Numerical Investigations of Nonlinear First-Order Ordinary Differential Equations

Alexis Guevara, Alex Gutierrez, Johnny Sierra, Amelia Threatt

August 15, 2022

Abstract

Since many real-world phenomena such as population change can be modeled through nonlinear ordinary differential equations, our team investigated numerical schemes for solutions of nonlinear first-order Ordinary Differential Equations. We compare Non-Standard Finite Differences (NSFDs) derived from non-local approximations to unity approximations of Standard Finite Difference methods (SFDs) like Euler's scheme to show that NSFD approximations have less error than standard methods. We applied many NSFD unity approximations to cubic decay equations $u' = -u^3 + F(u)$, where $F(u) = 0, u^2, u^3$. We were successful in creating NSFD schemes for the cubic decay, Bernoulli, and combustion equations. Our investigation came to end while looking for an NSFD for the Bratu initial value problem. For the Bratu IVP we were successful in finding an NSFD scheme for it as a 2D system.

1 Introduction

Why do we care about Ordinary Differential Equations? Differential Equations are important because they can be used to model many real-world phenomena, such as population change (ODE), heat movement (PDE), or other behaviors that deal with change over time. However, unless you have experience in numerical analysis you know solutions of ODE's are continuous functions often found through integration. The problem with relying on calculus to solve differential equations is that most ODE's modeling real-world phenomena are impossible to integrate so numerical solutions are necessary. Therefore, we are trying to find numerical solutions of ODE's that outperform standard methods. Our research involves investigation of Non-Standard Finite Differences to solve Initial Value Problems on the cubic decay equations which is in the form $u' = -u^3 + F(u)$ such that $F(u) = 0, u^2, u^3$. Moreover, a differential equation is an equation that relates a function and its derivative. One simple example is

$$f' = f.$$

This differential equation says the derivative of f is itself. From calculus, we know the exponential function has this property. Thus the general solution to the differential equation is $f = Ce^x$, where C is some constant.

When we can solve a differential equation and obtain a closed-form explicit solution, we say the solution is *analytical.* However, this can be impossible for some differential equations, and so we resort to *numerical* solutions that approximate the true solution.

Recall the definition of the derivative of a function f is given by

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}.$$

So, how do we discretize an ODE so that is can become a numerical approximation. First, we partition the domain into n parts. Then at each point we evaluate the function $f(x_n) = y_n$. After evaluating the function, we get corresponding values (x_n, y_n), there will be $n + 1$ of them. To evaluate a function we use a Standard Finite Difference. If we take h to be a fixed value (rather than letting h approach 0), we can approximate the derivative with

$$\frac{f(x+h) - f(x)}{h}, \; f(x_0) = y_0.$$

Set $x_n = x_0 + nh$, then if y_n is the approximation to $f(x_n)$, $y_n \approx f(x_n)$

$$\frac{y_{n+1} - y_n}{h} = f'(x_n, y_n) \tag{1}$$

What is shown in 1 is defined as a Standard Finite Difference (SFD). This one in particular is also referred to as Euler's Scheme. After applying some algebra to solve for y_{n+1} we get $y_{n+1} = y_n + hf(x_n, y_n)$. However, this standard difference method produces errors which is $|y_{n+k} - y_(x_{n+k})|$, we take the absolute value because the numerical approximation value at a point may be greater or less than the true value at that same point. Euler's scheme or method is a numerical approximation technique for first-order ordinary differential equations. It is worth noting that Euler's method produces a first-order scheme. The order of convergence refers to the rate at which the scheme converges to the true solution.

The more we partition the domain the less error we get since if you partition it infinite amount of times it will become the continuous function. Our goal is to minimize the error without having to partition so greatly. We can do this by converting an SFD into an NSFD. What takes a standard finite difference scheme to a NSFD scheme are the introduction of functions ψ and ϕ that have specific properties. An NSFD scheme is in the general form:

$$\frac{du}{dt} = \frac{u_{k+1} - \psi u_k}{\phi}, \tag{2}$$

Such that $\psi(h) = 1 + O(h^2)$ and $\phi(h) = h + O(h^2)$. However, we usually keep our $\psi(h) = 1$ and $\phi(h) = h$ to simplify our NSFD.

There are two rules for creating NSFD schemes. Note that with these rules, it is possible to create various NSFDs schemes.

1. The traditional denominator h is replaced by a non-negative non-standard denominator function $\phi(h)$ such that $\phi(h) = h + O(h^2)$ as $h \to 0$. An example is $1 - e^{-h}$ because expanding e^{-h} as a Taylor series gives: $1 - h + \frac{-h^2}{2!} + \frac{-h^3}{3!} + \ldots$, Thus $1 - e^{-h} = h + \frac{h^2}{2} + \ldots = h + O(h^2)$.

2. Right hand-sided terms are approximated in a non-local way for example by a suitable function of several points of the mesh i.e.:

$$-u^2 = -u(t_k)^2 = -u_k^2 = -u_{k+1}u_k$$

$$-u^3 = -u(t_k)^3 = -u_k^3 = \frac{2u_{k+1}^2 u_k^2}{u_{k+1} + u_k}$$

Although there are only two rules to create NSFDs there are many ways to discretize an ODE to create schemes with these rules. Here are 3 simple examples:

Consider the ODE: $u' = -u^3$

Table 1: NSFD Examples

Standard Method	Non-standard Denominator	Non-local Terms	Non-standard/local Denominator/Terms
$\frac{u_{k+1}-u_k}{h} = -u_k^3$	$\frac{u_{k+1}-u_k}{\cdot 1-e^{-h}} = -u_k^3$	$\frac{u_{k+1}-u_k}{h} = \cdot -u_k^2 u_{k+1}$	$\frac{u_{k+1}-u_k}{\cdot 1-e^{-h}} = \cdot -u_k^2 u_{k+1}$

There is only one way to discretize an ODE using an SFD but there are many ways to manipulate and approximate the SFD to create NSFDs. After creating NSFD schemes our goal is to find u_{k+1} in terms of u_k to compare it to the true solution and verify that the approximation is accurate.

Mickens mentions that dynamic consistency means that the numerical solutions generated by NSFDs replicate some of the properties of the solutions of the continuous system. In other words, properties are preserved through the discretization of a differential equation. This causes the numerically approximated solutions to have the same properties as the true solution. Some properties associated with dynamic consistency are positivity, boundedness, monotonicity, stability, fixed points, order, symmetries, asymptotic behavior, and many others.

In summary, a Non-Standard Finite Difference (NSFD) scheme is a numerical method used to solve differential equations, and is found to often be more accurate than existing standard difference methods, can preserve important properties of the ODE, and may yield exact solutions which means there is absolute zero error.

1.1 Unity Approximation

Our research focused on creating NSFD Unity approximation schemes using non-local terms. Since multiplying any function or term by one does not change the value we decided to approximate one with non-local terms. For example, since $1 = \frac{u}{u} = \frac{u_k}{u_k} = \frac{u_{k+1}}{u_k}$ then we can multiply any standard difference by 1 and approximate it using non-local terms to create NSFD Unity Approximation schemes. Consider the SFD for the ODE $u' = u^2$. The SFD would be

$$\frac{u_{k+1} - u_k}{h} = u_k^2$$

. To convert this to an NSFD Unity approximation, we simply multiply the right-hand term by 1.

$$\frac{u_{k+1} - u_k}{h} = u_k^2 \cdot 1$$

Next, we approximate that 1 using non-local terms, here are a few examples:

$$\frac{u_{k+1} - u_k}{h} = u_k^2 \cdot \frac{u_{k+1}}{u_{k-1}} \qquad \frac{u_{k+1} - u_k}{h} = u_k^2 \cdot \frac{u_k + u_{k-1}}{2u_k} \qquad \frac{u_{k+1} - u_k}{h} = u_k^2 \cdot \frac{u_k^2}{u_{k+1}^2} \qquad \frac{u_{k+1} - u_k}{h} = u_k^2 \cdot \frac{u_{k-1}^3}{u_{k+1}^3}$$

All of these are valid NSFD Unity approximations but some are better than others depending on the term it is being multiplied to. In this case, the term is u_k^2 so we hypothesize that the third option would have the best results since the Unity Approximation also contains a square. Furthermore, we have created some names for the unity approximations that we used throughout our research.

Table 2: Unity Approximations

Unity Approximation	No Approximation	Denominator: u_{k+1}	Denominator: u_k
Simple Unity Approximation	1	$\frac{u_k}{u_{k+1}}$	$\frac{u_{k+1}}{u_k}$
Square Unity Approximation	1	$\frac{u_k^2}{u_{k+1}^2}$	$\frac{u_{k+1}^2}{u_k^2}$
Cubic Unity Approximation	1	$\frac{u_k^3}{u_{k+1}^3}$	$\frac{u_{k+1}^3}{u_k^3}$
2-Point Averaging Unity Approximation	1	$\frac{u_k+u_{k+1}}{2u_{k+1}}$	$\frac{u_k+u_{k+1}}{2u_k}$
2-Point Square Averaging Unity Approximation	1	$\frac{u_k^2+u_{k+1}^2}{2u_{k+1}^2}$	$\frac{u_k^2+u_{k+1}^2}{2u_k^2}$
2-Point Cubic Averaging Unity Approximation	1	$\frac{u_k^3+u_{k+1}^3}{2u_{k+1}^3}$	$\frac{u_k^3+u_{k+1}^3}{2u_k^3}$

2 The Cubic Decay Equation

The cubic decay equation is the initial value problem

$$u' = -u^3, \quad u(0) > 0. \tag{3}$$

This differential equation is "separable" and an analytical solution is easily found with integration. Equation (4) gives the solution to (3) with initial value $u(0) = 1$.

$$u(t) = \frac{1}{\sqrt{2t+1}} \tag{4}$$

For any numerical scheme of (3), we only take a limited range of t values near $t = 0$. Thus we will keep our focus on the interval $[0, 1]$. To partition $[0, 1]$, we let $h = 1/2^n$, where n is a natural number. The constant h is referred as the step-size because we start at the initial point, $t_0 = 0$, and take a "step" that is h long to reach the next point $t_1 = t_0 + h = h$. From t_1, take another step that is h long to reach $t_2 = t_1 + h = t_0 + 2h = 2h$. Repeat this process until we reach the point $t_{2^n} = t_0 + 2^n h = 1$

With the explicit solution known, we can find the error of the estimated value of t_k with the simple formula

$$\text{error} = |u_{\text{exact}} - u_{\text{approx}}|,$$

where u_{exact} is the exact value of $u(t_k)$ found by using (4) and u_{approx} is the approximated value of $u(t_k)$ obtained by using a numerical scheme.

The first numerical scheme we will look at is the SFD scheme known as Euler's Method. We discretize (3) to get

$$\frac{u_{k+1} - u_k}{h} = -u_k^3. \tag{5}$$

We solve for u_{k+1} to obtain

$$u_{k+1} = -hu_k^3 + u_k. \tag{6}$$

We judge an NSFD scheme as "good" if it does better than Euler's Method. The criteria for good schemes are satisfying one of the following:

1. The NSFD scheme is order 2 or higher (Euler's Method is order 1).
2. The NSFD scheme is order 1 but has less error than Euler's Method.

There are many ways to create NSFD schemes. For this research, our method was to use unity approximations. We have tried many unity approximations and most were order 1, but had less error than Euler's method. However, one unity approximation did give us an order 2 scheme.

To do a unity approximation, we take (5) and multiply the right hand side by 1 (a.k.a. "unity").

$$\frac{u_{k+1} - u_k}{h} = -u_k^3 \cdot 1. \tag{7}$$

For the approximation, we have tried three types of averaging schemes in the form

$$1 \approx \frac{u_{k+1}^p + u_k^p}{2u_k^p}$$

where $p = 1, p = 2$, or $p = 3$. For $p = 2$ and $p = 3$, we say this averaging is a "square averaging" and a "cubic averaging", respectively.

The continuity of u and a very small step size h is what guarantees that u_{k+1} is very near u_k. Hence the numerator of the fraction is almost the same value as the denominator, which makes the fraction approximately 1. If $p = 3$, then (7) becomes

$$\frac{u_{k+1} - u_k}{h} = -u_k^3 \cdot \frac{u_{k+1}^3 + u_k^3}{2u_k^3}, \tag{8}$$

which simplifies to

$$\frac{u_{k+1} - u_k}{h} = \frac{1}{2}(u_{k+1}^3 + u_k^3).$$

The next step is to solve for u_{k+1}. This is algebraically complicated due to the cubic term. We used WolframAlpha to help solve for u_{k+1}. Equation (9) gives the explicit expression for u_{k+1}.

$$\begin{aligned} u_{k+1} = {} & \frac{\sqrt[3]{-27h^3u_k^3 + 54h^2u_k + \sqrt{864h^3 + \left(54h^2u_k - 27h^3u_k^3\right)^2}}}{3\sqrt[3]{2}h} \\ & - \frac{2\sqrt[3]{2}}{\sqrt[3]{-27h^3u_k^3 + 54h^2u_k + \sqrt{864h^3 + \left(54h^2u_k - 27h^3u_k^3\right)^2}}} \end{aligned} \tag{9}$$

The scheme given in (6) is known to be order 1. Thus we first check if the scheme in (9) is higher than order 1. To do this, we represent the step-size h as Δt_k, where k is a non-negative integer. The step size will be given by the formula

$$\Delta t_k = \frac{1}{2^{k+1}}.$$

To investigate the order, we run through consecutive k values. For Table 3 we have $k = 1, \ldots, 5$. We calculate Δt_k ratio as follows:

$$\frac{\Delta t_k}{\Delta t_{k-1}} = \frac{1/2^{k+1}}{1/2^{k+2}} = 1/2, \quad \text{for all } k.$$

There are many ways to capture the error of the method. For this research project, we use the L^∞ error. This error is found by the formula in (10).

To compute the order of our method, we need the Δt_k ratio and the ratio of errors. If E_k represents the L^∞ error for the step-size Δt_k then the ratio of error is found by computing E_k/E_{k-1}. To compute the order we use the formula in (11).

$$L^\infty(\text{error}) = \max_{1\le i\le N} |\text{error}_i| \tag{10}$$

$$\text{order} = \frac{\log_2(E_k/E_{k-1})}{\log_2(\Delta t_k/\Delta t_{k-1})} \tag{11}$$

We organize these values into an Error Analysis table. Table 3 is the error analysis for the scheme given in (9).

Table 3: Error Analysis Between Cubic Averaging Method and True Solution for Differential Equation $u'(t) = -u^3$, $u(0) = 1$, and is bounded between $(0, 1)$; Δt is cut in half

Δt	Δt Ratio	L^∞ Error	Ratio of Errors	Order
$1/2^2$	$\sim$	$7.64464041e-03$	$\sim$	$\sim$
$1/2^3$	$1/2$	$1.83461895e-03$	$1.83461895e-03/7.64464041e-03 = 0.2310$	2.1140
$1/2^4$	$1/2$	$4.55164060e-04$	$4.55164060e-04/1.83461895e-03 = 0.2481$	2.0110
$1/2^5$	$1/2$	$1.13543772e-04$	$1.13543772e-04/4.55164060e-04 = 0.2495$	2.0029
$1/2^6$	$1/2$	$2.83715248e-05$	$2.83715248e-05/1.13543772e-04 = 0.2499$	2.0006

The table gives an indication that the NSFD scheme is order 2. This is seen as better than Euler's method because the error of the NSFD scheme converges to 0 faster than the SFD. Other unity approximations (see Table 4) to the cubic average differential equation were order 1 but generally performed better than the SFD by having less error.

Scheme	Unity
A	$\frac{u_k}{u_{k+1}}$
B	$\frac{u_{k+1}}{u_k}$
C	$\frac{u_k + u_{k+1}}{2u_k}$
D	$\frac{u_k + u_{k+1}}{2u_{k+1}}$
E	$\frac{u_{k+1}^2}{u_k^2}$
F	$\frac{u_k^2}{u_{k+1}^2}$
G	$\frac{u_k^2 + u_{k+1}^2}{2u_k^2}$
H	$\frac{u_k^2 + u_{k+1}^2}{2u_{k+1}^2}$

Table 4: Various unity approximations for (7).

Mickens, by using a change of variables, was able to find an exact NSFD of (3) in [1]. This can also be achieved by using the NSFD scheme in Equation (12).

$$\frac{u_{k+1} - u_k}{h} = -\left(\frac{2u_{k+1}}{u_{k+1} + u_k}\right) u_{k+1} u_k^2 \tag{12}$$

Solving for u_{k+1} in (12) produces an exact solution, given in (13).

$$u_{k+1} = \sqrt{\frac{u_k^2}{1+2u_k^2 h}} \tag{13}$$

The results for this first investigation of NSFDs was positive as we were able to create schemes and find schemes that outperformed standard methods.

3 The Bernoulli Equation

The Bernoulli differential equation is

$$u' = -u^3 + u, \quad u(0) = u_0 > 0. \tag{14}$$

To solve for an explicit solution, integrate both sides and use partial fractions.

$$u' = \frac{du}{dt} = -u^3 + u = u(1-u^2)$$
$$\int \frac{du}{u(1-u^2)} = \int dt$$

$$\frac{1}{u}\cdot\frac{1}{(1-u^2)} = \frac{1}{u}\cdot\frac{1}{1+u}\cdot\frac{1}{1-u} = \frac{A}{u} + \frac{B}{1+u} + \frac{C}{1-u}$$
$1 = A(1+u)(1-u) + B(1-u)(u) + C(1+u)(u)$
$u = 0$
$1 = A\cdot 1\cdot 1 + 0 + 0 \to A = 1$
$u = 1$
$1 = 0 + 0 + C\cdot 2\cdot 1 \to C = \frac{1}{2}$
$u = -1$
$1 = 0 + B\cdot 2\cdot -1 + 0 \to B = \frac{-1}{2}$
$$\frac{1}{u(1-u^2)} = \frac{1}{u} - \frac{\frac{1}{2}}{u-1} + \frac{\frac{1}{2}}{1-u}$$

$$\int \frac{1}{u}du - \frac{1}{2}\int \frac{1}{1+u}du + \frac{1}{2}\int \frac{1}{1-u}du = \int dt$$

$$\ln u - \frac{1}{2}\ln(1+u) - \frac{1}{2}\ln(1-u)$$

$$\ln u - \frac{1}{2}\ln(1-u^2)$$

$$\ln\left(\frac{u}{\sqrt{1-u^2}}\right) = t + c$$

$$\frac{u}{\sqrt{1-u^2}} = e^t + e^c$$

let $e^c = D$

$$\frac{u}{\sqrt{1-u^2}} = De^t$$

Solving for u in terms of t

$$u(t) = \frac{\frac{u_0}{\sqrt{1-(u_0^2)}}\cdot e^t}{\sqrt{1+\frac{u_0}{\sqrt{1-(u_0^2)}}^2\cdot e^{2t}}} \tag{15}$$

This equation was first introduced as a modified Combustion equation, but then later found to meet the requirements of a Bernoulli differential equation.

Definition 3.1 (Bernoulli Differential Equation)**.** Bernoulli differential equations take the form

$$u' + p(t)u = q(t)u^n$$

Where $p(t)$ and $q(t)$ are continuous functions on a given interval, and n is some real number.

When $p(t) = q(t) = -1$ and $n = 3$, then $u' = -u^3 + u$ thus proving it is a Bernoulli differential equation. Bernoulli differential equation are special in the way that they are nonlinear differential equations with known solutions, and along with that all Bernoulli differential equation have exact NSFD Schemes.

Using the exact NSFD Scheme from the paper by Wang and Roeger [6], where we had to solve (16) for u_{k+1} yields (17)

$$\frac{u_{k+1} - u_k}{1 - e^{-h}} = \frac{u_k^2(1 - e_{k+1}^2)}{\frac{u_{k+1}^2 - u_k^2}{u_{k+1} - u_k} \cdot \frac{1-e^{-h}}{1-e^{-2h}}} \tag{16}$$

The equation below is the exact NSFD for this specific Bernoulli Equation.

$$u_{k+1} = \frac{\sqrt{e^{2h}} \cdot u_k}{\sqrt{(e^{2h} - 1)\, u_k^2 + 1}} \tag{17}$$

Table 5: Error Analysis of Exact NSFD Scheme (17) with initial condition $u_0 = 0.5$

Δx	L^1 Error	L^2 Error	L^∞ Error
0.250000	$5.55111512e-16$	$3.33066907e-16$	$2.22044605e-16$
0.125000	$8.88178420e-16$	$3.51083347e-16$	$2.22044605e-16$
0.062500	$1.04360964e-14$	$3.01195956e-15$	$1.22124533e-15$
0.031250	$6.66133815e-15$	$1.47287728e-15$	$6.66133815e-16$
0.015625	$6.37268016e-14$	$9.34567779e-15$	$1.99840144e-15$

This scheme showed an L^∞ of order 10^{-16}, despite this NSFD Scheme being labeled as exact the expect error would be 0, however to the computer 10^{-16} is 0. So although this error is not exactly 0, it being order 10^{-16} is an indicator that this scheme is numerically exact compared to the analytical solution.

Although there exists an exact NSFD scheme, the work is not done for this equation. Given that there exists an analytic explicit solution this is an optimal equation to further test some of our averaging unity approximations. Following the same logic from the cubic decay equation we implemented an averaging unity approximation, however since there are two terms in this equations we used two separate averaging unity approximations, with corresponding powers to the term its being multiplied by.

$$\frac{u_{k+1} - u_k}{h} = -u_k^3 \cdot 1 + u_k \cdot 1 = -u_k^3 \cdot \left(\frac{u_k^3 + u_{k+1}^3}{2u_k^3}\right) + u_k \cdot \left(\frac{u_k + u_{k+1}}{2u_k}\right) \rightarrow$$

$$u_{k+1} = \frac{1}{3\sqrt[3]{2}h}\left(\left(-27h^3u_k^3 + 27h^3u_k + 54h^2u_k + \sqrt{108\,(2-h)^3\, h^3 + (-27h^3u_k^3 + 27h^3u_k + 54h^2u_k)^2}\right)^{1/3}\right)$$
$$- \frac{\sqrt[3]{2}(2-h)}{\left(\left(-27h^3u_k^3 + 27h^3u_k + 54h^2u_k + \sqrt{108\,(2-h)^3\, h^3 + (-27h^3u_k^3 + 27h^3u_k + 54h^2u_k)^2}\right)^{1/3}\right)}$$

Table 6: Error Analysis Between Cubic Average + Average Unity Approximation Scheme and True Solution for Differential Equation $u'(t) = -u^3 + u$, $u(0) = 0.5$, and is bounded between $(0, 1)$; Δt is cut in half

Δt	Δt Ratio	L^∞ Error	Ratio of Errors	Order
0.250000	0.50000	$1.49874077e-03$	$\sim$	$\sim$
0.125000	0.50000	$3.75058277e-04$	$3.75058277e-04/1.49874077e-03 = 0.2502$	1.9988
0.062500	0.50000	$9.37847180e-05$	$9.37847180e-05/3.75058277e-04 = 0.2500$	2.0000
0.031250	0.50000	$2.34473894e-05$	$2.34473894e-05/9.37847180e-05 = 0.2500$	2.0000
0.015625	0.50000	$5.86466333e-06$	$5.86466333e-06/2.34473894e-05 = 0.2501$	1.9994

After examination of the error in Table 4 after calculating the order using the same methods as beforehand it can be seen to be approaching a second order convergence. This NSFD Scheme used similar averaging unity approximations as the cubic decay equation and for it to have provide similar results is promising and leads us to want to apply this technique on other equations.

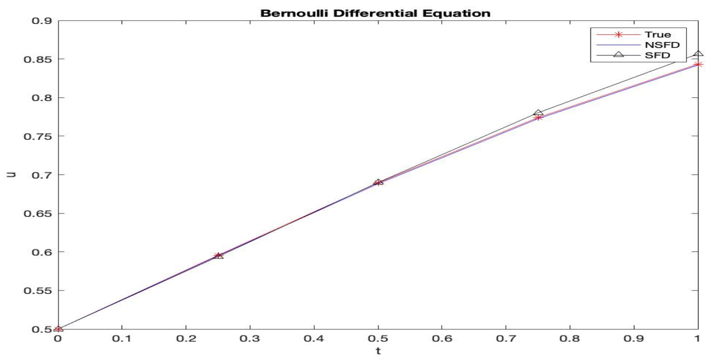


Figure 1: True, NSFD, and SFD Plot

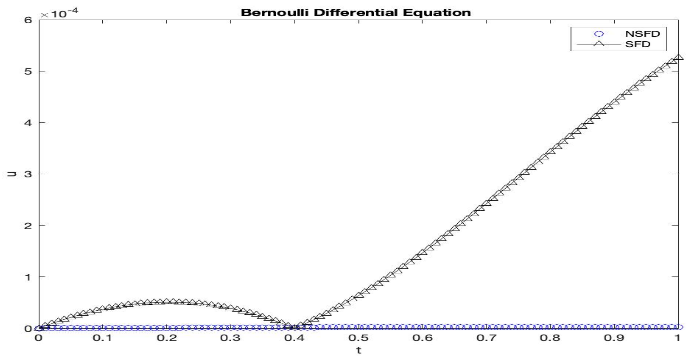


Figure 2: SFD and NSFD Error Plot

As expected in Figure 2 it is apparent that the error of the NSFD for the Bernoulli Differential equation has a smaller area under the curve compared to the area under the curve for the Standard Finite Difference Scheme. The smaller area indicates a a smaller difference between the numerical solution and the analytical explicit solution at a given point.

4 The Combustion Equation

The Combustion Differential Equation is

$$u' = -u^3 + u^2, \quad u(0) = u_0 > 0. \tag{18}$$

In an attempt to solve for an explicit solution for the differential equation, an approach was made by separating variables and integrating both sides of the equation, which yields

$$\int \frac{1}{-u^3+u^2}\,du = \int 1\,dt. \tag{19}$$

To solve the integral on the left-hand side, we will utilize partial fraction decomposition.

$$\begin{aligned}\frac{1}{-u^3+u^2} &= \frac{1}{u}\cdot\frac{1}{1-u^2}\\ &= \frac{1}{u}\cdot\frac{1}{u+1}\cdot\frac{1}{1-u}\\ &= \frac{A}{u}+\frac{B}{1+u}+\frac{C}{1-u}\end{aligned}$$

We should have that

$$A(1+u)(1-u)+B(1-u)(u)+C(1+u)(u)=1$$

. By letting $u=0, u=1$, and $u=-1$, we find that $A=1, B=-1/2, C=1/2$. We will also write $(1-u)$ as $-(u-1)$. Therefore, the decomposition then becomes

$$\int \frac{1}{-u^3+u^2}\,du = \int \frac{1}{u}\,du - \frac{1}{2}\int \frac{1}{u+1}\,du - \frac{1}{2}\int\frac{1}{u-1}\,du$$

which evaluates to

$$\log(u) - \frac{1}{2}\log(u+1) - \frac{1}{2}\log(u-1)$$

. Substituting the results to Equation 19 results in the equation becoming

$$\log(u) - \frac{1}{2}\log(u+1) - \frac{1}{2}\log(u-1) = t + C.$$

Thus, the combustion differential equation does not have an explicit solution. As a result, we will use a 7th order Runge-Kutta method for our numerical comparisons to SFD and NSFD schemes. Our approach to constructing a unity approximation uses similar ideas to those in creating a unity approximation for the Cubic Decay Equation and the Bernoulli Equation mentioned earlier by utilizing an averaging term with its power corresponding to the term it is multiplied by. Again, using Wolfram Alpha to solve the differential equation for u_{k+1} produces

$$\frac{u_{k+1}-u_k}{h} = -u_k^3\cdot 1 + u_k^2\cdot 1 = -u_k^3\cdot\left(\frac{u_k^3+u_{k+1}^3}{2u_k^3}\right) + u_k^2\cdot\left(\frac{u_k^2+u_{k+1}^2}{2u_k^2}\right)\rightarrow$$

$$\begin{aligned}u_{k+1} = -((\sqrt[3]{2}(6h-h^2))&\frac{1}{(3h(-27h^3u_k+27h^3u_k^2+2h^3+54h^3u_k-18h^2}\\ &+\sqrt{(4(6h-h^2)^3+(-27h^3u_k^3+27h^3u_k^2+2h^3+54h^2u_k-18h^2)^2))^{1/3}}))\\ &+\frac{1}{3\sqrt[3]{2}h}((-27h^3u_k^3+27h^3u_k^2+2h^3+54h^2u_k-18h^2\\ &+\sqrt{4\left(6h-h^2\right)^3+\left(-27h^3u_k^3+27h^3u_k^2+2h^3+54h^2u_k-18h^2\right)^2}^{1/3})+\frac{1}{3}\end{aligned}$$

We can now analyze the error between the 7th order Runge-Kutta method and the discrete schemes to determine the order of the numerical schemes for the combustion equation. From observing the table below, we can deduce that the unity approximation we constructed utilizing averaging terms results in a second order Non-Standard Finite Difference scheme.

Table 7: Error Analysis Between Our Unity Approximation Scheme and 7th order Runge-Kutta Method for Differential Equation $u'(t) = -u^3 + u^2$, $u(0) = 2$, and is bounded between $(0, 1)$; Δt is cut in half

Δt	Δt Ratio	L^∞ Error	Ratio of Errors	Order
0.250000	0.50000	$8.62138310e-02$	$\sim$	$\sim$
0.125000	0.50000	$2.13858740e-02$	$2.13858740e-02/8.62138310e-02 = 0.2481$	2.0110
0.062500	0.50000	$4.90232643e-03$	$4.90232643e-03/2.13858740e-02 = 0.2292$	2.1253
0.031250	0.50000	$1.19981796e-03$	$1.19981796e-03/4.90232643e-03 = 0.2447$	2.0309
0.015625	0.50000	$2.98363181e-04$	$2.98363181e-04/1.19981796e-03 = 0.2487$	2.0075

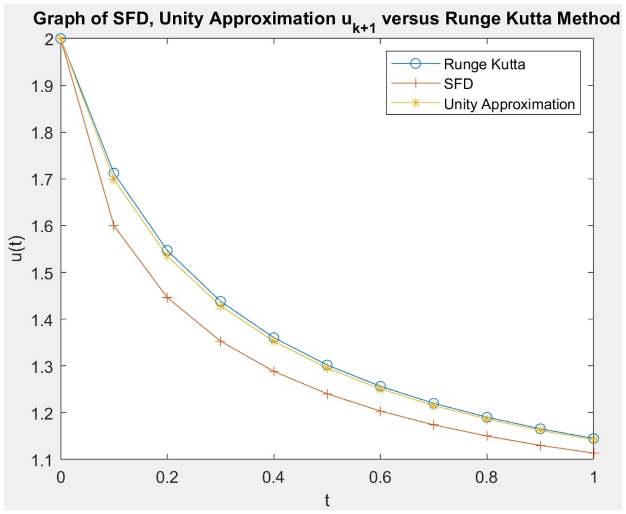


Figure 3: Plot of SFD Scheme and our NSFD Unity Approximation Scheme against the 7th Order Runge-Kutta Method for the Combustion Equation

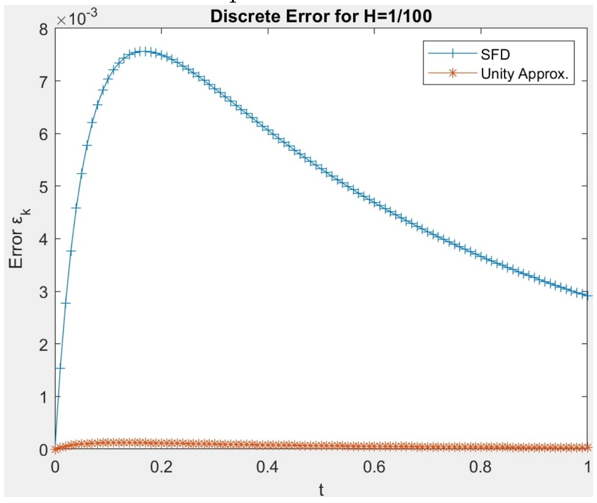
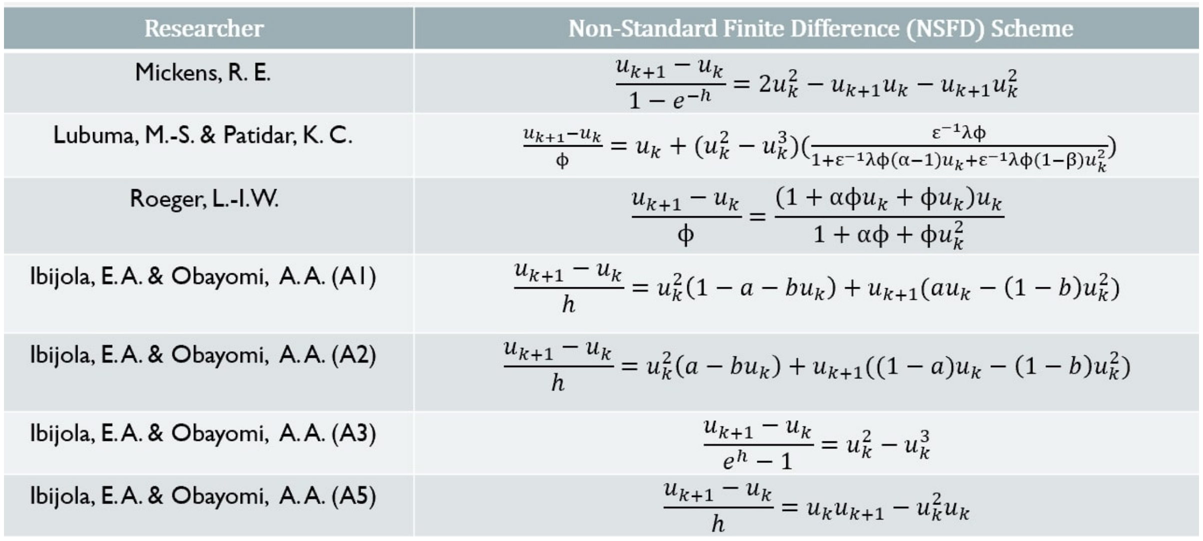


Figure 4: Error Plot of SFD Scheme and our NSFD Unity Approximation Scheme for the Combustion Equation

Table 8 indicates the NSFD scheme constructed for the combustion equation is order 2. Although the 7th order Runge-Kutta method is a numerical approximation and is treated as the true solution for the combustion equation in this research, the objective of constructing an order 2 NSFD scheme utilizing unity approximations was a success. Alongside that, the numerical scheme results in a more preferable approximation than Euler's method, which is order 1, since the error of the NSFD scheme converges to 0 faster than the SFD scheme. Figure 3 graphs the NSFD unity approximation scheme and the SFD scheme in comparison to the 7th order Runge-Kutta method on the interval $(0, 1)$ and with 10 partitions. Understanding that Euler's method is order 1 and the NSFD scheme constructed is order 2 from Table 8, it can be observed that the NSFD scheme creates less error and creates a more accurate approximation of the 7th order Runge-Kutta method for the combustion equation. Figure 4 presents a better visual representation of the error created between both numerical methods using 100 partitions, with the error in Figure 4 being proportional to the area under each curve.

After observing the NSFD unity approximation constructed for the combustion model was second order, other NSFD schemes constructed by other researchers were used to identify its effectiveness. The NSFD schemes that we considered include the following [2] [3] [4] [5]:



Researcher	Non-Standard Finite Difference (NSFD) Scheme
Mickens, R. E.	$\frac{u_{k+1}-u_k}{1-e^{-h}} = 2u_k^2 - u_{k+1}u_k - u_{k+1}u_k^2$
Lubuma, M.-S. & Patidar, K. C.	$\frac{u_{k+1}-u_k}{\phi} = u_k + (u_k^2 - u_k^3)(\frac{\varepsilon^{-1}\lambda\phi}{1+\varepsilon^{-1}\lambda\phi(\alpha-1)u_k+\varepsilon^{-1}\lambda\phi(1-\beta)u_k^2})$
Roeger, L.-I.W.	$\frac{u_{k+1}-u_k}{\phi} = \frac{(1+\alpha\phi u_k+\phi u_k)u_k}{1+\alpha\phi+\phi u_k^2}$
Ibijola, E.A. & Obayomi, A.A. (A1)	$\frac{u_{k+1}-u_k}{h} = u_k^2(1-a-bu_k) + u_{k+1}(au_k - (1-b)u_k^2)$
Ibijola, E.A. & Obayomi, A.A. (A2)	$\frac{u_{k+1}-u_k}{h} = u_k^2(a-bu_k) + u_{k+1}((1-a)u_k - (1-b)u_k^2)$
Ibijola, E.A. & Obayomi, A.A. (A3)	$\frac{u_{k+1}-u_k}{e^h-1} = u_k^2 - u_k^3$
Ibijola, E.A. & Obayomi, A.A. (A5)	$\frac{u_{k+1}-u_k}{h} = u_k u_{k+1} - u_k^2 u_k$

Figure 5: Table of Proposed NSFD Schemes by Researchers for the Combustion Equation

After plotting the schemes against the 7th order Runge-Kutta method in Figure 6, it can be identified that the numerical schemes close are the NSFD scheme proposed by Roeger and the other being our unity approximation scheme constructed for the combustion equation. Closely observing the differences in error between both methods against the 7th order Runge-Kutta method show that our NSFD unity approximation constructed yields less error, by noting that the error in Figure 7 is proportional to the area under each curve.

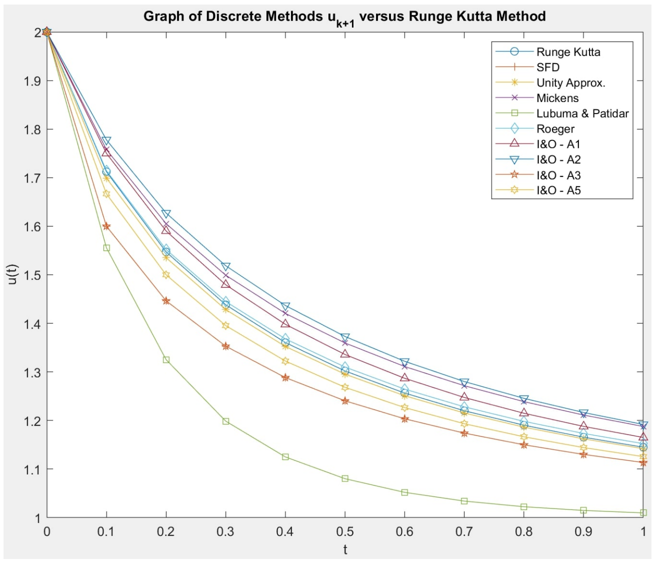


Figure 6: Plot of Proposed NSFD Schemes against 7th Order Runge-Kutta Method for the Combustion Equation

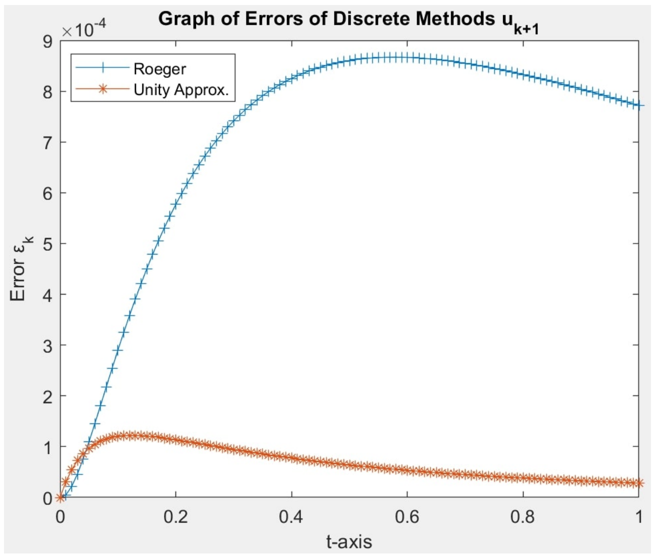


Figure 7: Error Plot of Roeger's NSFD scheme and our NSFD Unity Approximation Scheme for Combustion Equation

After acknowledging the results from Figure 7, our NSFD unity approximation scheme serves to be successful as it is a numerical scheme that creates significantly less error than the NSFD schemes in Table 5. The results from the research emphasize the significance of considering unity approximations as an approach in constructing higher-order NSFD schemes for first-order differential equations.

5 NSFD's for the Bratu IVP

$$u'' = 2e^u, \qquad u(0) = 0, \qquad u'(0) = 0$$

The Bratu IVP is a second ordering differential equation with the initial conditions where $u(0) = u'(0) = 0$. This equation does have an exact analytic solution of $u(t) = -2\ln(\cos(t))$, to verify this we check if the initial conditions hole for $u(t)$ and $u'(t)$. To do this we twice differentiate the assumed true solution and check the initial conditions at each step.

Verification:

$$u(0) = 2\ln(\cos(0)) = 2\ln(1) = 2(0) = 0\checkmark \tag{20}$$

Now differentiate

$$u'(t) = \frac{-2(-\sin(t))}{\cos(t)} = 2\tan(t) \rightarrow u'(0) = 2\tan(0) = 0 \checkmark \tag{21}$$

Next we check if u'' equals the Bratu equation $2e^u$

$$u'' = 2\sec^2(t) = \frac{2}{\cos^2(t)} = 2e^{ln\frac{1}{\cos^2(t)}} = 2e^{-2\ln(\cos(t))} = 2e^u \checkmark$$

Now that we have verified the exact solution we can use numerical methods to approximate the solution because we know we have an exact solution to compare to. Similar to previous equations we start with a Standard Finite Difference:

$$\frac{u_{k+1} - 2u_k + u_{k-1}}{h^2} = 2e^{u_k} \tag{22}$$

We can solve for u_{k+1} with the help of Wolfram Alpha, our scheme will have two unknown steps, u_k and u_{k-1}, this is due to the fact that this is a second-order ODE, unlike the previous ODE's worked on in this research.

$$u_{k+1} = 2u_k - u_{k-1} + 2h^2e^{u_k} \tag{23}$$

Scalar Bratu IVP Schemes

Using the SFD there are various non-standard changes that can be made to begin generating scalar NSFD schemes. Such as identifying a non-standard denominator. To substitute the h^2 in the denominator of equation 22 we start by taking the Taylor series for $-\ln(\cos(h))$ as seen in equation 24:

$$\cos(h) = \sum_{n=0}^{\infty}(-1)^n \frac{h^{2n}}{(2n)!} = 1 - \frac{h^2}{2!} + \frac{h^4}{4!} - \cdots$$

$$\frac{u_{k+1} - 2u_k + u_{k-1}}{-\ln(\cos(h))} = 2e^{u_k} \tag{24}$$

Next, we will create NSFD Scheme E through unity approximations ie:

$$\frac{u_{k+1} - 2u_k + u_{k-1}}{-\ln(\cos(h))} = 2e^{u_k \cdot 1_A} \cdot 1_B \tag{25}$$

However aside from generating NSFD schemes with a non-standard denominator, our focus is on unity approximations, so all other scalar NSFD schemes listed below were created by implementing unity approximations in various parts of the SFD.

Recall SFD : $u_{k+1} = 2u_k - u_{k-1} + 2h^2 e^{u_k}$

Scheme A

$u_k : 2u_k - u_{k-1}$

$$u_{k+1} = 2h^2 e^{2u_k - u_{k-1}} + 2u_k - u_{k-1} \tag{26}$$

Scheme B

$e^{u_k} : e^{u_{k-1}}$

$$u_{k+1} = 2h^2 e^{u_{k-1}} + 2u_k - u_{k-1} \tag{27}$$

Scheme C

$e^{u_k} : \frac{e^{u_k}+e^{u_{k-1}}}{2}$

$$u_{k+1} = 2h^2[\frac{e^{u_k} + e^{u_{k-1}}}{2}] + 2u_k - u_{k-1} \tag{28}$$

Scheme D

$e^{u_k} : 2e^{u_k} - e^{u_{k-1}}$

$$u_{k+1} = 2h^2[2e^{u_k} - e^{u_{k-1}}] + 2u_k - u_{k-1} \tag{29}$$

Scheme E

$h^2 : \frac{1}{2}\ln(\cos(h))$

$$u_{k+1} = -4e^{u_k}\ln(\cos(h)) + 2u_k - u_{k-1} \tag{30}$$

Scheme F

$e^{u_k} : e^{(\frac{u_k+u_{k-1}}{2u_k})u_k}$

$$u_{k+1} = 2h^2 e^{(\frac{u_k+u_{k-1}}{2u_k})u_k} + 2u_k - u_{k-1} \tag{31}$$

Schemes A through F all showed an order of convergence of 1, despite our lengthy exploration of scalar difference equations, we decided to move onto 2D schemes, which allowed us to explore Heun's method which is a known second order converging numerical scheme.

2D Bratu IVP

To examine the 2D Bratu IVP, we transformed the second order equation into a system of first order ODE's, which is further explained below.

$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} u \\ u' \end{bmatrix}$ and $\vec{F}(\vec{x}) = \begin{bmatrix} x_2 \\ -2e^{x_1} \end{bmatrix}$ so $\frac{d}{dt}\begin{bmatrix} u \\ u' \end{bmatrix} = \begin{bmatrix} u' \\ -2e^{u} \end{bmatrix} = \vec{F}(u, u')$

Or in other words, $u = x_1, u' = x_2$ so that $\frac{dx_1}{dt} = x_2$ and $\frac{dx_2}{dt} = -2e^{x_1}$

General case of 2D first-order ordinary differential equations:

$$u' = f_1(u, v), \qquad u(0) = u_0 \tag{32}$$
$$v' = f_2(u, v), \qquad v(0) = v_0 \tag{33}$$

This 2-D system can be written in general as $\dfrac{d\vec{x}}{dt} = \vec{F}(\vec{x}(t))$ where $\vec{x} = \begin{bmatrix} u(t) \\ v(t) \end{bmatrix}$
Since its a vector of first-order ODE's we can use Euler's method to compare our NSFD schemes to.

Now we consider the Bratu IVP written in two first order equations in u and v.

$$u = v, \qquad u_0 = 0$$

$$v = 2e^u, \qquad v_0 = 0$$

Like we have been doing throughout this research we being with examining Euler's Method for this 2D systems.

$$\frac{u_{k+1} - u_k}{h} = v_k$$

$$\frac{v_{k+1} - v_k}{h} = 2e^{u_k}$$

Solve for $u_{k+1} and v_{k+1}$:

$$u_{k+1} = v_k h + u_k \qquad u_0 = 0$$

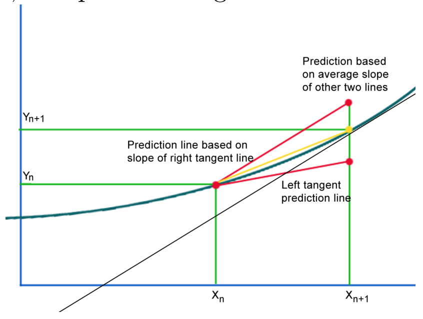
$$v_{k+1} = 2e^{u_k}h + v_k, \qquad v_0 = 0$$

With this being a SFD it maintained its first order convergence. This scheme will be used to compare our other 2D NSFD schemes to see if there is any improvement.

Heun's Method

Our goal is to find a scheme with second order convergence, so we implemented Heun's Method which also uses a 2D system and has a guaranteed second order convergence.

Definition 5.1 (Heun's Method)**.** Heun's method is referred to as a modified Euler's Method, and utilizes an explicit Trapezoid rule, and possesses a guaranteed second order convergence.



$$y_{i+1} = y_i + \frac{h}{2}[f(t_i, y_i) + f(t_{i+1}, y_i + hf(t_i, y_i))]$$

Recall: $t_i + h = t_{i+1}$**.**

One can consider $y_i + hf(t_i, y_i)$ **as** $\tilde{y}$ So, Heun's is also written as:

$$y_{i+1} = y_i + \frac{h}{2}[f(t_i, y_i) + f(t_{i+1}, \tilde{y}_{i+1}]$$

Next we transform Heun's method into a 2D system:

1. $u_{k+1} = u_k + \frac{h}{2}[f_1(t_k, u_k, v_k) + f_1(t_{k+1}, \tilde{u}_{k+1}, \tilde{v}_{k+1})]$
2. $v_{k+1} = v_k + \frac{h}{2}[f_2(t_k, u_k, v_k) + f_2(t_{k+1}, \tilde{u}_{k+1}, \tilde{v}_{k+1})]$
3. $\tilde{u}_{k+1} = u_k + hf_1(t_k, u_k, v_k)$
4. $\tilde{v}_{k+1} = v_k + hf_2(t_k, u_k, v_k)$

For Bratu recall: $f_1(u_k, v_k) = v_k$ and $f_2(u_k, v_k) = 2e^{u_k}$, so that using Heun's Method produces

$$u_{k+1} = u_k + \frac{h}{2}[v_k + \tilde{v}_{k+1}] \rightarrow u_{k+1} = u_k + \frac{h}{2}[v_k + v_k + h2e^{u_k}] \tag{34}$$

$$v_{k+1} = v_k + \frac{h}{2}[2e^{u_k} + 2e^{\tilde{u}_{k+1}}] \rightarrow v_{k+1} = v_k + \frac{h}{2}[2e^{u_k} + 2e^{u_k+hv_k}] \tag{35}$$

Second Order NSFD approximation (modified Heun's) method for Bratu IVP as a scalar:

$$\frac{u_{k+1} - u_k}{h} = v_k + he^{u_k}, \qquad \frac{v_{k+1} - v_k}{h} = e^{u_k} + e^{u_k+hv_k} \tag{36}$$

$$u_{k+1} = u_k + v_k h + h^2 e^{u_k}$$
$$v_{k+1} = v_k + he^{u_k} + he^{u_k+hv_k}$$

Table 8: Error Analysis Between modified Heun's method (1) for Bratu IVP as a scalar

Δt	Δt Ratio	L^∞ Error	Ratio of Errors	Order
0.250000	0.50000	0.062946	$\sim$	$\sim$
0.125000	0.50000	0.005297	0.084152	1.7854
0.062500	0.50000	0.000352	0.066554	1.9546
0.031250	0.50000	0.000022	0.063421	1.9894
0.015625	0.50000	0.000001	0.062724	1.9974

There's a second way of modified Heun's for Bratu as 2D system which showed to be first order.

$$u_{k+1} = u_k + v_k h + h^2 e^{u_k}$$
$$v_{k+1} = v_k + h\left(e^{u_{k+1}} + e^{u_k}\right) \tag{37}$$

After modifying Heun's method and applying NSFD propertied and taking advantage of its second order convergence we were able to discover one other second order 2D system aside from the normal Heun's Method.

5.1 2D to Scalar Bratu IVP

Using what we have discovered from working with a scalar scheme and only finding first order schemes. Along with that looking at our second order 2D scheme our goal now was to find a second order scalar scheme by transforming our 2D into a scalar scheme. Prior to attempting this our expectation was that the order of convergence would remain the same as we transformed from a 2D system to a scalar scheme, thus we assumed with this we would be able to find a second order scalar scheme.

Beginning with our approximation of Heun's $u'' = 2e^u$ through a forward difference quotient.

Recall: $\dfrac{u_{k+1} - u_k}{h} = v_k + he^{u_k}$ and $\dfrac{v_{k+1} - v_k}{h} = e^{u_k} + e^{u_k+hv_k}$

From those difference quotients We know that:

$$v_k = \frac{u_{k+1} - u_k}{h} - he^{u_k}$$

When we increment the iteration by one we get:

$$v_{k+1} = \frac{u_{k+2} - u_{k+1}}{h} - he^{u_{k+1}}$$

Using the forward difference we solve u'' to be:

$$u'' = \frac{v_{k+1} - v_k}{h} = \frac{\frac{u_{k+2}-u_{k+1}}{h} - he^{u_{k+1}} - \left[\frac{u_{k+1}-u_k}{h} - he^{u_k}\right]}{h} = \frac{u_{k+2} - 2u_{k+1} + u_k}{h^2} \tag{38}$$

Apply the right hand side of the equation $\frac{v_{k+1}-v_k}{h}$:

$$\frac{u_{k+2} - 2u_{k+1} + u_k}{h^2} = e^{u_k} + e^{u_k+hv_k}$$

Shift the k value:
$$\frac{u_{k+1}-2u_k+u_{k-1}}{h^2}=e^{u_{k-1}}+e^{u_{k-1}+hv_{k-1}}$$
Approximate $e^{u_k+hv_k}=e^{u_{k+1}}$ and apply algebra to solve for u_{k+1}:
$$u_{k+1}=h^2e^{u_k}+h^2e^{u_k-h^2e^{u_{k-1}}}+2u_k-u_{k-1}$$
Next we allowed u'' to be a centered difference as opposed to a forward difference:
$$u''=\frac{v_{k+1}-v_{k-1}}{2h}, \qquad NOTE: v_{k-1}=\frac{u_k-u_{k-1}}{h}-he^{u_{k-1}}$$
Similarly to before we get:
$$\frac{\frac{u_{k+2}-u_{k+1}}{h}-he^{u_{k+1}}-[\frac{u_k-u_{k-1}}{h}-he^{u_{k-1}}]}{2h}=\frac{u_{k+2}-u_{k+1}-u_k+u_{k-1}}{2h^2}=e^{u_k}+e^{u_k+hv_k}$$
Approximate $e^{u_k+hv_k}=e^{u_{k+1}}$ and solve for u_{k+2} the shift k values to get u_{k+1}:
$$u_{k+2}=u_{k+1}+u_k-u_{k-1}+h^2e^{u_{k+1}}-h^2e^{u_{k-1}}+2h^2e^{u_k}+2h^2e^{u_{k+1}}$$
$$u_{k+1}=u_k+u_{k-1}-u_{k-2}+h^2(e^{u_k}-e^{u_{k-2}})+2h^2(e^{u_{k-1}}+e^{u_k}) \tag{39}$$
Since we require 3 initial conditions let $u_{-1}=u_0=u_1=0$ Now try it without the approximation: $e^{u_k+hv_k}=e^{u_{k+1}}$

$$u_{k+2}=u_{k+1}+u_k-u_{k-1}+h^2e^{u_{k+1}}-h^2e^{u_{k-1}}+2h^2e^{u_k}+2h^2e^{u_k+hv_k}$$
Moving the index down by 1 produces
$$u_{k+1}=u_k+u_{k-1}-u_{k-2}+h^2e^{u_k}-h^2e^{u_{k-2}}+2h^2e^{u_{k-1}}+2h^2e^{u_{k-1}+hv_{k-1}}$$
Recall $v_{k-1}=\frac{u_k-u_{k-1}}{h}-he^{u_{k-1}}$
So, the new equation with red part of the equation replaced with v_{k-1} becomes

$$u_{k+1}=u_k+u_{k-1}-u_{k-2}+h^2e^{u_k}-h^2e^{u_{k-2}}+2h^2e^{u_{k-1}}+2h^2e^{u_{k-1}+h\left(\frac{u_k-u_{k-1}}{h}-he^{u_{k-1}}\right)}$$
which becomes
$$u_{k+1}=u_k+u_{k-1}-u_{k-2}+h^2e^{u_k}-h^2e^{u_{k-2}}+2h^2e^{u_{k-1}}+2h^2e^{u_k-h^2e^{u_{k-1}}}$$

In conclusion of our research into applications of NSFD onto the Bratu IVP, we have found 1 second order 2D Systems however after out numerous attempts of applying non-standard properties to scalar difference equation, and transforming 2D systems into Scalar difference equations we were only able to find schemes of first order convergence.

6 Future Work

Since we were unable to find NSFD schemes of second order convergence for the Bratu IVP we will continue to investigate the cause of this problem. Using the methods and ideas to find

7 Conclusion

Standard and Non-Standard Finite Difference Schemes are utilized to numerically approximate solutions of nonlinear first-order ODEs. We know Non-Standard Finite Difference Schemes can be generated to produce better approximations than Standard Finite Differences. We used non-local terms to approximate the right hand term and generated Unity Approximations to find NSFD Schemes that can produce higher-order schemes. Our Unity approximation NSFD schemes are comparable to other NSFDs that rely on more complicated approaches to approximate differential equations. We were able to create second-order numerical schemes for the Cubic Decay equation, the Bernoulli equation, and the Combustion equation. We constructed an exact NSFD unity approximation scheme for the Cubic Decay equation. Although we did not find a successful scalar NSFD for the Bratu IVP, we found two second-order numerical schemes for the 2D system version of the Bratu IVP.

References

[1] K.F. Gurski. A simple construction of nonstandard finite-difference schemes for small nonlinear systems applied to sir models. *Computers & Mathematics with Applications*, 66(11):2165–2177, 2013. Progress on Difference Equations.

[2] E.A. Ibijola and A.A. Obayomi. A new family of numerical schemes for solving the combustion equation. *Journal of Emerging Trends in Engineering and Applied Sciences*, 3:533–539, 2012.

[3] R.E. Mickens. Nonstandard finite difference schemes for differential equations. *Journal of Difference Equations and Applications*, 8(9):823–847.

[4] J.M.-S. Lubuma K.C. Patidar. Contributions to the theory of non-standard finite difference methods and applications to singular perturbation problems. *Advances in the Applications of Nonstandard Finite Difference Schemes*, pages 513–558.

[5] Lih-Ing W. Roeger. General nonstandard finite-difference schemes for differential equations with three fixed-points. *Computers Mathematics with Applications*, 57(3):379–383, 2008.

[6] Lisha Wang and Lih-Ing W. Roeger. Nonstandard finite difference schemes for a class of generalized convection–diffusion–reaction equations. *Numerical Methods for Partial Differential Equations*, 31, 2015.