Homework 3 Write-up

Sarah Vaughn

1 question 1

The unittest of the Matrix class from the last homework assignment is located in the python file called test_Matrix_class.py. The objective was to test the matrix class using some numpy matrix operations. Unfortunately, most of my matrix class failed their test but I believe that this is because of the Matrix class itself and not the unittest.

2 question 2

Using the data given in the text file for the value of the matrix A_{ul}, I could calculate the values for the matrices B_{ul}, B_{lu}, and J using the following equation:

$$B_{u_l} = \frac{c^2}{2hv^3} \cdot A_{u_l} \tag{1}$$

$$B_{l_u} = \frac{g_u}{g_l} \cdot B_{u_l} \tag{2}$$

$$J = \frac{2hv^3}{c^2} \cdot \frac{1}{c^{hv/kT} - 1} \tag{3}$$

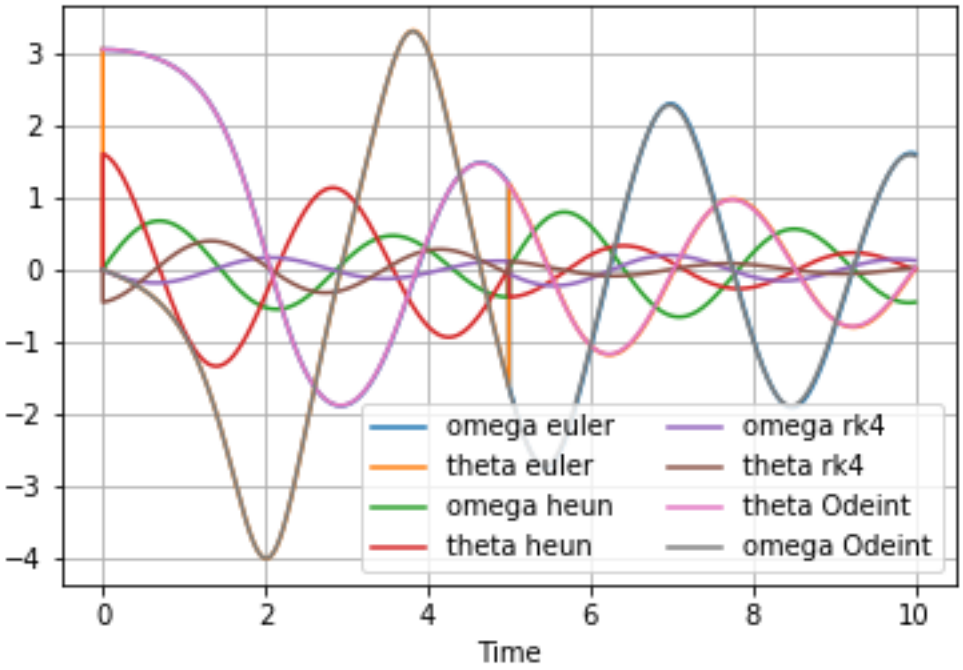
where $g_n = 2n^2$ and Boltzmann constant k = 1.3807e-16 $[cm^2 \cdot g \cdot s^{-2} \cdot K^{-1}]$, Plancks constant is h = 6.626e-27$[cm^2 * g * s^-1]$, and the speed of light is c = 3e10 $[cm \cdot s^{-1}]$. Then the values of these matrices will correspond to a new matrix A where all the upper triangle values (or when $u > l$) are $-(A_{ul} + (B_{ul} \cdot J))$ and the values of lower triangle of the matrix (or when $u < l$) are $-(B_{ul} \cdot J)$ and finally all of the diagonal values in the new matrix will be -(sum of all values in the particular column). This gives us the equation $A \cdot n = b$ where A is the new matrix and n is a list of number densities and b is a 1x8 matrix of zero values. The n's are the thing we want to find to see how they differ as a function of time. Some simple matrix manipulation will find these number densities.multiplying the inverse of matrix A by matrix b should result in the solution to the n values: $n = b \cdot A^{-1}$. Unfortunately, the code that I wrote doesn't run like it should. I think that it has to do with my matrix class from before. In particular, I think its something to do with the way the matrices are built and the values are filled in. The code doesn't produce the values it should to plot the number densities versus time.

3 question 3

ODE package using Euler's method, Heun's method and 4th order Runga-Kutta to solve the given function.

4 question 4

This code is testing the ODE package from question 3 with the scipy package called odeint. The example that is used for this test is from the documentation for the scipy package.



The resulting plots should all overlap if they are fitting the functions the same as the exact solution or in this case, the Odeint theta and omega. Something isn't right about my ODE solver because they aren't lining up with the odeint solver. It seems like there is something in the Heuns method and the RK4 method that have some factor that is effecting the amplitude of the equations that are being processed. It could also be the way I defined the functions that were processed but they were initialized nearly the same way.

5 question 5

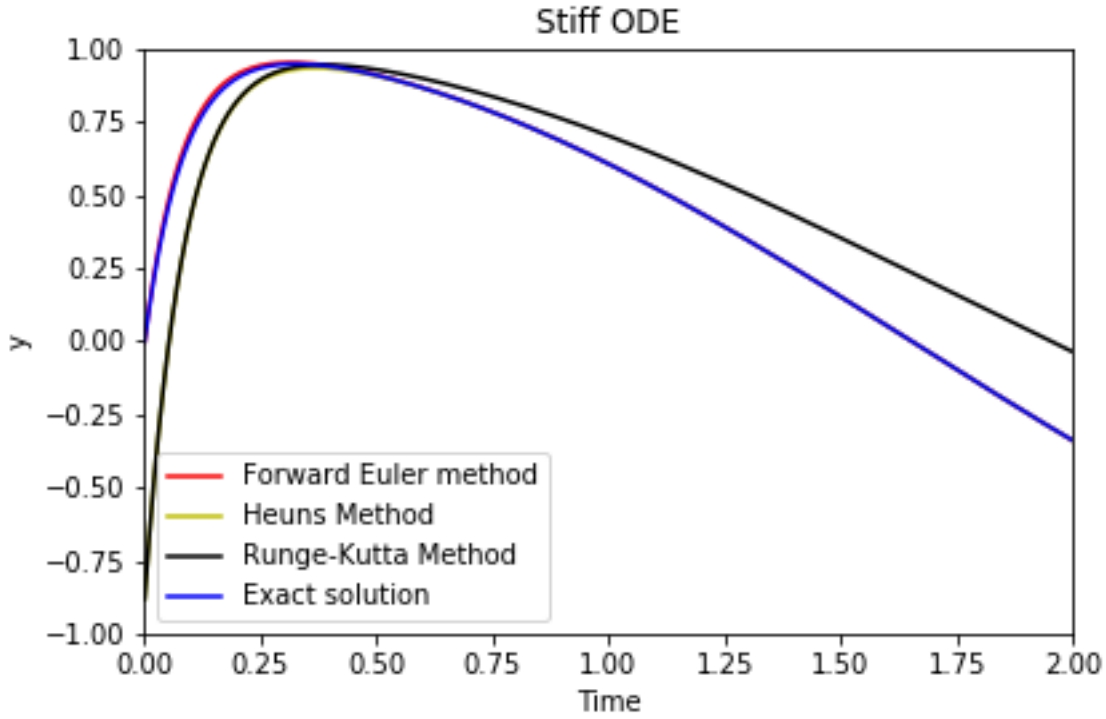
The next test of the ODE solver was solving the stiff ODE equation:

$$\frac{dy}{dt} = -\lambda(y - cos(t)) \tag{4}$$

and the exact solution is:

$$y(t) = -\frac{\lambda^2}{1+\lambda^2}e^{-\lambda t} + \frac{\lambda}{1+\lambda^2}sin(t) + \frac{\lambda^2}{1+\lambda^2}cos(t) \tag{5}$$

After applying my ODE solver to the stiff ODE and plotting it, the resulting curve is:



Again, the Heuns method and the RK4 method don't match up with the exact very well and seem to be off from the start. It should start at 0.00 and it is initialized in the code but it definitely seems like a problem with the ODE solver but I don't see the difference in how I wrote the these versus how I wrote the Euler.