kinds of problems can be answered by integrating fractional diffusion into broader biological models.

7.1 Fractional Diffusion and Effects on LTD

In Chapter 6 we have established that we can use fractional diffusion to simplify an existing computational model of the signaling events leading to LTD, and still capture an important functionality of the original model (the required signaling inputs resulting in a supralinear calcium response that matches the true experimental results). When discussing our modified model results, we focused on species concentrations at the location of a single spine where LTD would occur due to the localized PF input. However, it is interested to observe how fractional diffusion affects the diffusion profile of IP_3 along the entire dendrite, and what consequences this has on calcium signals in neighboring spines or dendritic segments. Fig. 7.1.1 demonstrates the difference in the diffusion profile when our modified model uses fractional diffusion, and regular diffusion. It is clear that elevated $[IP_3]$ occurs along the dendrite near the location of the single spine that was activated by the parallel fiber of interest. Given that IP_3R are spread along the ER through the dendrite and protruding spines, does elevated $[IP_3]$ make it easier for localized calcium influx at these neighboring locations, to trigger some sort of elevated calcium response? Since a single Purkinje neuron can be connected to many parallel fibers, each synapsing onto a single spine, it would be interesting to consider how anomalous diffusion of IP_3 along the dendrite, affects coincidental PF and CF activation, and the induction of LTD, in several spines in the same vicinity. In addition, while we have only considered a one-dimensional problem in Chapter 6, we can construct a two-dimensional domain and make use of the two-dimensional explicit and implicit numerical methods discussed in Chapters 3-5. In that scenario we could consider how the signaling related to LTD, is different from one ‘side’ of a dendrite to another - how does the size and radius of the dendrite affect the propagation of these signals and the spread of LTD?



A



B

Figure 7.1.1: Diffusion profiles along the length of the dendrite. **A)** The IP_3 diffusion profiles from our model when using fractional diffusion, and using regular diffusion, at $t = 0.1$ s. **B)** The IP_3 diffusion profiles from our model when using fractional diffusion, and using regular diffusion, at $t = 0.5$ s. It is clear that at all times in the simulation, the diffusion profile resulting from fractional diffusion is greater in magnitude and retains the characteristic persistent central peak, in comparison with the Gaussian diffusion profile (which, while in A, starts out peaked due to a localized initial condition and source term for IP_3, very quickly smoothes out as seen in B.

7.2 Calcium Signaling

There is potential to use fractional diffusion to model underlying biophysical behavior of calcium. As discussed in [1], calcium signaling is important on many spatiotemporal times scales. In all cell types, calcium is involved as a signaling molecule on all spatial scales, from the nanodomain to full intracellular and intercellular calcium waves.

Tan et. al. take a closer look at modeling calcium sparks in cardiac myocytes, with fractional diffusion [50]. Their research was motivated by the fact that existing models based on classical Fickian diffusion, failed to accurately capture spatial characteristics of recorded calcium sparks, especially with regards to full-width-half-maximum (FWHM) measurements, which were not reproduce able using models with classical diffusion (referred to as the FWHM paradox). Possibly underlying reasons for this discrepancy is related to the ongoing discussion in the scientific community about the structure of the cytoplasm being more complex than previously thought, having characteristic on multiple length scales, as well as viscoelastic properties, none of which are taken into account with classical diffusion models. Tan et. al find their model best fits spatiotemporal characteristics of experimental data and is able to FWHM paradox, when they use anomalous space diffusion (i.e., the fractional operator is in relation to the spatial derivative, not the time derivative). Chen builds upon these results and uses space-fractional diffusion to model the transport of calcium at the scale of intracellular waves, and are able to recreate calcium waves under physiological conditions and replicate wave velocities observed in experimental data [8].

Given this growing interest in modeling calcium sparks and general calcium diffusion in terms of fractional diffusion, there are several important questions to consider. Firstly, while we have shown in this thesis that there are clear biophysical mechanisms that cause anomalous diffusion which can be modeled with the time-fractional diffusion equation, what are the underlying mechanisms behind space-fractional diffusion? In the same way that anomalous diffusion is non-Markovian in time (with the state of the system at the next time step depending on the history of the system), space-fractional diffusion is non-Markovian in space. That is, when numerically simulating this type of equation [49], the state of the system at a particular location, depends on the entire domain of the system, not just local values of neighboring grid points. It is interesting to consider what kind of physical behavior leads to this spatial non-locality. [50, 8] suggest some hypotheses that include the long range effects of an electric field caused by the calcium ions themselves, or complex characteristic structures of the cytoplasm. On a larger spatial scale, we can also consider whether the modeling of calcium with fractional diffusion, is able to predict wave size, propagation distance, or details about the conditions under which calcium sparks and individual calcium-release-units (CRUs), may or may not coalesce into a larger intracellular wave.

7.3 ATP Signaling in Astrocyte Networks

Several times in this thesis we have mentioned that our original motivation for exploring anomalous diffusion and the fractional diffusion equation, was observing the unusual peaked diffusion profiles of ATP in glial cells in the retina (Fig. 7, [38]). This suggested that ATP possibly undergoes anomalous diffusion in the extracellular space; some potential underlying mechanisms for this behavior include a general complex and tortuous extracellular environment, which is well known to be extremely complex in nature. Given these observed profiles, it would be interesting to include fractional diffusion of ATP in a signaling model, such as the one developed in [29], where ATP is a vital signaling component in the propagation of intercellular calcium waves in an astrocyte network. In much the same manner as the intracellular calcium waves discussed in the last section, some possible questions to explore would be the effects of ATP subdiffusion on the propagation of the astrocytic intercellular calcium waves, including wave size and traveling velocity, or the pattern of activation of astrocytes in the network.

Bibliography

[1] GJ Augustine, Fidel Santamaria, and Keiko Tanaka, *Local calcium signaling in neurons*, Neuron **40** (2003), 331–346.

[2] Boris Baeumer, *Subordinated advection-dispersion equation for contaminant transport*, Water Resources . . . **37** (2001), no. 6, 1543–1550.

[3] Gerd Baumann, Robert F Place, and Zeno Földes-Papp, *Meaningful interpretation of subdiffusive measurements in living cells (crowded environment) by fluorescence fluctuation microscopy.*, Current pharmaceutical biotechnology **11** (2010), no. 5, 527–43.

[4] Michael J Berridge, *Neuronal Calcium Signaling*, Neuron **21** (1998), 13–26.

[5] Armin Biess, Eduard Korkotian, and David Holcman, *Barriers to diffusion in dendrites and estimation of calcium spread following synaptic inputs.*, PLoS computational biology **7** (2011), no. 10, e1002182.

[6] C. Chen, F. Liu, and K. Burrage, *Finite difference methods and a Fourier analysis for the fractional reaction–subdiffusion equation*, Applied Mathematics and Computation **198** (2008), no. 2, 754–769.

[7] C.M. Chen, F. Liu, I. Turner, and V. Anh, *Numerical schemes and multivariate extrapolation of a two-dimensional anomalous sub-diffusion equation*, Numerical Algorithms (2009), 1–21.

[8] Xi Chen, Jianhong Kang, Ceji Fu, and Wenchang Tan, *Modeling calcium wave based on anomalous subdiffusion of calcium sparks in cardiac myocytes.*, PloS one **8** (2013), no. 3, e57093.

[9] Graham L Collingridge, Stephane Peineau, John G Howland, and Yu Tian Wang, *Long-term depression in the CNS.*, Nature reviews. Neuroscience **11** (2010), no. 7, 459–73.

[10] ZEA Fellah, C Depollier, and M Fellah, *Application of fractional calculus to the sound waves propagation in rigid porous materials: Validation via ultrasonic measurements*, Acta Acustica united with . . . **88** (2002), 34–39.

[11] Joseph E. Flaherty, *Course notes - partial differential equations*, University Lecture, http://www.cs.rpi.edu/ flaherje/.

[12] Neville J. Ford and A. Charles Simpson, *The numerical solution of fractional differential equations: Speed versus accuracy*, Numerical Algorithms **26** (2001), 333–346.

[13] Rudolf Gorenflo, Francesco Mainardi, D Moretti, G Pagnini, and P Paradisi, *Discrete random walk models for space-time fractional diffusion*, Chemical physics **284** (2002), no. 1, 521–541.

[14] S Havlin and D Ben-Avraham, *Diffusion in disordered media*, Advances in Physics **51** (2002), no. 1, 187–292.

[15] B. Henry, T. Langlands, and S. Wearne, *Fractional Cable Models for Spiny Neuronal Dendrites*, Physical Review Letters **100** (2008), no. 12, 128103.

[16] Nicholas Hernjak, Boris M Slepchenko, Kathleen Fernald, Charles C Fink, Dale Fortin, Ion I Moraru, James Watras, and Leslie M Loew, *Modeling and analysis of calcium signaling events leading to long-term depression in cerebellar Purkinje cells.*, Biophysical journal **89** (2005), no. 6, 3790–806.

[17] Arthur Konnerth, *Brief dendritic calcium signals initiate long-lasting synaptic depression in cerebellar Purkinje cells*, Proceedings of the . . . **89** (1992), no. August, 7051–7055.

[18] TAM Langlands and BI Henry, *The accuracy and stability of an implicit solution method for the fractional diffusion equation*, Journal of Computational Physics **205** (2005), no. 2, 719–736.

[19] Jing-Rebecca Li, *A fast time stepping method for evaluating fractional integrals*, SIAM Journal on Scientific Computing **31** (2010), no. 6, 4696–4714.

[20] John Li, Yue-Xian; Rinzel, *Equations for InsP3 Receptor-mediated*, J. Theor. Biol. **166** (1994), 461–473.

[21] F. Liu, C. Yang, and K. Burrage, *Numerical method and analytical technique of the modified anomalous subdiffusion equation with a nonlinear source term*, Journal of Computational and Applied Mathematics **231** (2009), no. 1, 160–176.

[22] F. Liu, P. Zhuang, V. Anh, I. Turner, and K. Burrage, *Stability and convergence of the difference methods for the space–time fractional advection–diffusion equation*, Applied Mathematics and Computation **191** (2007), no. 1, 12–20.

[23] María López-Fernández, Christian Lubich, and Achim Schädle, *Adaptive, fast, and oblivious convolution in evolution equations with memory*, SIAM Journal on Scientific Computing **30** (2008), no. 2, 1015–1037.

[24] C. Lubich, *Convolution quadrature and discretized operational calculus i*, Numerische Mathematik **52** (1988), no. 2, 129–145.

[25] ______, *Convolution quadrature and discretized operational calculus ii*, Numerische Mathematik **52** (1988), 413–425.

[26] Ch. Lubich, *Discretized fractional calculus*, SIAM Journal on Mathematical Analysis **17** (1986), no. 3, 704–719.

[27] Christian Lubich, *Convolution quadrature revisited*, BIT Numerical Mathematics **44** (2004), 503–514.

[28] Christian Lubich and Achim Schädle, *Fast convolution for nonreflecting boundary conditions*, SIAM Journal on Scientific Computing **24** (2002), no. 1, 161–182.

[29] Christopher L Macdonald, Diana Yu, Marius Buibas, and Gabriel a Silva, *Diffusion modeling of ATP signaling suggests a partially regenerative mechanism underlies astrocyte intercellular calcium waves.*, Frontiers in neuroengineering **1** (2008), 1.

[30] Richard L Magin, *Fractional Calculus in Bioengineering*, Begell House Publishers, January 2006.

[31] R.L. Magin and M. Ovadia, *Modeling the Cardiac Tissue Electrode Interface Using Fractional Calculus*, Journal of Vibration and Control **14** (2008), no. 9-10, 1431–1442.

[32] F. Mainardi, A. Mura, G. Pagnini, and R. Gorenflo, *Sub-diffusion equations of fractional order and their fundamental solutions*, Mathematical Methods in Engineering (2006), 20–48.

[33] B. Mathieu, P. Melchior, a. Oustaloup, and Ch. Ceyral, *Fractional differentiation for edge detection*, Signal Processing **83** (2003), no. 11, 2421–2432.

[34] W. McLean and K. Mustapha, *Convergence analysis of a discontinuous Galerkin method for a sub-diffusion equation*, Numerical Algorithms **52** (2009), no. 1, 69–88.

[35] Monica R Metea and Eric A Newman, *Calcium signaling in specialized glial cells*, Glia **54** (2006), no. 7, 650–655.

[36] R. Metzler and J. Klafter, *The random walk's guide to anomalous diffusion: a fractional dynamics approach*, Physics Reports **339** (2000), no. 1, 1–77.

[37] Parviz Moin, *Fundamentals of Engineering Numerical Analysis*, Cambridge University Press, 2010.

[38] E A Newman, *Propagation of intercellular calcium waves in retinal astrocytes and Müller cells*, The Journal of neuroscience : the official journal of the Society for Neuroscience **21** (2001), no. 7, 2215–2223.

[39] Keith B. Oldham and Jerome Spanier, *The Fractional Calculus: Theory and Applications of Differentiation and Integration to Arbitrary Order*, Dover Books on Mathematics, April 2006.

[40] I. Podlubny, *Fractional differential equations*, Academic Press New York, 1999.

[41] I. Podlubny, A. Chechkin, T. Skovranek, Y.Q. Chen, and B.M. Vinagre Jara, *Matrix approach to discrete fractional calculus II: partial fractional differential equations*, Journal of Computational Physics **228** (2009), no. 8, 3137–3153.

[42] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery, *Numerical recipes in C*, Cambridge Univ. Press Cambridge MA, USA:, 1992.

[43] Fidel Santamaria, Stefan Wils, Erik De Schutter, and George J Augustine, *Anomalous diffusion in Purkinje cell dendrites caused by spines.*, Neuron **52** (2006), no. 4, 635–48.

[44] M J Saxton, *Anomalous subdiffusion in fluorescence photobleaching recovery: a Monte Carlo study.*, Biophysical journal **81** (2001), no. 4, 2226–40.

[45] Achim Schädle, María López-Fernández, and Christian Lubich, *Fast and oblivious convolution quadrature*, SIAM Journal on Scientific Computing **28** (2006), no. 2, 421–438.

[46] N. Sebaa, Z.E.a. Fellah, W. Lauriks, and C. Depollier, *Application of fractional calculus to ultrasonic wave propagation in human cancellous bone*, Signal Processing **86** (2006), no. 10, 2668–2677.

[47] Christine Selhuber-Unkel, Pernille Yde, Kirstine Berg-Sø rensen, and Lene B Oddershede, *Variety in intracellular diffusion during the cell cycle.*, Physical biology **6** (2009), no. 2, 025015.

[48] Eugeniusz Soczkiewicz, *Application of Fractional Calculus in the Theory of Viscoelasticity*, Molecular and Quantum Acoustics **23** (2002), 397–404.

[49] Charles Tadjeran and Mark M. Meerschaert, *A second-order accurate numerical method for the two-dimensional fractional diffusion equation*, Journal of Computational Physics **220** (2007), no. 2, 813–823.

[50] Wenchang Tan, Chaoqi Fu, Ceji Fu, Wenjun Xie, and Heping Cheng, *An anomalous subdiffusion model for calcium spark in cardiac myocytes*, Applied Physics Letters **91** (2007), no. 18, 183901.

[51] Loukas Vlahos, Heinz Isliker, Yannis Kominis, and Kyriakos Hizanidis, *Normal and Anomalous Diffusion: A Tutorial*, 2008.

[52] V. Volterra, *Leçons sur la théorie mathématique de la lutte pour la vie*, Paris, France (1931).

[53] Gerhard Werner, *Fractals in the nervous system: conceptual implications for theoretical neuroscience.*, Frontiers in physiology **1** (2010), no. July, 15.

[54] Bruce J West, *Fractal physiology and the fractional calculus: a perspective.*, Frontiers in physiology **1** (2010), no. October, 12.

[55] Bruce J West, Paolo Grigolini, Ralf Metzler, and Theo F Nonnenmacher, *Fractional Diffusion and Levy Stable Processes*, Physical Review E **55** (1997), no. 1, 99–106.

[56] D Yu, M Buibas, SK Chow, IY Lee, Z Singer, and GA Silva, *Characterization of Calcium-Mediated Intracellular and Intercellular Signaling in the rMC-1 Glial Cell Line*, Cellular and Molecular Bioengineering **2** (2009), no. 1, 144–155.

[57] SB Yuste and L. Acedo, *On an explicit finite difference method for fractional diffusion equations*, Arxiv preprint cs/0311011 (2003).