Comparison of Runge-Kutta and Newmark Methods

Adam Lukasiewicz

SUNY Polytechnic Institute

A Brief History

Carl Runge was a German mathematician born in 1856. He studied mathematics and physics in Berlin and later became a professor of mathematics at the University of Hanover. Runge made significant contributions to numerical analysis, including the development of the Runge-Kutta method, a family of numerical integration methods used to solve ordinary differential equations.

Martin Kutta was another German mathematician who independently developed a similar family of numerical integration methods around the same time as Runge. The methods developed by Kutta are also now known as Runge-Kutta methods. Kutta was born in 1867 and studied mathematics and physics in Vienna. He later became a professor of mathematics at the University of Brno in what is now the Czech Republic.

Nathan Newmark was an American structural engineer who was born in 1910. He studied civil engineering at the University of Illinois and later became a professor of civil engineering at the same institution. Newmark made significant contributions to the field of structural dynamics, including the development of the Newmark method, a numerical integration method used to solve the equations of motion for large-scale structural systems under dynamic loading conditions.

Comparing The Methods

1. Purpose: The Runge-Kutta method is a general-purpose numerical integration method for solving first-order ordinary differential equations (ODEs), while the Newmark method is a specific numerical method used for solving second-order ODEs of motion in structural dynamics and related fields. While both methods can be used to solve differential equations, they are often applied to different types of problems.

2. Accuracy: Both methods are known for their accuracy, but the Runge-Kutta method is generally more accurate than the Newmark method due to its higher order of approximation. The Runge-Kutta method can achieve a high level of accuracy by using higher-order polynomials to approximate the solution at each time step, but this comes at the cost of increased computational complexity.

The Newmark method is a second-order method, meaning that it uses a quadratic approximation to the solution at each time step. This results in a slightly lower level of accuracy than the Runge-Kutta method but allows for faster computation times.

3. Stiffness: Both methods can handle stiff equations, but they may require different approaches to ensure numerical stability. The Runge-Kutta method can handle stiff equations by using smaller step sizes, which can increase the computational time. The Newmark method is particularly useful for problems where the stiffness matrix and mass matrix are constant.

4. Computation: The Runge-Kutta method requires more computational resources, as it involves solving multiple differential equations at each time step. The computational complexity of the Runge-Kutta method increases with the order of the method used. In contrast, the Newmark method requires less computational resources, as it involves solving a single set of equations at each time step.

5. Stability: The Runge-Kutta method is conditionally stable, meaning that the time step size must be sufficiently small to ensure stability. The Runge-Kutta method can become unstable if the time step is too large, leading to numerical errors and instability in the solution.

The Newmark method is unconditionally stable, meaning that it can handle larger time step sizes without sacrificing stability. This makes the Newmark method particularly useful for problems with long time scales or where the computational cost of small time steps is prohibitive.

My Experiment

To create a fair comparison which was easy to understand, I used python to define a system of masses and springs of magnitude "n". I also included a changeable damping or friction factor. Then, I programmed the two methods to solve for the positions and velocities of the masses. Additionally, I included the energy equation of the spring-mass system under each method to give additional insight on how these methods compare. I made the iterations "N", and step size "h", definable by the user. Spring-mass systems are plentiful in real life and are a good way to visualize the functionality of these solver methods.

```python
# define Mass Spring Damper
n = 2
M = np.identity(n)
S = np.identity(n)
D = 0.4*np.identity(n)

# initial conditions u0 v0 a0
u0 = np.ones(n)
v0 = np.zeros(n)
a0 = np.zeros(n)

# acceleration function
def acceleration(u,v,M,S,D):
    a = - M.T @ S @ u - M.T @ D @ v
```

Figure 1. Defining the Mass, Spring, and Damper matrices. Setting the initial position (u0), initial velocity (v0), and initial acceleration (a0). Defining the acceleration equation.

```python
# runge kutta method
def ruku(u,v,dt,M,S,D):
    v1 = v
    a1 = acceleration(u,v,M,S,D)
    v2 = v + 0.5 * dt * a1
    a2 = acceleration(u+0.5*dt*v1, v+0.5*dt*a1, M,S,D)
    v3 = v + 0.5 * dt * a2
    a3 = acceleration(u+0.5*dt*v2, v+0.5*dt*a2, M,S,D)
    v4 = v + dt * a3
    a4 = acceleration(u+dt*v3, v+dt*a3, M,S,D)

    U = u + dt * (v1 + 2 * v2 + 2 * v3 + v4) / 6
    V = v + dt * (a1 + 2 * a2 + 2 * a3 + a4) / 6

    Ed_rk = - V.T @ D @ V
    E_rk = 0.5 * (V.T @ M @ V + U.T @ S @ U + V.T @ D @ V)


    return U, V, E_rk
```

Figure 2. The Runge-Kutta solver with energy equation.

```python
# newmark method
def newmark(u,v,a,dt,M,S,D):
    # compute ustar and vstar
    ustar = u + v * h + a * (h**2 / 4)
    vstar = v + a * (h / 2)

    # solve for a_jplus1
    b = acceleration(u,v,M,S,D)
    A = (M + (S) * (h**2 / 4) + D * h / 2)
    a_jplus1 = nla.solve(A,b)

    # compute u_jplus1 and v_jplus1
    u_jplus1 = ustar + a_jplus1 * (h**2 / 4)
    v_jplus1 = vstar + a_jplus1 * (h / 2)

    Ed_n = (- v.T @ D @ v)
    E_n = 0.5 * (v.T @ M @ v + u.T @ S @ u + v.T @ D @ v)


    return u_jplus1, v_jplus1, a_jplus1, E_n
```

Figure 3. The Newmark solver with energy equation.

What I found

Three graphs are generated from my python code every time it is ran. The first graph pertains to the Newmark method. The second graph pertains to the Runge-Kutta method. The third graph is a graph of the energy data generated by both the Newmark and Runge-Kutta methods.

In the Newmark graph, the red line shows the position, and the blue line shows the velocity of one of the masses in the system. In the Runge-Kutta graph, the yellow line shows the position, and the green line shows the velocity of the same mass.

In the energy graph, the purple line is the system energy according to the Newmark solver, and the cyan line is the system energy according to the Runge-Kutta solver.

Friction VS No Friction

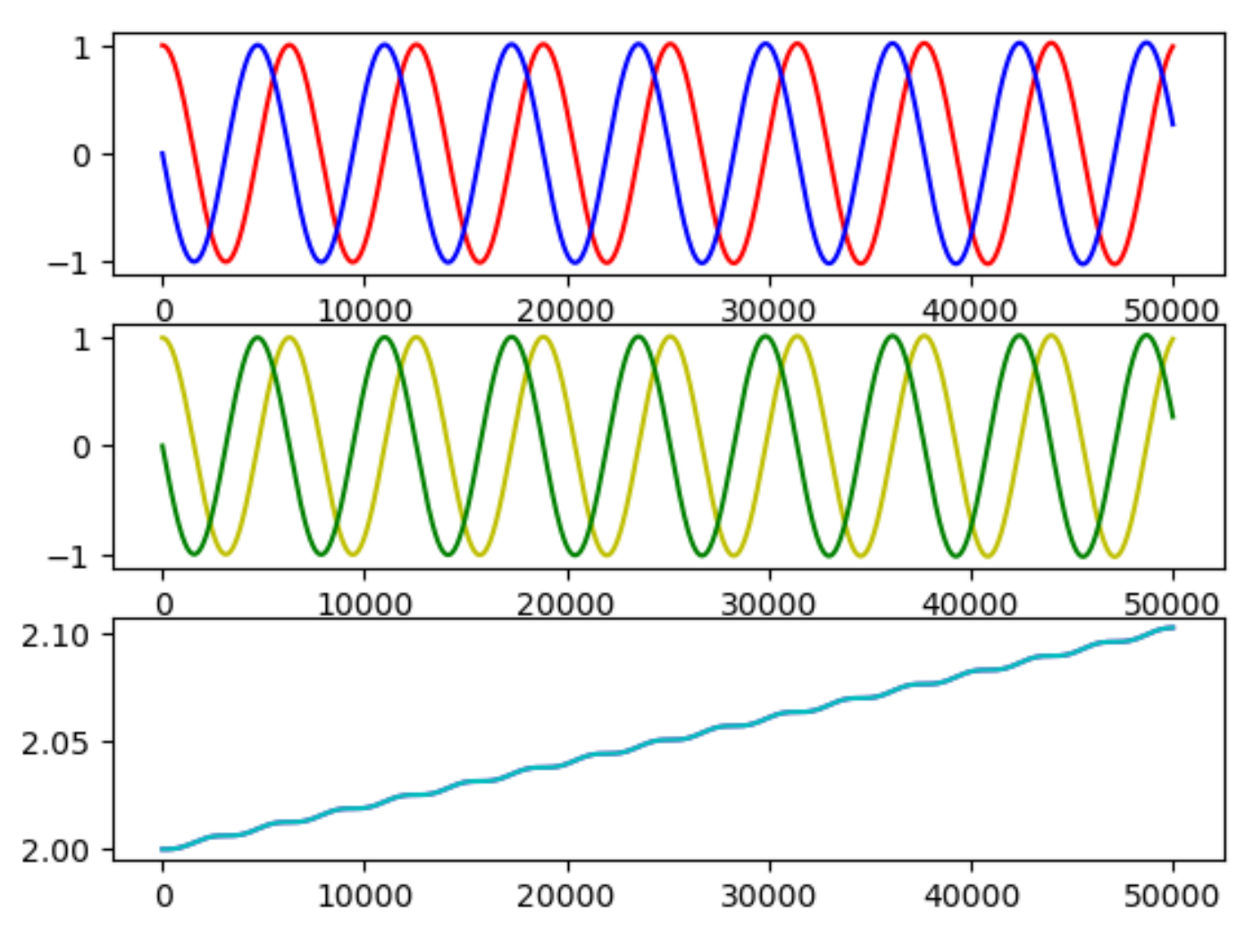

Figure 4. Without Friction

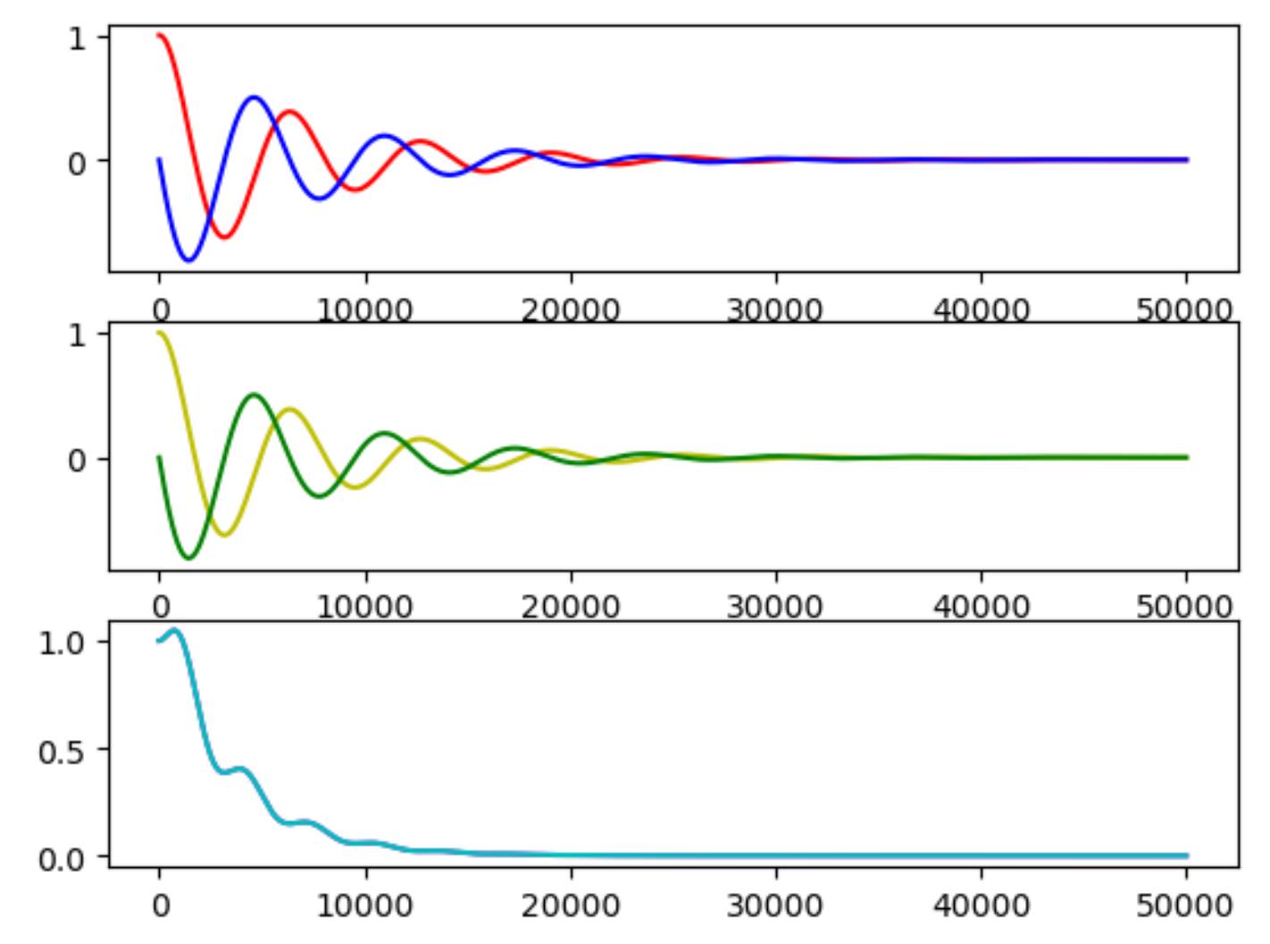

Figure 5. With Friction

With a step size of 1e-4 and 50,000 iterations, both solvers worked very well. The position and velocity of the block were consistent between the two methods, and the energy of the system was as well. I found it very worrying at first to see the energy of the system grow over time. "Perhaps an issue with my code" I thought. It was not. Don't let that steep incline fool you, the energy grows less than 0.1 over 50,000 iterations. The reason behind this is the error of the solver as the "h" value is always overshooting.

Finding The Breaking Point

I had proven the two methods can give identical results. Next I set out to find out where they differ. I tried every possible combination of initial values, mass/spring/damper matrices, and increased the iteration count until my computer could not handle it. None of those evoked a noticeable difference. What did work was increasing the "h" step size value. Thus far I had been using 1e-3 at most. At 1e-1, the Runge-Kutta began showing inaccurate values for system energy. By 0.4 the difference between Newmark and Runge-Kutta was immense. This was without tampering with any other variables.

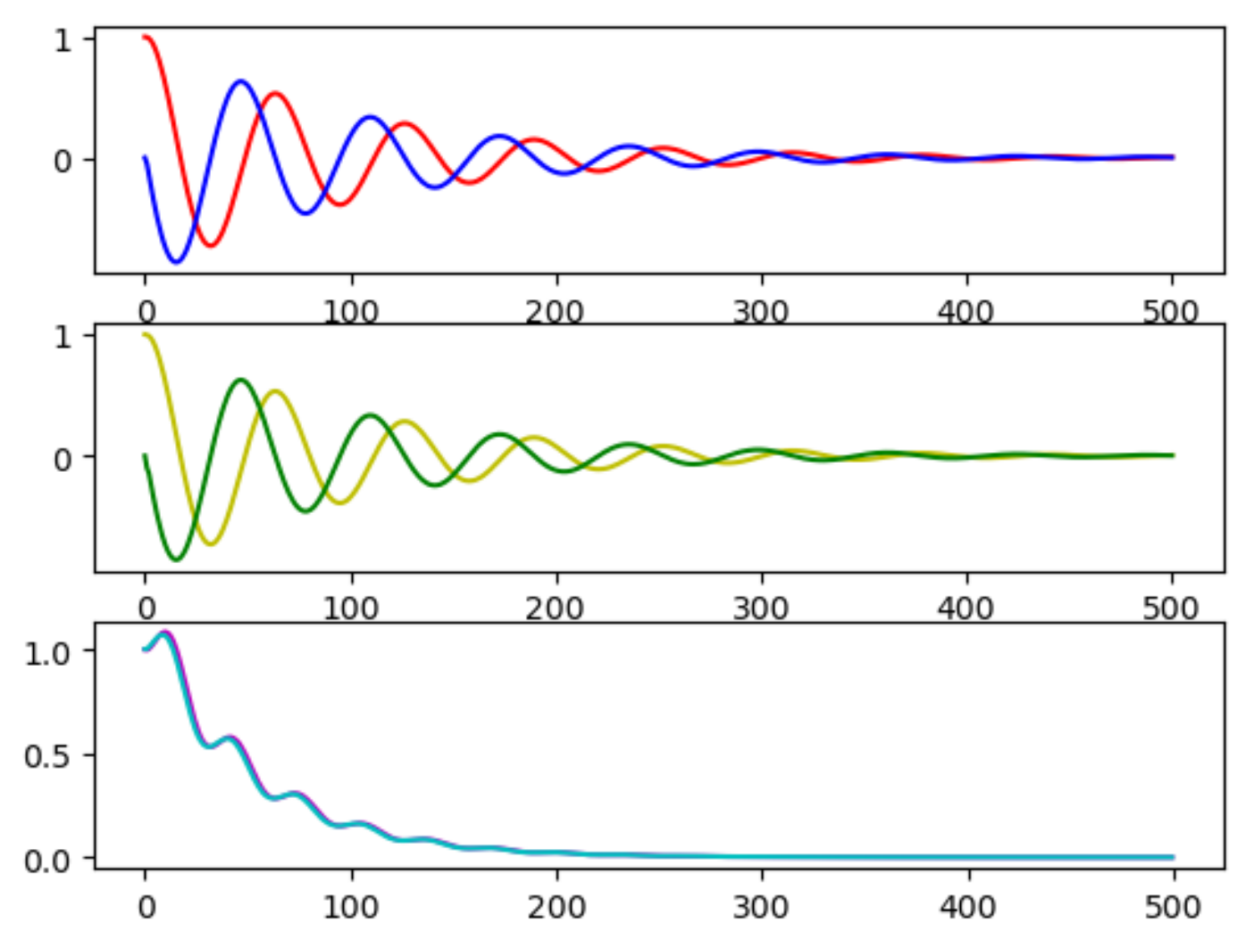

Figure 6. 0.1 Step Size

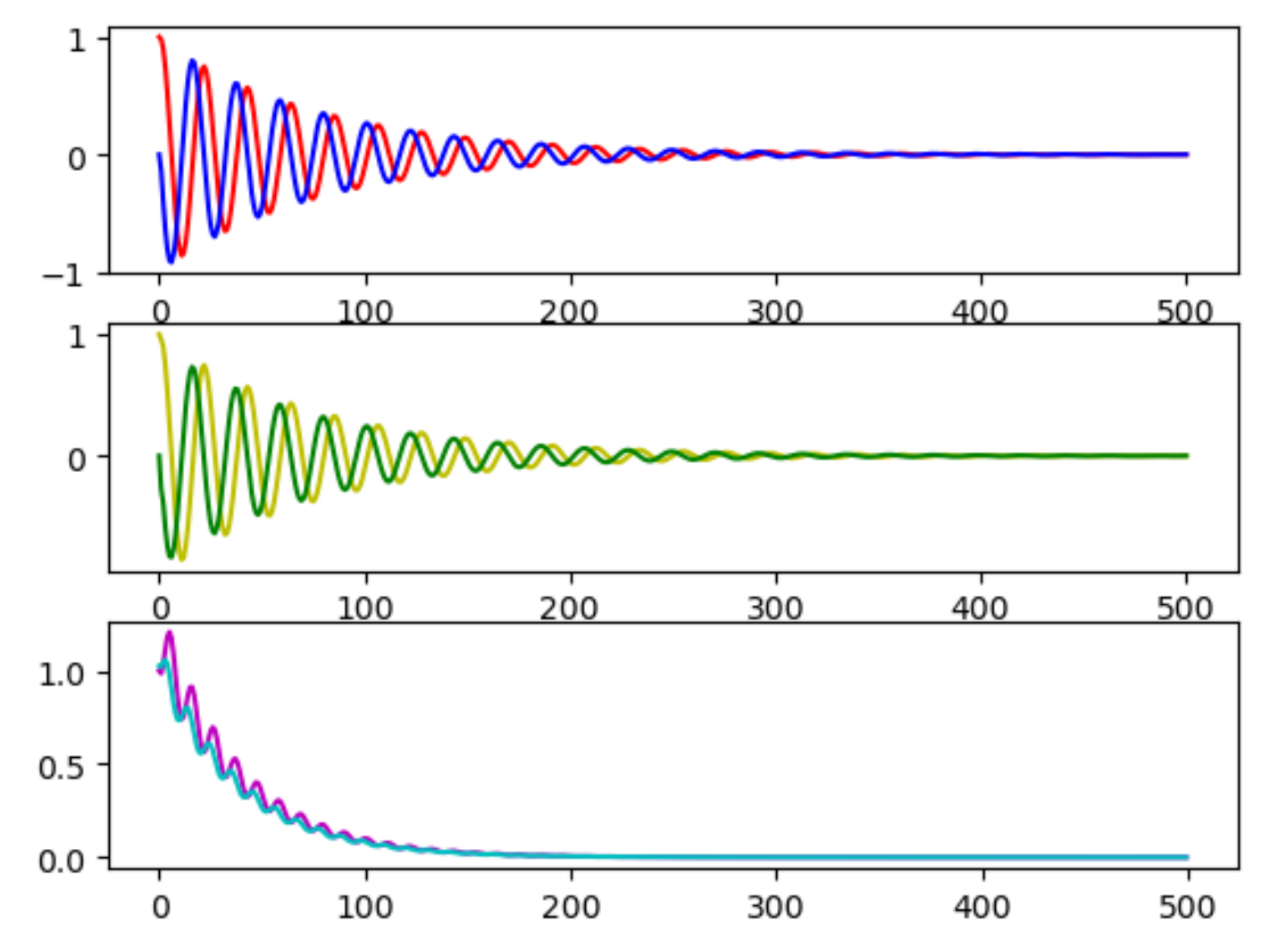

Figure 7. 0.4 Step Size

Thank you Dr. Andrea Dziubek and TA Chawn Neal for all your help and support. You made MAT 460 a memorable and positive experience.

lukasiak@sunypoly.edu