# Numerical Methods for Initial Value Problems

Kian Greene, Cody Kohn, Joseph Sauerbrun, and M. Catherine Yopp

14 February 2022

## Abstract

Numerical ODE Integrators can be used to approximate the initial value problems (IVPs) that model physical situations. In particular, this report discusses Euler schemes, Taylor series expansion, and Runge-Kutta (RK) methods. Chapter 1 investigates the different rates of convergence of these methods to an exact solution, with the RK4 scheme having the fastest rate of convergence. Chapter 2 explores epidemic modeling, which employs an RK4 scheme to examine how an epidemic will spread according to various factors. Chapter 3 examines the stem curve of a palm tree in strong winds, and again uses an RK4 method to simulate how it will curve under a given wind speed. Ultimately, even though there are many different methods that can be used to approximate IVPs, the RK4 scheme especially is a very efficient numerical integrator that can be used to model many different IVPs with excellent accuracy because of its simplicity.

# Chapter 1

# Rate of Convergence of ODE Integrators

## 1.1 Introduction

This chapter of the report will investigate the rate of convergence of different ODE Integrator schemes, including various Euler, Runge-Kutta (RK), and Taylor series methods. The particular schemes examined were:

- Explicit Euler
- Improved Euler
- Optimal RK2
- Third-Order Taylor Series
- Classical RK4

The initial value problem (IVP) being examined with these methods is

$$\frac{dy}{dt} = 7t^2 - \frac{4y}{t} \quad (-1 \leq t \leq 6) \tag{1.1}$$

$$y(1) = 2 \tag{1.2}$$

$$y = t^3 + \frac{1}{t^4} \tag{1.3}$$

where it is first investigated with the initial condition of (1.2), which has the exact solution of (1.3), and then examined with the initial condition of $y(1) = 1$.

After the ODE Integrator schemes were implemented, they were compared to the exact solution (1.3) to determine how well they approximated the true values.

## 1.2 Math Modeling and Analysis/Methods

### 1.2.1 Explicit Euler

The first method used to investigate the IVP was the Explicit Euler scheme, which has the form:

$$Y_{n+1} = Y_n + hf(t_n, Y_n), \quad n = 0, 1, 2, ..., N-1 \tag{1.4}$$

where $N$ is the total number of data points and $h = \frac{t_{max} - t_0}{N}$.

The advantage of using the Explicit Euler method is that is is relatively quick and easy to do. However, it is usually not the most accurate approximation of the exact function, especially compared to the other methods applied in this IVP.

### 1.2.2 Improved Euler

The second scheme used to investigate the IVP was the Improved Euler scheme, which has the form:

$$Y_{n+1} = Y_n + \frac{h}{2}[f(t_n, Y_n) + f(t_{n+1}, Y_n + hf(t_n, Y_n))], \quad n = 0, 1, 2, ..., N-1 \tag{1.5}$$

The advantage of using the Improved Euler method is that, as the name suggests, it is an improvement to the Explicit Euler method based around trapezoidal Riemann sums. However, the calculations involved for each step can get rather complicated as the number of terms increases.

### 1.2.3 Optimal RK2

The third method used to investigate the IVP was the Optimal RK2 scheme, which has the form:

$$Y_{n+1} = Y_n + \frac{h}{2}(K_1 + K_2), \quad n = 0, 1, 2, ..., N-1 \tag{1.6}$$

$$K_1 = f(t_n, Y_n) \tag{1.7}$$

$$K_2 = f(t_n + h, Y_n + hK_1) \tag{1.8}$$

The advantage of using the RK2 scheme is that it imitates a second-order Taylor series method but does not require the computation of derivatives of $f(t, y)$. However, it is less accurate than higher-order Taylor Series and RK methods.

### 1.2.4 Third-Order Taylor Series

The fourth method used to investigate the IVP was the third-order Taylor series scheme, which has the form:

$$Y_{n+1} = Y_n + h[f(t_n, Y_n) + \frac{h}{2!}\frac{df}{dt}(t_n, Y_n) + \frac{h^2}{3!}\frac{d^2 f}{dt^2}(t_n, Y_n)], \quad n = 0, 1, 2, ..., N-1 \tag{1.9}$$

The advantage of using the third-order Taylor series scheme is that Taylor series are self-starting, given an initial value $Y_0$. A disadvantage, though, is that derivatives of $f(t, y)$ would have to be calculated in advance to the desired order.

### 1.2.5 Classical RK4

The fifth method used to investigate the IVP was the Classical RK4 scheme, which has the form:

$$Y_{n+1} = Y_n + \frac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4), \quad n = 0, 1, 2, ..., N-1 \tag{1.10}$$

$$K_1 = f(t_n, Y_n) \tag{1.11}$$

$$K_2 = f(t_n + \frac{1}{2}h, Y_n + \frac{1}{2}hK_1) \tag{1.12}$$

$$K_3 = f(t_n + \frac{1}{2}h, Y_n + \frac{1}{2}hK_2) \tag{1.13}$$

$$K_4 = f(t_n + h, Y_n + hK_3) \tag{1.14}$$

The advantage of using the RK4 scheme is that it is a relatively fast, accurate approximation that does not require computing the derivatives of $f(t, y)$. The main disadvantage is calculating the $K$'s instead, which depends on $f(t, y)$ itself.

## 1.3 Problem Solving Procedure/Solution

The analysis of the first problem can provide realization of each approximation method that is used and how each method compares to the others. The set up and procedure of this problem is straight forward as it only requires knowledge of how to calculate each method and how to write the code for each method individually that will later be graphed and compared once all the code is written and executed.

By creating functions in the Matlab code for each method of approximation and an additional one to find the exact solution it is easy to calculate the numerical values needed to evaluate each method. After the functions are properly working a graph is needed to plot each method along the time period to show the progression of each method over time and the deviance of each from the exact. As shown by the graph below it can be seen that each approximation method generally holds to the curve of the known exact function that was given.

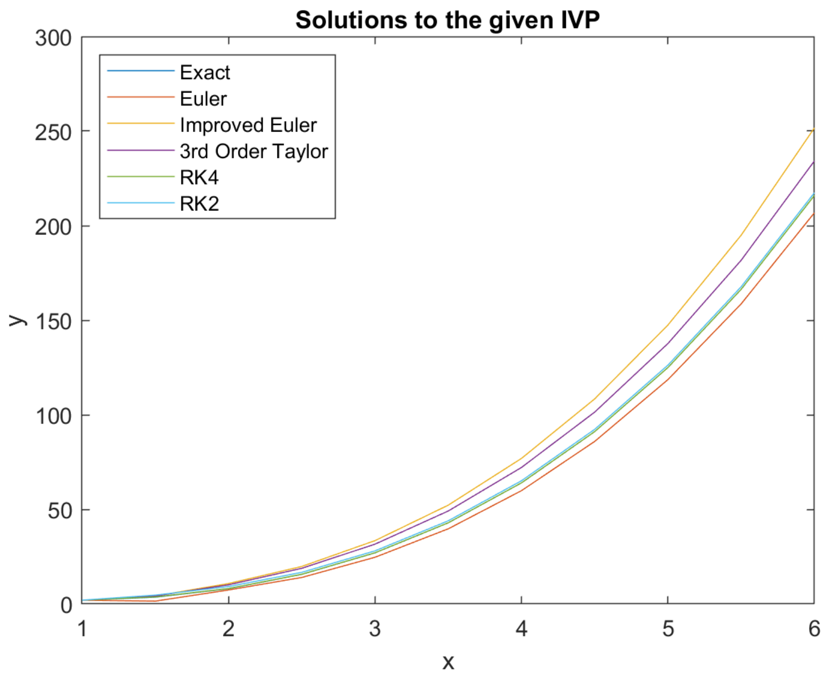

Figure 1.1: Comparing ODE Integrators to Exact Solution

However, it can be noted that while all the plots start out closely bunched as $t$ starts, as time progresses most approximation methods diverge from the exact and the error increases. Please note that for most of the graph the RK4 and exact functions nearly overlap and cover one another. This shows the clear solution that the RK4 method is the closest and most successful approximation method along the entire range of times in this question. This can be further and more accurately seen when viewing the percent error table that was calculated alongside the graph in Matlab.

Table 1.1: Percent Errors of Each Integrator Method

| n | Euler's | Improved | Taylor's | RK4 | RK2 |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 58.01295896 | 18.38012959 | 17.79697624 | 3.264794816 | 30.62634989 |
| 2 | 8.527131783 | 33.8501292 | 25.47293202 | 0.83870344 | 13.89867669 |
| 3 | 10.54656052 | 26.63252527 | 20.4135512 | 0.28776048 | 6.776572415 |
| 4 | 8.652879342 | 23.93407221 | 17.04786059 | 0.123818423 | 3.883517228 |
| 5 | 7.361337233 | 21.81488604 | 14.59943059 | 0.062006793 | 2.516548952 |
| 6 | 6.411962161 | 20.22152666 | 12.75461479 | 0.034513845 | 1.772978165 |
| 7 | 5.681536461 | 18.9870264 | 11.31953366 | 0.020755126 | 1.322866108 |
| 8 | 5.101214704 | 18.00395923 | 10.172916 | 0.013239199 | 1.028249258 |

In Table 1.1 it is clear to notice that the percent errors for the RK4 method are the smallest at all time intervals: it starts with a percent error of just over 3 percent and ends with a percent error of thirteen-thousandths of a percent. The RK4 method is in fact so much more accurate and precise than other methods that it is closer to the exact value at the second step of the approximation than all the other methods are at the last step of the approximation. At the first time step, the ranking of methods from best to worst goes RK4, Third Order Taylor's, Improved Euler's, RK2, Euler's; at the final time step, though, the ranking of methods goes RK4, RK2, Euler's, Third Order Taylor's, Improved Euler's. Clearly, as time goes on, the RK methods get greatly more accurate at approximation; Taylor's and Euler's methods also become more accurate but not to the same degree as the RK methods. Moreover, the Improved Euler's method percent error decreases by only two percent between the first and last error calculations as it tends to overshoot the exact value.

## 1.4 Discussion/Conclusions

As it can be seen above for both $t$ values very close to the $t_0$ and as $t$ increases the RK4 method stands above the rest by a wide margin. The other methods do generally approximate the exact solution with the largest percent error at the final time step being roughly seventeen percent with the Improved Euler's method. This claim holds for both initial conditions too. When changing the initial condition from 2 to 1 the approximations will obviously be off for the first step as the calculation of the exact does not depend on the initial condition so it will remain at where it was prior, but the results for error are generally the same and provide the same results for best to worst methods as stated above. However, the RK4 method from this problem is shown to be the best method for approximation in both short term approximation and long term approximation. An interesting continuation of this problem may be to continue $t$ to see at what point the other approximation methods may converge as closely as the RK4 method does or if the math behind them simply does not hold up.

Chapter 2

# Modeling the Spread of an Epidemic

## 2.1 Introduction

The modeling of an epidemic is a powerful and useful tool for governments and medical industries understand how many people could eventually die from a disease and how long it will last. This is why W. O. Kermack and A. G. McKendrick theorized a model to demonstrate how a group of people are affected by the disease based on pre-determined transmission and mortality rates.

The Kermack-McKendrick model is most commonly represented in a triple ordinary differential equation (ODE) system (Equations 2.1-2.3)

$$\frac{dH}{dt} = -cHI \tag{2.1}$$

$$\frac{dI}{dt} = cHI - mI \tag{2.2}$$

$$\frac{dD}{dt} = mI \tag{2.3}$$

where $H$ is the number of healthy villagers, $I$ is the number of infected villagers, $N$ is the number of villagers in the group of interest, $c$ is the weekly transmission rate of the disease, and m is the weekly mortality rate of infected individuals. The model can also be represented as a system composed of a single ODE, an exponential equation, and a linear equation (Equations 2.4-2.6)

$$\frac{dD}{dt} = m(N - D - H) \tag{2.4}$$

$$H = H_0 e^{\frac{-cD}{m}} \tag{2.5}$$

$$I = N - H - D \tag{2.6}$$

This paper will analyze a village of 3500 villagers with $I_0 = 150$, $c = .001$, and $m = 1.8$ using both representations of the Kermack-McKendrick model to understand how long an epidemic can affect the health of individuals in an isolated environment.

## 2.2 Math Modeling and Analysis Methods

The method used to analyze the Kermack-McKendrick model is the fourth stage explicit Runge-Kutta (RK4) method. The RK4 method analyzes the slope of a function at four different points within a time step, $h$; once at the beginning, twice at the midpoint, and once at the endpoint. This is preferred over the Explicit Euler method—that only analyzes the slope at the beginning of the time step—and the Improved Explicit Euler method that only analyzes the slope at the beginning and midpoint of the time step.

Another effective method of approximation is the Taylor Series scheme due to its ability to be self-starting. The drawback of this scheme relative to the problem at hand, is that each equation of the system relies on the solution to another equation in the system. This proves difficult when computing the derivatives necessary to complete an nth-order Taylor series.

One of the most effective numerical analysis methods is RK45, which is a combination of the RK4 and RK5 methods. This method is often used in place of the RK4 method when a function changes drastically over a given interval since the RK4 method does not account for truncation error at every stage. The RK4 method was chosen over the RK45 method because none of the differential equations in the system explicitly include the variable of time so their solutions should not drastically change with respect to time. The RK4 method is also less computationally expensive.

A drawback of the RK4 method is its computational cost which is increased by decreasing values of the time steps, $h$. But smaller time steps increase the instances of slope approximation and as h approaches zero, the solution is close enough to the exact solution to a negligible degree of error. This means it is desired to find the smallest time step such that the error of approximation between solutions of different time steps is approximately zero. For this computation, it was found that 1000 steps is the threshold at with the approximated solutions doesn't change in comparison to higher step computations.

To implement this scheme, the following equations for every step are used until the end of the specified time variable

$$
\begin{gathered}
\mathrm{K}_1 = f(t, Y) \\
\mathrm{K}_2 = f(t + \tfrac{1}{2}h, Y + \tfrac{1}{2}hK_1) \\
\mathrm{K}_3 = f(t + \tfrac{1}{2}h, Y + \tfrac{1}{2}hK_2) \\
\mathrm{K}_4 = f(t + h, Y + hK_3)
\end{gathered}
$$

Figure 2.1: The slope evaluation of $K_n$ to be implemented in the RK4 method.

where the function $f(t, Y)$ indicates the right-hand side of ODE and each $K$ is an instance of slope evaluation. Each differential equation is analyzed at each step and yields an array of solutions approximated by

$$Y_{n+1} = Y_n + \frac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4) \tag{2.7}$$

where $Y_{n+1}$ is the approximated value of the solution at $t_0 + h$.

## 2.3 Problem Solving Procedure and Solution

Since the triple ODE system contains coupled equations, it is better to create a vector $\vec{x}$ with its components being Equations 2.1-2.3

$$
\begin{gathered}
{x_1}' = -cx_1x_2 \\
{x_2}' = cx_1x_2 - mx_2 \\
{x_3}' = mx_2
\end{gathered}
$$

while the single ODE system can be represented as is. As shown in Figures 2.2 and 2.3, both systems output identical information.

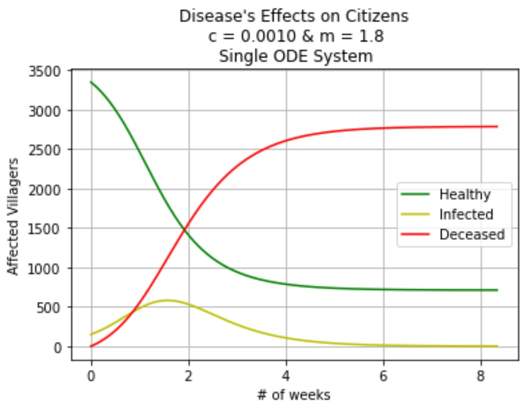

Figure 2.2: The solution to the single ODE system.

For the given initial conditions of this case study, it would take approximately 8.4 weeks for the epidemic to end, leaving approximately 2787 deceased and 713

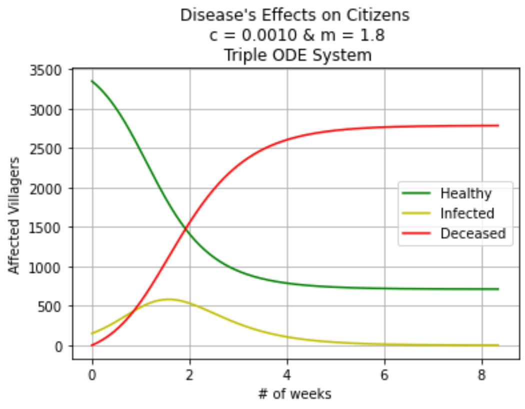

Figure 2.3: The solution to the triple ODE system.

alive and well (Appendix A).

For models such as these—that represent isolated and coupled systems—there are only two necessary stopping criteria: the approximation error being less than a predetermined tolerance or the value of a solution(s) reaching a certain predetermined value. This is because the sum of these kinds of systems' solutions will converge to the sum of the initial conditions. This is shown in the Table 2.1, Table 2.2 and Appendix A; the sum of the healthy, deceased, and infected villagers equals 3500 which is the total number of villagers. For this case, the latter type of criterion was chosen since, ideally, there are zero infected people at the end of an epidemic.

## 2.4 Discussion/Conclusions

Observing the data at different values of $m$ (Table 2.1), it can be concluded that the higher the weekly mortality rate of the infected, the shorter the lifespan of the epidemic and the less individuals die. At values higher than $m = 2.7$, there are more healthy villagers than deceased after the epidemic. At values less than $m = 0.8$, almost all citizens are deceased at the end of epidemic. The data also shows that the lower the transmission rate, the shorter the epidemic's lifespan and the more individuals live (Table 2.2). For values lower than $c = .0007$, there are more healthy individuals than deceased at the epidemic's end. But, for values greater than $c = .002$, almost all citizens are deceased at the end. This data supports the hypothesis that preventative measures can save lives and reduce the length of an epidemic. It also shows that the quicker the disease deteriorates a person's health, the faster the epidemic ends.

Table 2.1: The Effects of Varying Mortality Rates (m)

| m | Healthy Remaining | Deceased | Epidemic Lifespan (weeks) |
|---|---|---|---|
| 0.7 | 24 | 3476 | 13.3 |
| 0.8 | 46 | 3454 | 12.1 |
| 0.9 | 76 | 3424 | 11.2 |
| 1.0 | 114 | 3386 | 10.5 |
| 1.1 | 162 | 3338 | 9.9 |
| 1.2 | 218 | 3282 | 9.5 |
| 1.3 | 283 | 3217 | 9.2 |
| 1.4 | 355 | 3145 | 8.9 |
| 1.5 | 435 | 3065 | 8.7 |
| 1.6 | 522 | 2978 | 8.6 |
| 1.7 | 615 | 2885 | 8.5 |
| 1.8 | 713 | 2787 | 8.4 |
| 1.9 | 817 | 2683 | 8.3 |
| 2.0 | 926 | 2574 | 8.3 |
| 2.1 | 1039 | 2461 | 8.2 |
| 2.2 | 1155 | 2345 | 8.2 |
| 2.3 | 1273 | 2227 | 8.2 |
| 2.4 | 1394 | 2106 | 8.2 |
| 2.5 | 1516 | 1984 | 8.2 |
| 2.6 | 1638 | 1862 | 8.1 |
| 2.7 | 1759 | 1741 | 8.1 |
| 2.8 | 1878 | 1622 | 8.0 |

Table 2.2: The Effects of Varying Transmission Rates (c)

| c | Healthy Remaining | Deceased | Epidemic Lifespan (weeks) |
|---|---|---|---|
| 0.0001 | 3317 | 183 | 3.4 |
| 0.0002 | 3264 | 235 | 4.4 |
| 0.0003 | 3174 | 326 | 6.0 |
| 0.0004 | 2996 | 504 | 8.7 |
| 0.0005 | 2637 | 863 | 12.2 |
| 0.0006 | 2106 | 1394 | 13.0 |
| 0.0007 | 1603 | 1897 | 11.7 |
| 0.0008 | 1214 | 2286 | 10.3 |
| 0.0009 | 926 | 2574 | 9.2 |
| 0.0010 | 713 | 2787 | 8.4 |
| 0.0011 | 555 | 2945 | 7.7 |
| 0.0012 | 435 | 3065 | 7.3 |
| 0.0013 | 344 | 3156 | 6.9 |
| 0.0014 | 273 | 3227 | 6.6 |
| 0.0015 | 218 | 3282 | 6.3 |
| 0.0016 | 175 | 3325 | 6.1 |
| 0.0017 | 141 | 3359 | 6.0 |
| 0.0018 | 114 | 3386 | 5.8 |
| 0.0019 | 93 | 3407 | 5.7 |
| 0.0020 | 76 | 3424 | 5.6 |
| 0.0021 | 62 | 3438 | 5.5 |
| 0.0022 | 50 | 3450 | 5.4 |

# Chapter 3

# Stem Curve of a Tall Palm in a Strong Wind

## 3.1 Introduction

In 1993, D. Winter modeled the stem curve of a wind-loaded Mexican Fan palm in steady Santa Ana winds with the following equations

$$\frac{d^2\theta}{ds^2} = \frac{W_s}{EI}(1 - \frac{s}{L} + \frac{W_c}{W_s})sin\theta + \frac{D}{EI}cos\theta \tag{3.1}$$

$$\frac{dx}{ds} = sin\theta \tag{3.2}$$

$$\frac{dz}{ds} = cos\theta \tag{3.3}$$

where $\theta$ is the angle of the stem relative to the vertical position, $s$ is the arc length measured along the stem, $x$ is the horizontal displacement of the stem, and $z$ is the height of a location along the stem. $x$ and $z$ are treated as functions of $s$ [1]. The parameters are the total stem weight $W_s = 22700N$, the Young's modulus of the stem $E = 6 \times 10^9 N/m^2$, the moment of inertia of the stem $I = 5.147 \times 10^-4m^4$, the length of the stem $L = 30m$, the total canopy weight $W_c = 1385.5N$ and the wind drag force on the canopy $D = 4.135U^2N$, where $U$ is the wind speed in $m/s$.

This model was explored through a simulation of the stem curve of a palm tree with the initial conditions $\theta(0) = \theta'(0) = x(0) = z(0) = 0$ and a wind speed of $18m/s$.

## 3.2 Math Modeling and Analysis/Methods

The fourth-order Runge-Kutta (RK4) method was used to analyze Winter's model because of the relatively complicated equation for $\frac{d^2\theta}{ds^2}$. Although a higher-order Taylor Series may have worked, calculating the derivatives of $f(t, y)$ would likely get complicated; therefore, the similar RK method was employed, which mimics a Taylor series but without the need to compute the derivatives of $f(t, y)$.

Specifically, the RK4 method was used, which has the form:

$$Y_{n+1} = Y_n + \frac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4), \quad n = 0, 1, 2, ..., N-1 \tag{3.4}$$

$$K_1 = f(t_n, Y_n) \tag{3.5}$$

$$K_2 = f(t_n + \frac{1}{2}h, Y_n + \frac{1}{2}hK_1) \tag{3.6}$$

$$K_3 = f(t_n + \frac{1}{2}h, Y_n + \frac{1}{2}hK_2) \tag{3.7}$$

$$K_4 = f(t_n + h, Y_n + hK_3) \tag{3.8}$$

Although the RK4 scheme does not require calculating the derivatives of $f(t, y)$, the $K$'s have to be computed instead, which depend on the complexity of $f(t, y)$ itself.

## 3.3 Problem Solving Procedure/Solution

The RK4 method was used to solve the system of equations given by (3.1), (3.2), and (3.3) for $\theta$, $x$, and $z$ across a range of $N = 21$ points, where the arc length $s$ was set from 0 to 30 meters. A wind speed of 18 $m/s$ was used for the simulations.

Figure 3.1:

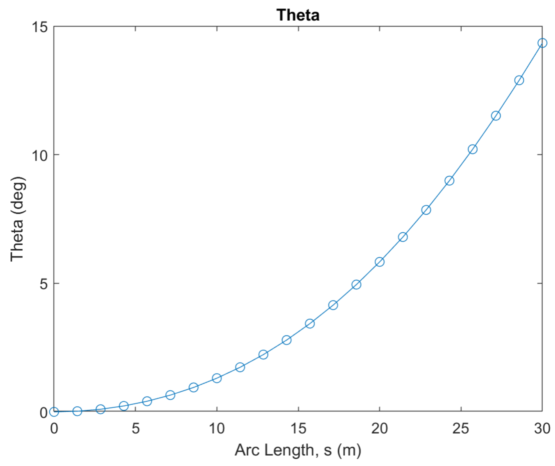

Figure 3.1 shows the angle of the stem relative to the vertical position, $\theta$ in degrees, as a function of the arc length measured along the stem, $s$. As the arc length increases, the angle of the stem relative to the vertical position increases non-linearly.

Figure 3.2:

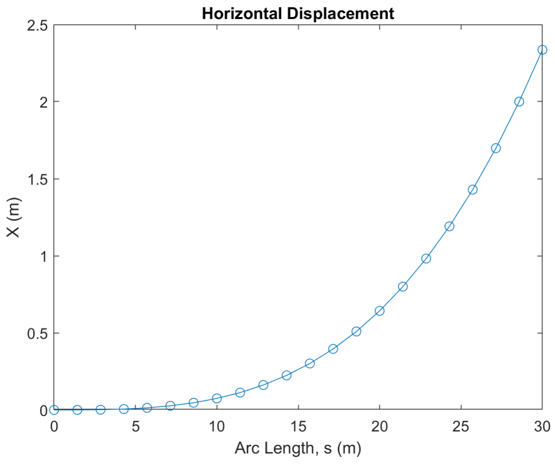

Figure 3.2 shows the horizontal displacement of the stem, $x$ in meters, as a function of the arc length measured along the stem, $s$. As the arc length increases, so does the horizontal displacement of the stem in a non-linear fashion.

Figure 3.3:

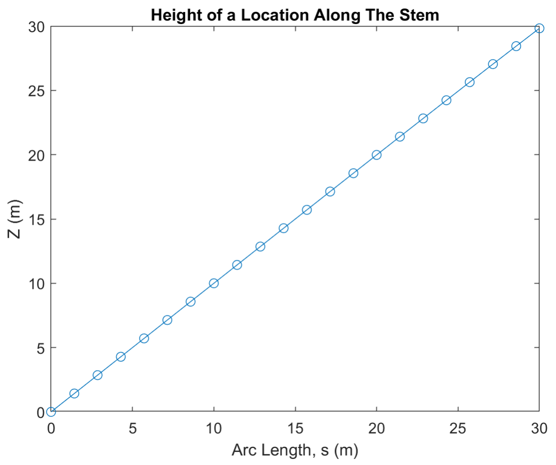

Figure 3.3 shows and height of a location along the stem, $Z$ in meters, as a function of the arc length measured along the stem, $s$. Clearly, the plot displays a linear relationship between the two variables.

## 3.4 Discussion/Conclusions

Solving the coupled set of differential equations provides the angle of the stem relative to the vertical position, the horizontal displacement of the stem, and the height of a location along the stem. These equations were solved using the Runge-Kutta method and plotted as functions of arc length measured along the stem.

# Appendix A

## Table of Affected Villagers Under Initial Conditions

| Week | Health | Infected | Deceased |
|---|---|---|---|
| 0 | 3350 | 150 | 0 |
| 0.1 | 3329 | 160 | 11 |
| 0.2 | 3272 | 185 | 42 |
| 0.3 | 3208 | 214 | 78 |
| 0.4 | 3135 | 246 | 119 |
| 0.5 | 3054 | 280 | 167 |
| 0.6 | 2964 | 316 | 220 |
| 0.7 | 2867 | 353 | 280 |
| 0.8 | 2762 | 390 | 347 |
| 0.9 | 2652 | 428 | 421 |
| 1 | 2536 | 463 | 501 |
| 1.1 | 2417 | 495 | 587 |
| 1.2 | 2297 | 524 | 679 |
| 1.3 | 2177 | 547 | 775 |
| 1.4 | 2060 | 565 | 876 |
| 1.5 | 1945 | 576 | 978 |
| 1.6 | 1836 | 582 | 1083 |
| 1.7 | 1732 | 581 | 1187 |
| 1.8 | 1635 | 574 | 1291 |
| 1.9 | 1544 | 562 | 1394 |
| 2 | 1461 | 545 | 1493 |
| 2.1 | 1385 | 525 | 1590 |
| 2.2 | 1316 | 502 | 1682 |
| 2.3 | 1253 | 477 | 1770 |
| 2.4 | 1196 | 450 | 1854 |
| 2.5 | 1145 | 423 | 1932 |
| 2.6 | 1099 | 395 | 2006 |
| 2.7 | 1058 | 367 | 2075 |

| Week | Health | Infected | Deceased |
|---|---|---|---|
| 2.8 | 1021 | 340 | 2138 |
| 2.9 | 988 | 314 | 2197 |
| 3 | 959 | 289 | 2252 |
| 3.1 | 933 | 266 | 2301 |
| 3.2 | 909 | 243 | 2347 |
| 3.3 | 888 | 222 | 2389 |
| 3.4 | 870 | 203 | 2427 |
| 3.5 | 853 | 185 | 2462 |
| 3.6 | 838 | 168 | 2494 |
| 3.7 | 825 | 152 | 2523 |
| 3.8 | 813 | 138 | 2549 |
| 3.9 | 802 | 125 | 2573 |
| 4 | 793 | 113 | 2594 |
| 4.1 | 784 | 102 | 2613 |
| 4.2 | 777 | 92 | 2631 |
| 4.3 | 770 | 83 | 2647 |
| 4.4 | 764 | 75 | 2661 |
| 4.5 | 758 | 68 | 2674 |
| 4.6 | 754 | 61 | 2685 |
| 4.7 | 749 | 55 | 2696 |
| 4.8 | 745 | 49 | 2705 |
| 4.9 | 742 | 45 | 2714 |
| 5 | 739 | 40 | 2721 |
| 5.1 | 736 | 36 | 2728 |
| 5.2 | 733 | 32 | 2734 |
| 5.3 | 731 | 29 | 2740 |
| 5.4 | 729 | 26 | 2745 |
| 5.5 | 727 | 23 | 2749 |
| 5.6 | 726 | 21 | 2753 |
| 5.7 | 724 | 19 | 2757 |
| 5.8 | 723 | 17 | 2760 |
| 5.9 | 722 | 15 | 2763 |
| 6 | 721 | 14 | 2766 |
| 6.1 | 720 | 12 | 2768 |
| 6.2 | 719 | 11 | 2770 |
| 6.3 | 718 | 10 | 2772 |
| 6.4 | 718 | 9 | 2774 |
| 6.5 | 717 | 8 | 2775 |
| 6.6 | 716 | 7 | 2776 |
| 6.7 | 716 | 6 | 2778 |
| 6.8 | 715 | 6 | 2779 |
| 6.9 | 715 | 5 | 2780 |
| 7 | 715 | 5 | 2781 |

| Week | Health | Infected | Deceased |
|---|---|---|---|
| 7.1 | 714 | 4 | 2781 |
| 7.2 | 714 | 4 | 2782 |
| 7.3 | 714 | 3 | 2783 |
| 7.4 | 714 | 3 | 2783 |
| 7.5 | 713 | 3 | 2784 |
| 7.6 | 713 | 2 | 2784 |
| 7.7 | 713 | 2 | 2785 |
| 7.8 | 713 | 2 | 2785 |
| 7.9 | 713 | 2 | 2785 |
| 8 | 713 | 2 | 2786 |
| 8.1 | 713 | 1 | 2786 |
| 8.2 | 713 | 1 | 2786 |
| 8.3 | 712 | 1 | 2786 |
| 8.4 | 713 | 0 | 2787 |

# Appendix B

## Code Implementations

### Chapter 1: ODE Integrators

```
%MA448 Lab 1 – Numerical Methods for IVP
%Problem #1 – Rate of Convergence of ODE Integrators
%Write a program to impliment the a)Expicit Euler's Method,
%b)Improved Euler's Method, c)Optimal order RK2, d)third-order
%Taylor Method, e)the classical fourth-order Runge–Kutta method
%on the following IVP and investigate the convergence of each method

clear all

t0=1;
tmax=6;
N=11; %step number

y1=2; %initial y-value

%Finds estimated output using each method
[t,Y] = euler(t0,tmax,y1,N);
[t,Y1] = improved_euler(t0,tmax,y1,N);
[t,Y2]=taylor3(t0,tmax,y1,N);
[t,Y3] = RK4(t0,tmax,y1,N);
[t,Y4] = RK2(t0,tmax,y1,N);

%Finds the exact value of the desired output
y = exact(t);

%Finds the percent error of each method from the exact that has been found
[E1] = error(y,Y,t0,tmax,N);
[E2] = error(y,Y1,t0,tmax,N);
[E3] = error(y,Y2,t0,tmax,N);
[E4] = error(y,Y3,t0,tmax,N);
```

```
[E5] = error(y,Y4,t0,tmax,N);

%Plots both estimated and exact values onto a graph
figure;
plot(t,y,'-',t,Y,'-',t,Y1,'-',t,Y2,'-',t,Y3,'-',t,Y4,'-');
title('Solutions to the given IVP')
legend('Exact','Euler','Improved Euler','3rd Order Taylor','RK4','RK2','Location
xlabel('x')
ylabel('y')

%Puts the data into a table
fprintf('\n==============================================================')
fprintf('\n    n              tn              Exact            Eulers
Improved      Taylor      RK4          RK2\n')
for i=1:N
    fprintf(' %3.6d \t %3.6f \t %3.6f \t%3.6f \t%3.6f \t%3.6f \t%3.6f \t%3.6f\n'
end
fprintf('==============================================================\n')

%Puts the ERROR data into a table
fprintf('\n=====================Errors=====================')
fprintf('\n   n           Eulers         Improved       Taylor
RK4            RK2\n')
for i=1:N
    fprintf(' %3.3d \t\t%3.3f \t\t%3.3f \t\t%3.3f \t\t%3.3f \t\t%3.3f\n',i-1,E1(
end
fprintf('==============================================================\n')


% Explicit Euler Method
function [t,Y]=euler(t0,tmax,y1,N)
    t=linspace(t0,tmax,N);
    dt=t(2)-t(1);
    Y=zeros(size(t));
    Y(1)=y1;
    for i=1:N-1
        Y(i+1)=Y(i)+dt*f(t(i),Y(i));
    end
end

%Improved Euler Method
function [t,Y]=improved_euler(t0,tmax,y1,N)
    t=linspace(t0,tmax,N);
    dt=t(2)-t(1);
    Y=zeros(size(t));
    Y(1)=y1;
    for i=1:N-1
        Y(i+1) = Y(i) + dt*0.5*(f(t(i),Y(i)) + f(t(i+1),Y(i)) + dt*f(t(i),Y(i)))
    end
end

% Taylor Series order 3
function [t,Y]=taylor3(t0,tmax,y1,N)
    t=linspace(t0,tmax,N);
    dt=t(2)-t(1);
    Y=zeros(size(t));
    Y(1)=y1;
    for i=1:N-1
        Y(i+1)=Y(i)+dt*(  f(t(i),Y(i)) + (dt/2)*df(t(i),Y(i)) ...
                                       + (dt^2/6)*d2f(t(i),Y(i)) );
    end
end

% Classical RK4 Method
function [t,Y]=RK4(t0,tmax,y1,N)
    t=linspace(t0,tmax,N);
    dt=t(2)-t(1);
    Y=zeros(size(t));
    Y(1)=y1;
    for i=1:N-1
        K1 = f(t(i),Y(i));
        K2 = f(t(i)+0.5*dt,Y(i)+0.5*dt*K1);
        K3 = f(t(i)+0.5*dt,Y(i)+0.5*dt*K2);
        K4 = f(t(i)+dt,Y(i)+dt*K3);
        Y(i+1) = Y(i) + (dt/6)*(K1 + 2*K2 + 2*K3 + K4);
    end
end

% Optimal RK2 Method
function [t,Y]=RK2(t0,tmax,y1,N)
    t=linspace(t0,tmax,N);
    dt=t(2)-t(1);
    Y=zeros(size(t));
    Y(1)=y1;
    for i=1:N-1
        K1 = f(t(i),Y(i));
        K2 = f(t(i)+2*dt/3,Y(i)+2*dt*K1/3);
        Y(i+1) = Y(i) + (dt/4)*(K1 + 3*K2);
    end
end

%Percent Error Function
```

```
function [PercError] = error(Ex,App,t0,tmax,N)
    t=linspace(t0,tmax,N);
    dt=t(2)-t(1);
    Exact = Ex;
    Approximation = App;
    for i=1:N
        Offset = Exact(i) - Approximation(i);
        PercError(i) = 100*abs(Offset/Exact(i));
    end
end

% RHS function y'=f(t,y)
function yprime = f(t,y)
    yprime = 7*t.^2 - (4*y/t);
end

%y''=f'(t,y)
function y2prime = df(t,y)
    y2prime = 14*t + (4*y)/(t^2);
end

%y'''=f''(t,y)
function y3prime = d2f(t,y)
    y3prime = 14 - (8*y)/(t^3);
end

% Exact solution
function y = exact(t)
    y = t.^3 + 1./(t.^4);
end
```

# Chapter 2: Epidemic Modeling

```
import numpy as np
import matplotlib.pyplot as plt

"""
M. Catherine Yopp - 5 Feb 2022
MA 448 - Lab 1

THe following code analyzes the Kermack-McKendrick Epidemic Model
in two forms:
    (1) H' = -cHI, I' = cHI - mI, D' = -mI
    (2) D' = m(N - D - H0e^(-cD/m)), H = H0e^(-cD/m), I = N - H - D

Where H = number of healthy individuals, I =  number of infected individuals,
D = number of dead as functions of time, c is the transmission rate of disease
to healthy individuals andm is the mortality rate of infected individuals.

"""


def f(t,x):

    f1 = -c*x[0]*x[1] #the healthy over time
    f2 = (c*x[0]*x[1]) - (m*x[1]) #the infected over time
    f3 = m*x[1] #the deceased over time

    dxdt = [f1, f2, f3]
    return dxdt

def rk4sys(f, time, x):

    M = len(time)
    N = len(x)
    X = np.zeros((M,N))

    h = (time[-1] + time[0])/M
    X[0,:] = x

    for i in range(M -1):
        K1 = f(time[i], X[i,:] )
        K2 = f(time[i] + .5*h, X[i,:] + np.multiply(K1, h*.5))
        K3 = f(time[i] + .5*h, X[i,:]  + np.multiply(K2, h*.5))
        K4 = f(time[i] + h, X[i,:]  + np.multiply(K3, h))
        phi = np.multiply(K1 + np.multiply(K2,2) +
                          np.multiply(K3, 2) + K4, 1/6)
        X[i+1, :] = X[i,:] + np.multiply(phi, h)

        if X[i+1,1] < ideal_infec :

            X.resize((i+1,N))
            newt = np.zeros(i+1)

            for j in range(i+1):
                newt[j] = time[j]
            break

    return newt, X

def dydt(time,y):
    function = m*(cits - y - (H0*np.exp(-c*y/m)))
    return function

def rk4(t0, tmax, y0, steps):
    trk,h = np.linspace(t0,tmax,steps, retstep = True)
    y = np.zeros(steps)
    H = np.zeros(steps)
    I = np.zeros(steps)
    y[0] = y0
    H[0] = H0
    I[0] = I0

    for n in range(steps-1):
          K1 = dydt(trk[n], y[n],)
          K2 = dydt(trk[n] + .5*h, y[n] + .5*h*K1)
          K3 = dydt(trk[n] + .5*h, y[n] + .5*h*K2)
          K4 = dydt(trk[n] + h, y[n] + h*K3)

          y[n+1] = y[n] + ((h/6) * (K1 + (2*K2) + (2*K3) + K4))

          H[n+1] = H0 * np.exp(-c*y[n+1]/m)
          I[n+1] = cits - y[n+1] - H[n+1]

          if I[n+1] < ideal_infec:
              y.resize(n+1)
              H.resize(n+1)
              I.resize(n+1)
              newtrk = np.zeros(n+1)
              for j in range(n+1):
                  newtrk[j] = trk[j]
              break
    return newtrk, y, H, I
################################################################################
if __name__ == "__main__":

    # The triple ode system
    t0 = 0
    tmax = 20 #weeks
    steps = 1000
    #the solution converges by this many steps
    time = np.linspace(t0, tmax, steps)

    m = 1.8 #the mortality rate of the infected per week
    c = .001 #the transmission rate of disease to healthy individuals per week
    ideal_infec = 1
    x0 = [3350, 150, 0]
    [t, xsol] = rk4sys(f, time, x0)

    plt.plot(t,xsol[:,0],'g-',t,xsol[:,1] ,'y-',t,xsol[:,2], 'r-')
    plt.title("Disease's Effects on Citizens\n c = {0:.4f} & m = {1:.1f}\n \
              Triple ODE System".format(c,m), loc = 'center')

    plt.legend(['Healthy','Infected', 'Deceased'], loc = 'best')
    plt.xlabel('# of weeks')
    plt.ylabel('Affected Villagers')
    plt.grid()
    plt.show()

    week = abs((t0 - t[-1])/len(t))
    print('week\t\thealthy\t\t infected\t   deceased\t\t')
    for k in range(len(t) -1):
        print('{0: .1f}\t\t{1: .0f}\t\t\t{2: .0f}\t\t\t{3: .0f}\t\t' \
              .format(week, xsol[k][0], xsol[k][1], xsol[k][2] ))

        week += abs((t0 - t[-1])/len(t))
    print('{0: .1f}\t\t{1: .0f}\t\t\t{2: .0f}\t\t\t{3: .0f}\t\t' \
              .format(week + abs((t0 - t[-1])/len(t)), xsol[-1][0] + 1, \
                      xsol[-1][1] -1 , xsol[-1][2] ))

    #The single ode system
    cits = 3500
    H0 = 3350
    I0 = 150
    y0 = 0

    [trk, Y, H, I] = rk4(t0, tmax, y0, steps)
```

```
plt.plot(trk,H,'g—',trk,I  ,'y—',trk,Y, 'r—')
plt.title("Disease's Effects on Citizens\n c = {0:.4f} & m = {1:.1f}\n \
             Single ODE System".format(c,m), loc = 'center')

plt.legend(['Healthy','Infected', 'Deceased'], loc = 'best')
plt.xlabel('# of weeks')
plt.ylabel('Affected Villagers')
plt.grid()
plt.show()
```

# Chapter 3: Modeling Stem Curve of a Tall Palm in a Strong Wind

```
clear all;
clc;
clear;

s0=0;
smax=30;
N=21;
IC=[0,0,0,0];

ds=(smax-s0)/N;
tspan=[s0,smax];

[s,Y] = rk4sys(@rhs,tspan,IC,ds);

theta = Y(:,1);
theta = theta*(180/pi); % rad to deg
X = Y(:,3);
Z = Y(:,4);


options = odeset('RelTol', 1e-5, 'AbsTol', 1e-6);
[s_ODE45,Y_ODE45] = ode45(@rhs, tspan, IC, options);

% Plots

figure (1)
plot(s,theta,'o-');
title("Theta")
xlabel('Arc Length, s (m)')
ylabel('Theta (deg)')

figure (2)
plot(s,X,'o-');
title("Horizontal Displacement")
xlabel('Arc Length, s (m)')
ylabel('X (m)')

figure (3)
plot(s,Z,'o-');
title("Height of a Location Along The Stem")
xlabel('Arc Length, s (m)')
ylabel('Z (m)')
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Functions  for the numerical procedure
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% RHS function of the ODE
function ddt = rhs(s,theta)

    % Constants
    Ws = 22700; % N
    E = 6e9; % N/m^2
    I = 5.147e-4; %m^4
    L = 30; % m
    Wc = 1385.5; % N
    u = 18; % m/s
    D = 4.135*u^2; % N

    f1 = theta(2);
    f2 = (Ws/(E*I))*(1-(s/L)+(Wc/Ws))*sin(theta(1))+(D/(E*I))*cos(theta(1));
    f3 = sin(theta(1));
    f4 = cos(theta(1));
    ddt = [f1;f2;f3;f4];

end

%RK4 Method
function [s,Y] = rk4sys(rhs,tspan,IC,ds)
    s=[tspan(1):ds:tspan(end)]';
    M=length(IC);
    N=length(s);
    Y=zeros(N,M);
    Y(1,:) = IC';
    for i = 1:N-1
        K1 = rhs(s(i), Y(i,:))';
        K2 = rhs(s(i)+(1/2)*ds,Y(i,:)+(1/2)*ds*K1)';
        K3 = rhs(s(i)+(1/2)*ds,Y(i,:)+(1/2)*ds*K2)';
        K4 = rhs(s(i)+ds,Y(i,:)+ds*K3)';
        Y(i+1,:) = Y(i,:)+(ds/6)*(K1+2*K2+2*K3+K4);
    end
end
```

# Bibliography

[1] Donald F. Winter. On the stem curve of a tall palm in a strong wind. *SIAM Review*, 35(4):567–579, 1993.