AM111: Individual Project Report

Caitlin Chen

December 14, 2022

Abstract

The purpose of the project was to investigate the use of numerical methods to solve initial value problems, specifically the leaky bucket problem, based on the paper "Variations on a Theme of Euler" by R. Corless et al. We aimed to analyze numerical methods such as Forward Euler, Backward Euler, improved Euler (Heun's Method), and two variations of Runge-Kutta on a given IVP. Lastly, we wished to explore the concepts of backward error and optimal backward error through mathematical proofs and application to various ODE's which included both those written by us as well as various MatLab built-ins. My contribution included the implementation of the numerical methods from the original paper, replicating the experimental data on Python, and performing forward, backward, and Heun's method on the experimental data; collaboratively, we were able to analyze the efficacy of these methods with the optimal backward error against the actual data.

1 Introduction

We based our project on "Variations on a Theme of Euler" by Robert M. Corless and Julia E. Jankowski, which is an exploration of numerical methods for solving initial-value problems for ordinary differential equation with both implicit and explicit methods. We chose the leaky bucket problem due to its prevalence in our basis paper, its popularity as an initial-value problem, and its relevance in physics. The leaky bucket problem is a popular example of using mathematics to represent an aspect of the world more simply so that we can make predictions and build future strategies for designing and controlling systems. Due to the fact that we know the behavior shown in the leaky bucket problem should be properly modeled by Torricelli's Law, we have an underlying differential equation that we can use.

Numerical methods, such as the Forward Euler, Backward Euler, Improved Euler's (Heun's Method), and Runge-Kutta are powerful tools for solving a variety of initial value problems, including the Leaky Bucket Problem. These numerical methods work by approximating a problem's exact solution through a series of iterations. When the actual solution to a problem is known, backward Error can be used to determine how much the given data must be perturbed to produce the approximate solution given by a numerical method. One can calculate the optimal backward error by thinking of what an optimal interpolation would be for a given problem, and then explicitly computing the smallest possible backward error on each time step. This optimal backward error can be compared to the backward error of any supplied piecewise polynomial interpolant to determine the quality and accuracy of a given interpolation.

The interesting part of the project is in its comparisons between these numerical methods; our experimental data is flawed and does not perfectly conform to the theoretical prediction of the water's movement, but we want to evaluate the difference in the numerical methods in approximating a solution with our known ordinary differential equations. We can understand their accuracies with optimal backward error and by finding the residuals. While we collaborated on the analysis of numerical methods on the data, I contributed by converting the numerical methods and Sarah's raw experimental data and implementing the ODE function from the derivation of Torricelli's law and the ODE for $\frac{dh}{dt}$ and $\frac{dy}{dt}$ into our code. We then implemented forward, backward, and improved Euler's method onto our data and overlayed it over the experimental data to understand how they were predicting the movement of the water.

2 Background

An ODE (ordinary differential equation) model is a set of differential equations involving functions of only one independent variable and one or more of their derivatives with respect to that variable. In the context of physics in the leaky bucket problem, we know that the ODE for modeling the height of water (h) can be found by the velocity v, which is derived from the conservation of energy, such that the change in potential energy over the entire height h because the leak is at the bottom of the bucket, which is consistent with the experimental data as well.

The methods that we used is a combination of the forward Euler, backward Euler, Heun's method, and RK4 methods. I needed to learn how to implement the new ODE into our code and since I have less coding experience, it was necessary for me to learn how to become familiar with semantics regarding array iteration in order to implement the new implicit and explicit methods. The forward Euler method is used to approximate the next value of the function given the current value of the function and the derivative of the function. The backward Euler method is used to approximate the current value of the function given the next value and the derivative of the function. This method is used to create a numerical solution to the ODE which can then be used to analyze the behavior of the system. Although my teammates worked on the backward error method implementation, it was a new topic to me so I needed to learn that aspect of both the math and then how it was implemented into Python.

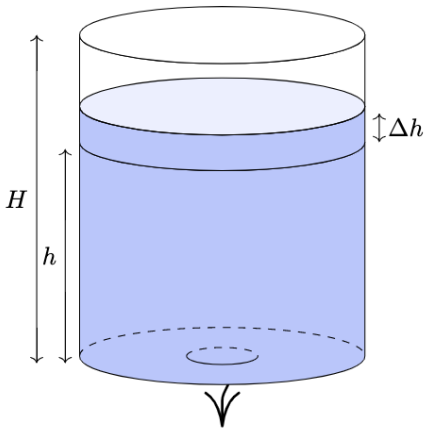

Figure 1: Diagram showing the leaky bucket problem and the relationship between h and v, of water leaving the bucket. Adapted from "Variations on a Theme of Euler," by Corless, Robert, 2016, retrieved from https://epubs-siam-org.ezp-prod1/10.1137/15M1032351

3 Methods

Since I was responsible for the implementation of the numerical methods, I implemented the ODE into our code based on Sarah's Matlab data and also based on the supplementary code used in Corless' paper. The fundamental math describes the relationship between h and dh/dt.

In some time t, the volume lost out of the bucket is $A * h'(t)\Delta t$, where A is the cross-sectional circle area of the cylindrical bucket, and $h'(t) = dh/dt$. Letting $y = h/H$, we then get: $2gh = v^2 = (\frac{A}{a}\frac{dy}{dt})^2$. Rearranging yields that $\frac{dy}{dt} = -\frac{a}{A}\sqrt{\frac{2g}{H}} * \sqrt{y}$. A rescaling of time simplifies this to $\frac{dy}{dt} = -\sqrt{y}$.

In terms of h, a similar derivation of $v = \frac{dh}{dt}$ comes from the conservation of energy from the sum of potential energy and kinetic energy before and after the water leaking is

$$p_1 + pgh_1 + \frac{1}{2}pv_1^2 = p_2 + pgh_2 + \frac{1}{2}pv_2^2$$

which yields that

$$\frac{dh}{dt} = \sqrt{2gh}.$$

We implemented this into the definition of our ODE in Python, and converted the units such that the experimental data was equivalent to the g of Torricelli's law. This is the ODE we used since we are directly interested in the h we have from the experimental data. The setup reflects that the initial boundary x_0 is the initial height of the water measured in the bucket in the array of experimental data.

In the implementation, we used varying step sizes ranging from $h = .3, .1, .05, .01$. Forward Euler is explicitly defined, where $x_{k+1} = x_k + h * \frac{dx}{dt}x_k$. Here, we plug in for $f(x)$ the previously defined ODE of dh/dt.

Backward Euler is implicitly solved such that we define $\frac{dx}{dt} = \frac{1}{h} * (x_k - x_{k-1}$, and then plugging that into the approximation into the original ODE $\frac{y_n - y_{n-1}}{h}$. In the code implementation, this is achieved using scipy.

Lastly, Heun's Method is a combination of the above, also known as Improved Euler's method, which is defined by first defining both forward and backward Euler within the function, and then returning the average of the two.

Both the methods have order 1, so they are consistent and global error grows in proportion to step size h. In the graphs, only solutions with small h show, which is because larger h the solution explodes towards the end when the raw data begins to deviate from the theoretical expectation.

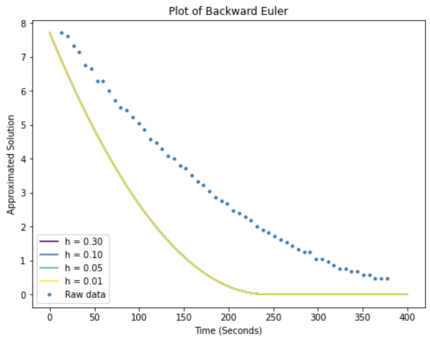

Figure 2: Backward Euler

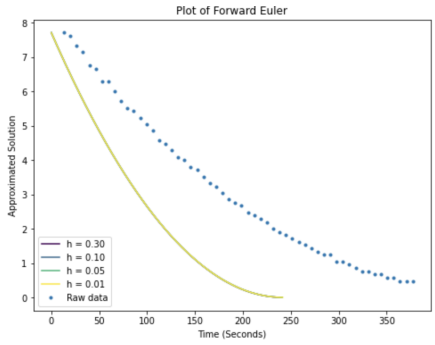

Figure 3: Forward Euler

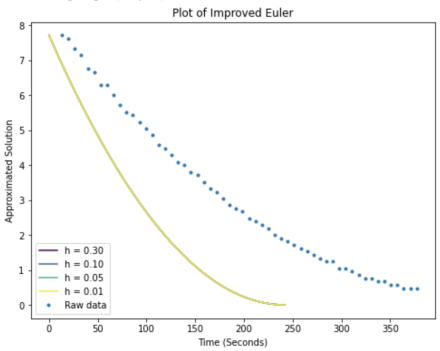

Figure 4: Improved Euler

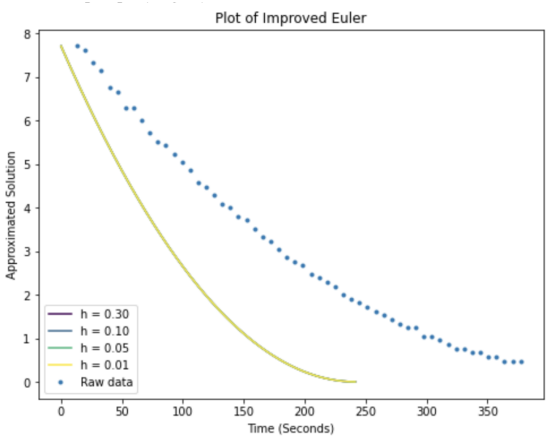

Figure 5: Plot of All Methods (One Graph) with h=0.01

4 Results

This was not exactly my area of focus, but we plotted the results for each of our functions with differing step sizes on their own graph, and then plotted all of the functions using a step size of h = 0.01 on the same graph to better compare and analyze our results. From our results, it seems that each of the five methods gave us roughly the same exact graph, and ultimately the same residuals.

5 Discussions

I really enjoyed learning how to implement the baseline code because moving forward, we can use the numerical methods on effectively any data set as long as we can plug in an ODE. Moving forward, I think it would be really interesting to use the numerical methods we have in potentially the growth/spread of Covid in a region given the data, and perhaps using a predicted ODE from an existing paper studying the growth of Covid. Something that did not work out was the graph of optimal backward error of the improved euler method, and attempting to plot the residuals and optimal backward error for different ODE solvers in python; instead they used Matlab.

References

AM111: Problem Set 9 for coding reference
R. Corless et al. *Variations on a Theme of Euler*, supplementary materials and background problem