VALIDATION AND ANALYSIS OF NUMERICAL METHODS FOR SOLVING FRACTIONAL-ORDER DIFFERENTIAL EQUATIONS

A Thesis Proposal

by

CONNOR L. MITCHELL

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Dr. Alan Freed
Committee Members,	Dr. Anastasia Muliana
	Dr. Raytcho Lazarov
Head of Department,	Dr. Andreas Polycarpou

May 2018

Major Subject: Mechanical Engineering

ABSTRACT

Fractional-order differential equations have been very effective in modeling the behavior of various phenomena that have been otherwise difficult to accurately simulate. A new class of predictor-corrector schemes for solving nonlinear fractional-order differential equations has been proposed with claims of both increased accuracy and decreased computational cost. The purpose of this study is to test and/or verify these claims against an existing standard, while also developing an operational software package that implements this new class of methods.

This validation analysis will compare this newly proposed class of schemes to the existing scheme that this new class was derived from. Independent simulations will be run for these methods, and compared against the findings of this new method's author. This research proposal outlines a plan achieve this goal and is broken down into three main objectives: (1) Independently develop software for the newly proposed and existing methods, (2) Conduct an error analysis on the methods' order of accuracy, and (3) Conduct a computational cost analysis of the two methods, with all three objectives using known fractional differential equations and their solutions. The results of this study could not only increase the validity of a new, more accurate and computationally efficient method, but also deliver a software package that can be used to increase fidelity in future research applications.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a thesis committee consisting of Dr. Alan Freed and Dr. Anastasia Muliana of the Department of Mechanical Engineering and Dr. Raytcho Lazarov of the Department of Mathematics. The data and mathematical models analyzed for this thesis were provided by Dr. Freed. All other work conducted for the thesis was completed by the student independently.

Funding Sources

All work up to this point in time for this thesis was independently funded without outside financial support.

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION AND LITERATURE REVIEW

1.1 Motivation

The primary motivation for this study is rooted in the simulation of mechanics of material deformation. The behavior and response of certain classes of materials, especially in the classes of viscoelasticity and viscoplasticity have been exceptionally difficult to accurately model using ordinary differential equations. One strategy to increase the fidelity and accuracy of these difficult-to-model behaviors is to use a heredity term, or a so-called memory effect of the material. In other words, the prediction of the state of the material at the current time in question is influenced by its state at all previous times. One effective approach to include this heredity is to model the material behavior using fractional-order differential equations (FDE's), which will be discussed in further detail later.

However, the inclusion of this material heredity mechanism in the model comes with a greatly increased computational cost. Many previous studies into specific applications using FDE's have failed due to computational expense issues. Therefore, a new method that decreases the computational expense of the memory effect would be a very useful advancement.

1.2 Fractional-Order Differential Equations

In order to discuss these new methods, let us first give an overview of what is meant by a fractional-order differential equation. An abbreviated explanation of some concepts used in this research are recalled below. For a more detailed and complete background into fractional calculus, see [1]. The following discussion and derivation will follow closely to the introduction of [2], as the methods proposed in this article are the primary focus of the research. The fractional-derivative

of a function is defined below as

$$D_0^\alpha y(t) = f(t, y(t)), \quad t = [0, T] \tag{1.1}$$

$$y^{(k)}(0) = y_k, \quad k = 0, 1, ..., m-1, \tag{1.2}$$

where $m - 1 < \alpha \leq m \in Z^+$. The fractional derivative expressed above may be defined in several different forms; however the form used moving forward will be the Caputo Derivative [3], which is defined below as

$$D_0^\alpha y(t) = \frac{1}{\Gamma(m-\alpha)} \int_0^t (t-\tau)^{m-1-\alpha} y^m(\tau) d\tau \tag{1.3}$$

This choice is made due to the convenience of defining the initial conditions easily with the Caputo derivative. The initial conditions of the Caputo derivative are expressed in terms of integer-order derivatives, which are much easier to define, and have a physical significance. The Riemann-Liouville derivative, in contrast, is much more useful for applications in the study of wave-based phenomena. This is due to the initial conditions of the wave not being of particular interest, but rather the wave at some point along its path. The initial conditions of the Riemann-Liouville derivative are defined in terms of initial values of fractional derivatives, as opposed to the integer-order values as mentioned with the Caputo derivative. A more detailed description of these different forms can be found in [4].

A continuous function $y(t)$ is the solution of Equation 1.1 if and only if it is the solution to the Volterra integral equation [5]

$$y(t) = g(t) + \frac{1}{\Gamma(\alpha)} \int_0^t (t-\tau)^{\alpha-1} f(\tau, y(\tau)) d\tau, \tag{1.4}$$

where the initial condition, $g(t)$ is defined as

$$g(t) = \sum_{k=0}^{m} y_k \frac{t^k}{k!}. \tag{1.5}$$

A discretized grid over which the solution is sought is constructed as

$$\Phi_N := \{t_j : 0 = t_0 < t_1, ... < t_j < ... < t_n < t_{n+1} < t_N = T\}. \tag{1.6}$$

Assuming this grid is uniform, i.e. $h = t_{j+1} - t_j, \forall j = 0, ..., N-1$, the solution can be expressed as a set of three terms, as seen below by rewriting Equation 1.4 [6],

$$y(t_{n+1}) = g(t_{n+1}) + y^*(t_{n+1}) + Y(t_{n+1}), \tag{1.7}$$

where $y^*(t_{n+1}) = y^*_{n+1}$ is defined as the lag term, and $Y(t_{n+1}) = Y_{n+1}$ is defined as the increment term. The lag term is the source of the heredity, or memory effect, in fractional-order differential equations. This lag term,

$$y^*(t_{n+1}) = \frac{1}{\Gamma(\alpha)} \int_0^t (t_{n+1} - \tau)^{\alpha-1} f(\tau, y(\tau)) d\tau \tag{1.8}$$

requires information from all prior time steps from 0 to the current time t_n in order to calculate the $y(t_{n+1})$ term. The increment term, defined as

$$Y(t_{n+1}) = \frac{1}{\Gamma(\alpha)} \int_{t_n}^{t_{n+1}} (t_{n+1} - \tau)^{\alpha-1} f(\tau, y(\tau)) d\tau, \tag{1.9}$$

is responsible for advancing solution to the next increment forward, as the name would imply.

All of the methods being analyzed in this study involve the same basic "Predict, Evaluate, Correct, Evaluate" (PECE) operating principle. That is, an estimated value y^P_{n+1} is predicted, and used to calculate the increment term to progress. This is followed by calculation of the corrector term, which is the approximation of the solution, using the notation $\tilde{y}_{n+1} \approx y_{n+1}$.

1.2.1 Adams-Bashforth-Moulton (ABM) Method

The Adams-Bashforth-Moulton Method [7] is re-derived below in order to highlight the components of the method, and to aid in setting up a comparison between the methods presented in this study. This method is the existing standard that will be used as a comparison and benchmark for the new schemes proposed in [2]. Equation 1.4 can be rewritten as

$$\begin{aligned} y(t_{n+1}) = g(t_{n+1}) + \frac{1}{\Gamma(\alpha)} \sum_{j=0}^{n-1} \int_{t_j}^{t_{j+1}} (t_{n+1} - \tau)^{\alpha-1} f(\tau, y(\tau)) \\ + \frac{1}{\Gamma(\alpha)} \int_{t_n}^{t_{n+1}} (t_{n+1} - \tau)^{\alpha-1} f(\tau, y(\tau)) d\tau. \end{aligned} \tag{1.10}$$

First, the predictor value, $\tilde{y}_{n+1}^P$ must be developed. A low-order approximation is used, obtained using a constant interpolation over each interval, $I_j = [t_j, t_{j+1}]$,

$$f(t) \approx f(t_j)\chi_{I_j}(t), \tag{1.11}$$

where

$$\chi_{I_j}(t) = \begin{cases} 1, & \text{if } t \in I_j, \\ 0, & \text{otherwise.} \end{cases} \tag{1.12}$$

Combining Equation 1.11 with Equation 1.10, the predictor equation is constructed:

$$\tilde{y}_{n+1}^P = g_{n+1} + \frac{1}{\Gamma(\alpha)} \sum_{j=0}^{n} C_{n+1}^j \tilde{f}_j, \tag{1.13}$$

where the coefficient is defined as

$$C_{n+1}^j = \int_{t_j}^{t_{j+1}} (t_{n+1} - \tau)^{\alpha-1} d\tau = \frac{h^\alpha}{\alpha}[(n+1-j)^\alpha - (n-j)^\alpha]. \tag{1.14}$$

Next, the corrector, $\tilde{y}_{n+1}$ must be developed. Using a linear Lagrange interpolation function, $f(t)$ is interpolated on each interval, $t \in I_j$ such that

$$f(t) \approx f_j L_j(t) + f_{j+1} L_{j+1}(t), \quad t \in I_j, \tag{1.15}$$

where

$$L_k(t) = \frac{t - t_l}{t_k - t_l}, \ (k, l) = (j, j+1), (j+1, j). \tag{1.16}$$

Combining Equation 1.10 and Equation 1.15, the equation for the corrector can be rewritten as

$$\tilde{y}_{n+1} = g_{n+1} + \frac{1}{\Gamma(\alpha)} \sum_{j=0}^{n-1} [B_{n+1}^{0,j} \tilde{f}_j + B_{n+1}^{1,j} \tilde{f}_{j+1}] + \frac{1}{\Gamma(\alpha)} [B_{n+1}^{0,n} \tilde{f}_n + B_{n+1}^{1,n} \tilde{f}_{n+1}^P], \tag{1.17}$$

with the coefficients, which can be evaluated explicitly, defined as

$$\begin{aligned} B_{n+1}^{0,j} &= \int_{t_j}^{t_{j+1}} (t_{n+1} - \tau)^{\alpha-1} L_j(\tau) d\tau, \\ B_{n+1}^{1,j} &= \int_{t_j}^{t_{j+1}} (t_{n+1} - \tau)^{\alpha-1} L_{j+1}(\tau) d\tau. \end{aligned} \tag{1.18}$$

1.2.2 New Proposed Method by Nguyen and Jang (MF-PCL)

The following development of the new method proposed by Nguyen and Jang in [2] will display only the second order scheme with a linear interpolator, in order to mirror the ABM method more closely. The third order scheme with quadratic interpolation is very similar in structure and derivation, and can be found in [2].

The construction of the newly proposed method begins with the following lemma.

Lemma *Assume $\phi(t) \in P_1[0, T]$ where P_1 is the space of all polynomials of degree less than or equal to one. Let $\phi_n, n = 0, ...N$ be the restricted value of $\phi(t)$ on grid Φ_N defined in Equation*

1.6. Then, there exist coefficients b^0_{n+1} and b^1_{n+1} such that the following equality

$$\int_{t_n}^{t_{n+1}} (t_{n+1} - \tau)^{\alpha-1} \phi(\tau) d\tau = \frac{h^\alpha}{\alpha(\alpha+1)} (b^0_{n+1} \phi_{n-1} b^1_{n+1} phi_n), \tag{1.19}$$

is exact. More precisely,

$$b^0_{n+1} = -1, \quad b^1_{n+1} = \alpha + 2; \tag{1.20}$$

The proof for this is shown in full detail in [2]. The new scheme is now defined below, where

$$\tilde{y}_{n+1} = g_{n+1} + \tilde{y}^*_{n+1} + \tilde{Y}_{n+1} \tag{1.21}$$

is the corrector formula, consisting of the lag term,

$$\tilde{y}^*_{n+1} = g_{n+1} + \frac{1}{\Gamma(\alpha)} \sum_{j=0}^{n-1} [B^{0,j}_{n+1} \tilde{f}_j + B^{1,j}_{n+1} \tilde{f}_{j+1}], \tag{1.22}$$

the increment term,

$$\tilde{Y}_{n+1} = \frac{1}{\Gamma(\alpha)} [B^{0,n}_{n+1} \tilde{f}_{n-1} + B^{1,n}_{n+1} \tilde{f}^P_{n+1}], \tag{1.23}$$

and the initial condition, g_{n+1}, as defined in Equation 1.5. Likewise, the predictor formula is defined as

$$\tilde{y}^P_{n+1} = g_{n+1} + y^*_{n+1} + \frac{h}{\Gamma(\alpha+2)} [b^0_{n+1} \tilde{f}_{n-1} + [b^1_{n+1} \tilde{f}_n], \tag{1.24}$$

which uses the same lag term as used in the corrector formula, and coefficients b^0_{n+1} and b^1_{n+1} defined in Equation 1.22.

2. RESEARCH OBJECTIVES

To guide this research, the following objectives are proposed, broken down into three major areas of interest. Each of these three objectives will be discussed in further detail regarding methods of testing and expected results.

1. Independently develop software for the proposed method and the existing standard.
2. Conduct an error analysis on the methods' order of accuracy using simulated test cases.
3. Conduct a computational cost analysis of the methods.

2.1 Independent Software Development and Simulation

The first step towards running a meaningful analysis of the methods in question requires creating a functional software tool. In order for this validation to be meaningful, the numerical methods should be independently written and simulated, using only the derived equations present in [2]. This research objective is a prerequisite to the remaining two objectives, as functional numerical methods must be constructed in order to collect meaningful data for error analysis and computational cost comparisons.

The numerical methods under analysis will be written using Matlab (as well as Python if time and resources allow). There also exists a third-party Matlab function [8] for the Adams-Bashforth-Moulton method that will be used for increased fidelity in simulation testing. The completion of this research objective will allow for future research of applications in the area of fractional-order differential equations, with a well-documented, easy to operate user interface.

2.2 Error Analysis

The second research objective, error analysis, will look at the accuracy of each of the numerical methods proposed. This objective will yield important information regarding simulation fidelity,

and will either confirm or disprove the alleged increase in accuracy in the newly proposed method. The focus here will be not necessarily on performance of the methods on a case-by-case basis, but rather on the order of error across several classes of equations at different values of α, in both magnitudes of $0 < \alpha \leq 1$ and $\alpha > 1$.

Jang and Nguyen state that the global error E_G for their proposed methods are on the order of $\mathcal{O}(h^2)$ for the 2^{nd} order scheme with linear interpolation, and $\mathcal{O}(h^3)$ for the 3^{rd} order scheme with quadratic interpolation ([2]). This would be a significant improvement over the Adams-Bashforth-Moulton method for cases where $0 < \alpha < 1$, which has been shown in [9] to be on the order of $\mathcal{O}(h^p)$, where $p = min(1 + \alpha, 2)$.

2.3 Computational Cost Analysis

The third and final major focus of this research will involve computational cost of the numerical methods. While increased accuracy is typically a welcomed advancement, it can oftentimes stymie the adoption of new numerical methods if it comes with an increased computational cost. However, the method presented by Nguyen and Jang claims to lower the computational cost while also decreasing (or maintaining for $\alpha \geq 1$) the order the error.

3. PROPOSED METHODS

The following chapter will discuss a detailed approach to completing each of the previously expressed research objectives.

3.1 Independent Software Development and Simulation

This section will be broken down into two further subsections in order to distinguish the two major components of this objective.

3.1.1 Independent Software Development

The software development portion of this objective will largely require converting mathematical equations into numerical methods in a programming language. As mentioned, Matlab is the primary language being used, with a Python version if time and resources permit.

The software package will consist of a main function file for each numerical method, with all of the supporting functions saved in separate files. The supporting files in the software package will include, but are not limited to:

- Linear Lagrange Interpolation Function
- Quadratic Lagrange Interpolation Function
- Initial Condition Calculator
- 2^{nd} Order Coefficient Calculator $(B\_Coefficents)$
- 3^{rd} Order Coefficient Calculator $(A\_Coefficents)$
- Start-up Algorithm

- Simulation and Usage-Example Test Script

Computation of integral functions will be achieved using Matab's adaptive quadrature function, *integral*. No independent integration function will be developed for this research. The functions will be of the following form:

$$[T, Y] = MF\_PCL(alpha, fdefun, t0, tfinal, y0, h) \tag{3.1}$$

$$[T, Y] = MF\_PCQ(alpha, fdefun, t0, tfinal, y0, h) \tag{3.2}$$

$$[T, Y] = ABM(alpha, fdefun, t0, tfinal, y0, h) \tag{3.3}$$

where Equation 3.1 corresponds to the 2^{nd} order scheme with linear interpolation, and Equation 3.2 corresponds to the 3^{rd} order scheme with quadratic interpolation. Equation 3.3 refers to the existing method being compared against, the Adams-Bashforth-Moulton method.

Function Inputs

- **alpha:** The order of the differential equation. Alpha is any number $\alpha \in R^{+}$
- **fdefun:** The fractional-order differential equation. fdefun should be a function handle, and should be a function of known variables and y and t. The notation for this is @(y,t) followed by the equation.
- **t0:** The initial time. t0 should be a scalar value.
- **tfinal:** The final time. tfinal should be a scalar number > t0.
- **y0:** Initial conditions. y0 is a matrix with rows of length equal to the size of the problem, and columns equal to m, which is defined above in the discussion following Equation 1.1 and Equation 1.2.
- **h:** The step size of the problem. h must be a scalar value > 0. The step size remains constant over the entire simulation.

Function Outputs

- **Y:** Solution to the FDE. Y should be a matrix matching in row size to the y0 input, and column size according to the number of steps taken.
- **T:** Time. T should be a vector of times ranging from the start time, t0, to the final time, tfinal, incremented by the step size, h.

3.1.2 Simulation

Once the software tools have been constructed, simulations will be run using an assortment of fractional differential equations and the results from each of the numerical methods will be compared against each other as well as the FDEs' known solutions. The four test cases that were used in [2] include the following:

Sample Equation 1

$$D_0^\alpha y(t) = \frac{40320}{\Gamma(9-\alpha)}t^{8-\alpha} - 3\frac{\Gamma(5+\frac{\alpha}{2})}{\Gamma(5-\frac{\alpha}{2})}t^{4-\frac{\alpha}{2}} + \frac{9}{4}\Gamma(\alpha+1) + \left(\frac{3}{2}t^{\frac{\alpha}{2}} - t^4\right)^3 - y(t)^{\frac{3}{2}} \tag{3.4}$$

$$y(0) = 0, \quad y'(0) = 0$$

whose exact solution is given by:

$$y(t) = t^8 - 3t^{4+\frac{\alpha}{2}} + \frac{9}{4}t^\alpha$$

Sample Equation 2

$$D_0^\alpha y(t) = \frac{\Gamma(4+\alpha)}{6}t^3 + t^{3+\alpha} - y(t) \tag{3.5}$$

$$y(0) = 0, \quad y'(0) = 0$$

whose exact solution is given by:

$$y(t) = t^{3+\alpha}$$

Sample Equation 3

$$D_0^\alpha y(t) = \frac{\Gamma(5+\alpha)}{24} t^4 + t^{8+2\alpha} - y^2(t) \tag{3.6}$$

$$y(0) = 0, \quad y'(0) = 0$$

whose exact solution is given by:

$$y(t) = t^{4+\alpha}$$

Sample Equation 4

$$D_0^\alpha y(t) = \begin{cases} \frac{2}{\Gamma(3-\alpha)} t^{2-\alpha} - y(t) + t^2 - t, & \alpha > 1, \\ \frac{2}{\Gamma(3-\alpha)} t^{2-\alpha} - \frac{1}{\Gamma(2-\alpha)} t^{1-\alpha} - y(t) + t^2 - t, & \alpha \leq 1, \end{cases} \quad \text{at final time T = 1.} \tag{3.7}$$

$$y(0) = 0, \quad y'(0) = -1$$

whose exact solution is given by:

$$y(t) = t^2 - t$$

These equations will be run using both numerical methods, and compared against their true solutions. These four test cases also have the added benefit for this analysis that they have also been simulated and reported in [2]. This gives four test cases to compare this study against not only the actual solutions, but also against the findings of Nguyen and Jang.

More simulations may also be conducted, including different/nonzero initial conditions, or higher values of alpha.

3.2 Error Analysis

Upon completion of the software development and simulations, error analysis can now be conducted. There are three major questions that this research will attempt to answer with regard to error analysis:

1. How does the pointwise error at the final time, $t_N = T$ compare between the methods?

2. How does the least squares error, E_{L^2}, over the time interval compare between the methods?

3. Has the method proposed by Nguyen and Jang yielded a constant order of error of $\mathcal{O}(h^2)$ for the 2^{nd} order scheme, and $\mathcal{O}(h^3)$ for the 3^{nd} order scheme, regardless of α?

Pointwise Error Calculation

The pointwise error of the simulations will be calculated for the methods in question using the following:

$$E_{pt}(T) = |y_N - \tilde{y}_N| \tag{3.8}$$

Least Squares Error Calculation

The second measure of error used for this analysis will be a least squares error over the full domain of the problem, calculated using the following:

$$E_{L^2} = \left(h \sum_{j=0}^{N} |y_j - \tilde{y}_j|^2 \right)^{\frac{1}{2}} \tag{3.9}$$

Order of Error

One of the hallmark achievements claimed by this new method is a reduction in order of error, specifically when $\alpha < 1$. This is due to the order of the ABM method having an error of order $\mathcal{O}(h^p)$, where $p = min(\alpha + 1, 2)$. If this is indeed true, it is clear that for values of alpha less than one, the order of error would be significantly larger in the ABM case. In order to assess this claim, the above two measures of error, Equation 3.8 and Equation 3.9 will be used to calculate the order of error for each of the methods.

3.3 Computational Cost Analysis

The second hallmark achievement that the method by Nguyen and Jang claims is that they have reduced the computational cost of their schemes by half of the ABM method. Their claim is that this increase in speed is a result of the removal of recalculating the memory effect in the predictor step.

The final objective of this research is to run a computational cost study of the methods in question. This will be achieved by running a number of simulation test cases in which both α and the number of steps/step size are varied. Each trial case will be run multiple times to collect a reasonable sample size and an average computational time will be calculated. The computational time will be collected using Matlab's $cputime$ function to remove human error from time collection measurements.

4. EXPECTED RESULTS

4.1 Simulation

The findings of [2] include two simulations comparing the performance of ABM, MF-PCL, and the true solution, using a grid size of $N = 20, T = 1$ second, and values of $\alpha = 0.25, 1.25$. These simulations use Equation 3.4 and it's associated initial conditions/true solution. These results are shown below in Figure 4.1. It is expected that this research will see identical, or highly similar results to this graph, given the same conditions. Minor variation may result from differences in tolerances and integrators used to compute the terms.

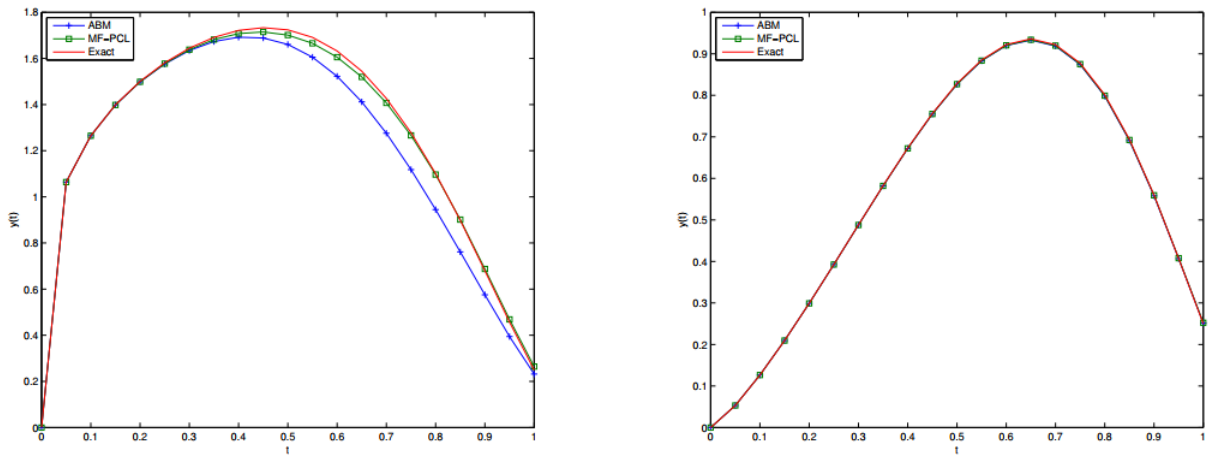

Figure 4.1: Expected simulation results of competing methods. Left: $\alpha = 0.25$, Right: $\alpha = 1.25$.

4.2 Error Analysis

There are also several tables of error calculations reported in [2]. For the sake of brevity, below is one sample table taken from the simulations using Equation 3.4. Note that it is expected to see the same order of accuracy for $0 < \alpha < 1$ as well as $\alpha \geq 1$ for the MF-PCL. Conversely, there should be a clear difference in the order of error between these two cases for α when looking at the ABM method.

	$\alpha = 0.25$							
	ABM				MF-PCL			
N	E_{pt}	Order	E_{L^2}	Order	E_{pt}	Order	E_{L^2}	Order
10	2.5007E-01	-	3.1370E-01	-	1.7439E-01	-	2.1027E-01	-
20	1.8095E-02	3.7886	8.6943E-02	1.8512	1.4567E-02	3.5815	1.4413E-02	3.8668
40	3.6054E-03	2.3274	2.4831E-02	1.8079	2.6392E-03	2.4645	1.6389E-03	3.1366
80	1.4522E-03	1.3119	8.0459E-03	1.6258	5.0674E-04	2.3808	2.4669E-04	2.7319
160	6.5805E-04	1.1420	2.8152E-03	1.5150	9.9526E-05	2.3481	4.1343E-05	2.5770
320	2.9689E-04	1.1483	1.0318E-03	1.4481	2.0182E-05	2.3020	7.4652E-06	2.4694
	$\alpha = 0.5$							
	ABM				MF-PCL			
N	E_{pt}	Order	E_{L^2}	Order	E_{pt}	Order	E_{L^2}	Order
10	1.7859E-02	-	4.9366E-02	-	2.6565E-02	-	1.4118E-02	-
20	1.8123E-03	3.3008	1.3769E-02	1.8421	5.2961E-03	2.3265	2.0946E-03	2.7528
40	4.1619E-04	2.1225	4.1517E-03	1.7296	1.0748E-03	2.3008	3.6047E-04	2.5387
80	1.7655E-04	1.2371	1.3188E-03	1.6544	2.3143E-04	2.2155	7.0952E-05	2.3450
160	7.9795E-05	1.1457	4.3342E-04	1.6054	5.3098E-05	2.1238	1.5567E-05	2.1883
320	3.3898E-05	1.2351	1.4570E-04	1.5728	1.2728E-05	2.0606	3.6510E-06	2.0921
	$\alpha = 1.25$							
	ABM				MF-PCL			
N	E_{pt}	Order	E_{L^2}	Order	E_{pt}	Order	E_{L^2}	Order
10	5.5326E-03	-	8.1359E-03	-	1.0161E-02	-	6.0442E-03	-
20	1.5932E-03	1.7960	1.8821E-03	2.1120	2.3430E-03	2.1166	1.3773E-03	2.1337
40	4.3283E-04	1.8801	4.4311E-04	2.0866	5.7010E-04	2.0391	3.3411E-04	2.0435
80	1.1434E-04	1.9205	1.0555E-04	2.0697	1.4133E-04	2.0122	8.2552E-05	2.0169
160	2.9741E-05	1.9428	2.5353E-05	2.0577	3.5232E-05	2.0041	2.0531E-05	2.0075
320	7.6631E-06	1.9564	6.1289E-06	2.0485	8.7988E-06	2.0015	5.1201E-06	2.0035

Figure 4.2: Expected results from error analysis.

4.3 Computational Cost

Finally, [2] has set a benchmark on computer run-time measurements. While it is not expected that the times themselves match, this does give a benchmark trend between the two methods, in terms of relative speeds. There is also no specification on how these values were obtained, how many samples of each run were recorded, nor if this is an average run-time or the fastest run for each test case. These are areas that this research will answer in more complete and well-defined terms. Figure 4.3 Below is a table of computation time values.

	$\alpha = 0.25$					
N	100	200	400	600	800	1000
ABM	0.002	0.010	0.040	0.090	0.160	0.262
MF-PCL	0.001	0.007	0.025	0.050	0.096	0.150
	$\alpha = 0.5$					
N	100	200	400	600	800	1000
ABM	0.003	0.010	0.040	0.090	0.160	0.250
MF-PCL	0.001	0.006	0.024	0.057	0.096	0.150

Figure 4.3: Expected results from computational cost analysis.

5. RESEARCH PLAN

5.1 Time-line

The time-line below is organized in a work-breakdown table over the course of the degree program, broken down by academic semester.

Objective	**Fall 2016**	**Spring 2017**	**Fall 2017**	**Spring 2018**
Software Development and Independent Simulation	Literature Review and Directed Study	Begin construction of methods in Matlab	Complete construction of methods in Matlab. Run simulations.	Write and defend thesis. Construct Python version if time/resources permit.
Error Analysis	Literature Review and Directed Study		Conduct error analysis on collected simulation data.	Write and defend thesis
Computational Cost Analysis	Literature Review and Directed Study		Run computational cost analysis on simulations.	Write and defend thesis

Table 5.1: Research time-line organized by academic semester.

5.2 Resources and Equipment

This research is contained entirely in numerical methods. No physical equipment, with the exception of a computer, is required to conduct this research. The only other resource required for this research is a Matlab license, which is provided by Texas A&M University at no cost to its students.

REFERENCES

[1] I. Podlubny, *Fractional Differential Equations*. Academic Press, 1999.

[2] T. B. Nugyen and B. Jang, "A high-order predictor-corrector method for solving nonlinear differential equations of fractional order," *Fractional Calculus and Applied Analysis*, vol. 20, No. 2, pp. 447–476, 2017.

[3] M. Caputo, "Linear models of dissipation whose q is almost frquency independent," *The Geophysical Journal of the Royal Astronomical Society*, vol. 13, pp. 529–539, 1967.

[4] M. Rahimy, "Applications of fractional differential equations," *Applied Mathematical Sciences*, vol. 4, pp. 2453–2461, 2010.

[5] K. Diethelm, *The Analysis of Fractional Differential Equations*. Springer, 2010.

[6] D. Conte and I. D. Prete, "Fast collocation methods for volterra integral equations of convolution type," *Journal of Computational and Applied Mathematics*, vol. 196, pp. 652–663, 2006.

[7] K. Diethelm, N. J. Ford, and A. D. Freed, "A predictor-corrector approach for the numerical solution of fractional differential equations," *Nonlinear Dynamics*, vol. 29, pp. 3–22, 2002.

[8] R. Garrappa, "Predictor-corrector pece method for fractional differential equations." `https://www.mathworks.com/matlabcentral/fileexchange/32918-predictor-corrector-pece-method-for-fractional-differential-equations`, 2012.

[9] K. Diethelm, N. J. Ford, and A. D. Freed, "Detailed error analysis for a fractional adams method," *Numerical Algorithms*, vol. 36, pp. 31–52, 2004.