18.330 Pset 3

Kathleen Brandes

March 7, 2019

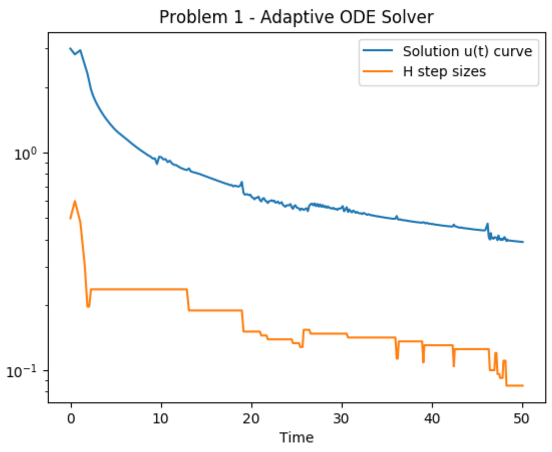
1 Problem 1

1.1 Part A

My implementation of the adaptive ode solver using improved euler and rk4 estimations can be found in problem1.jl, which is attached on stellar.

1.2 Part B

For the test function $\frac{du}{dt} = cos(tu(t)^2)$ when $u(0) = 3$ and $t_0 = 0$, $t_f = 50$. I plotted the time steps on the x-axis with the h steps on a logarithmic y-axis and the estimated results of $u(t)$ at each time step on a linear y-axis.

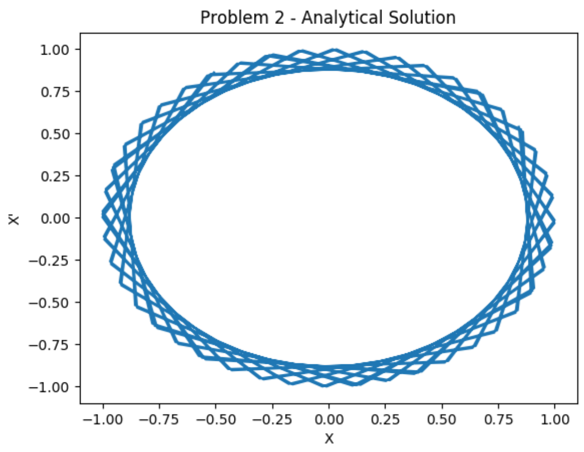


2 Problem 2

The code for problem 2 is attached to stellar in the file problem2.jl.

2.1 Part A

The initial ode equation is $x''(t) - \mu(1 - x(t)^2)x'(t) + x(t) = f(t)$. When $\mu = 0$ and $f(t) = 0$, then the equation can be simplified to $x''(t) = -x(t)$ with initial values of $x(0) = 0$ and $x'(0) = 1$, which has an exact analytical solution of $x(t) = \texttt{sin}(t)$. In phase space $(x(t), x'(t))$, the resulting plot of this ode is:



2.2 Part B

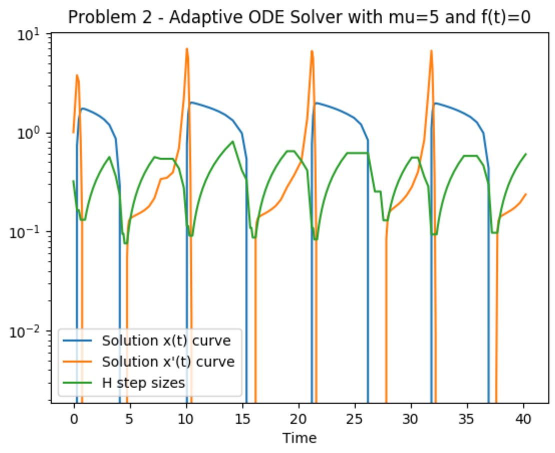
This problem uses the same second order ode $x''(t) - \mu(1 - x(t)^2)x'(t) + x(t) = f(t)$, but with a non-zero $\mu = 5$ value instead. It has the same initial conditions of $x(0) = 0$ and $x'(0) = 1$ and $f(t) = 0$ as part a. To solve the second order ode with the numerical method adaptive ode solver from problem 1, I converted it into a system of equations that were both first order ode's. I set $x_1 = x(t)$ and $x_2 = x'(t)$ such that $x_1' = x'(t) = x_2$ and $x_2' = x''(t) = \mu(1 - x_1^2)x_2 - x_1$. This created a system of first order ode's:

$$\begin{bmatrix} x_1'(t) \\ x_2'(t) \end{bmatrix} = \begin{bmatrix} x_2 \\ 5(1 - x_1^2)x_2 - x_1 \end{bmatrix}$$

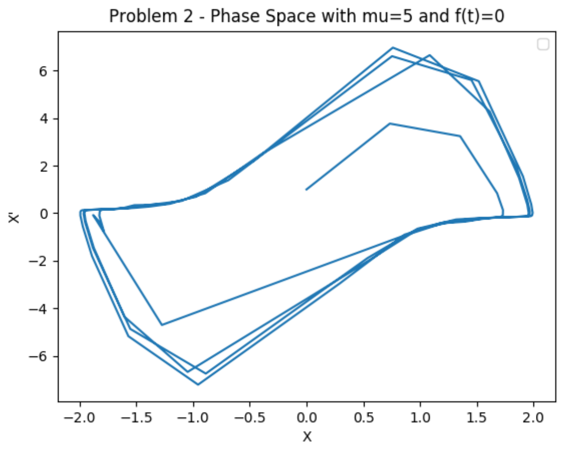
And the initial conditions were:

$$\begin{bmatrix} x_1(0) \\ x_2(0) \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

This resulting system of equations can be used in the adaptive ode solver from problem 1. The resulting plot was:



Here is the same results for $x(t)$ and $x'(t)$ plotted in phase space:



Having a non-zero μ, and in turn a first derivative $x'(t)$, the resulting phase space plot takes longer to converge on a consistent trajectory than the exact solution did. As well, the exact solution's trajectory is more of an elipse whereas this solution has its own form.

2.3 Part C

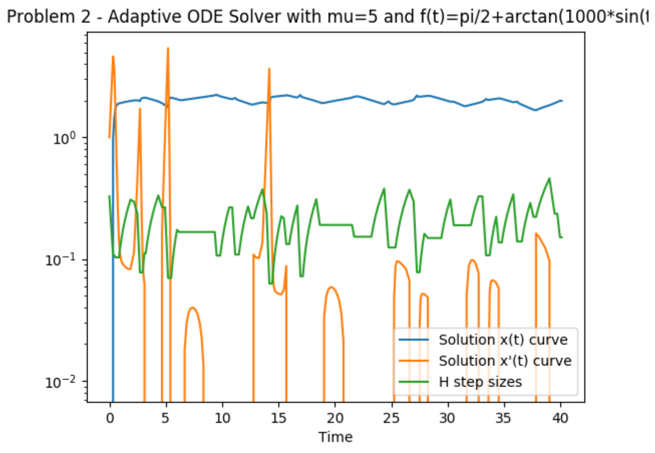
This part again uses the same second order ode, however now has a non-zero $f(t)$ function. The forcing function was $f(t) = \frac{\pi}{2} + \texttt{arctan}(1000\texttt{sin}(t))$ and $\mu = 5$, and the initial conditions of $x(0) = 0$ and $x'(0) = 1$ are the same. Using the same system of equations approach as part b, we get:

$$\begin{bmatrix} x_1'(t) \\ x_2'(t) \end{bmatrix} = \begin{bmatrix} x_2 \\ 5(1-x_1^2)x_2 - x_1 + \frac{\pi}{2} + \texttt{arctan}(1000\texttt{sin}(t)) \end{bmatrix}$$

And the initial conditions were:

$$\begin{bmatrix} x_1(0) \\ x_2(0) \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

This system of equations was used in the adaptive ode solver to get the resulting plot:



3 Problem 3

3.1 Part A

The integral is $I_n(x) = \frac{x^{n+1}e^x}{n!}\int_0^1 u^n e^{-ux}du$.

Evaluated for $n = 0$:

$$I_0(x) = \frac{x^{0+1}e^x}{0!}\int_0^1 u^0 e^{-ux}du = xe^x\int_0^1 e^{-ux}du = xe^x(-\frac{1}{x}e^{-ux}|_0^1)$$
$$= xe^x(\frac{1-e^{-x}}{x}) = e^x - 1$$

Evaluated for $n = 1$:
For u-substitution, let $s = u$ (so $ds = du$), and $dv = \int e^{-ux}du$ (so $v = \frac{-e^{-ux}}{x}$). Then we can solve the integral.

$$\begin{gathered}I_1(x) = x^2e^x \int_0^1 ue^{-ux}du = x^2e^x(sv - \int vds) = x^2e^x(u\frac{-e^{-ux}}{x}|_0^1 - \int_0^1 \frac{-e^{-ux}}{x}du) \\ = x^2e^x(\frac{-e^{-x}}{x} - \frac{-e^{-ux}}{x^2}|_0^1) = x^2e^x(\frac{-e^{-x}}{x} - (\frac{e^{-x}-1}{x^2})) \\ = x^2e^x(\frac{-xe^{-x}-e^{-x}+1}{x^2}) = -x - 1 + e^x\end{gathered}$$

Evaluated for $n = 2$: For u-substitution, let $s = u^2$ (so $ds = 2udu$), and $dv = \int e^{-ux}du$ (so $v = \frac{-e^{-ux}}{x}$). Then we can solve the integral.

$$\begin{gathered}I_2(x) = \frac{x^3e^x}{2}\int_0^1 u^2e^{-ux}du = x^2e^x(sv - \int vds) \\ = \frac{x^3e^x}{2}(u^2\frac{-e^{-ux}}{x}|_0^1 - \int_0^1 \frac{-e^{-ux}}{x}2udu) \\ = \frac{x^3e^x}{2}(\frac{-e^{-x}}{x} - \frac{-2}{x}\int_0^1 ue^{-ux}du) \\ = \frac{x^3e^x}{2}(\frac{-e^{-x}}{x} - \frac{-2}{x}(\frac{-xe^{-x}-e^{-x}+1}{x^2})) \\ = \frac{x^3e^x}{2}(\frac{-x^2e^{-x}}{x^3} - \frac{2xe^{-x}+2e^{-x}-2}{x^3}) \\ = -\frac{1}{2}x^2 - x - 1 + e^x\end{gathered}$$

Evaluated for $n = 3$: For u-substitution, let $s = u^3$ (so $ds = 3u^2du$), and $dv = \int e^{-ux}du$ (so $v = \frac{-e^{-ux}}{x}$). Then we can solve the integral.

$$\begin{gathered}I_3(x) = \frac{x^4e^x}{6}\int_0^1 u^3e^{-ux}du = \frac{x^4e^x}{6}(sv - \int vds) \\ = \frac{x^4e^x}{6}(u^3\frac{-e^{-ux}}{x}|_0^1 - \int_0^1 \frac{-e^{-ux}}{x}3u^2du) \\ = \frac{x^4e^x}{6}(\frac{-e^{-x}}{x} - \frac{-3}{x}\int_0^1 ue^{-ux}du) \\ = \frac{x^4e^x}{6}(\frac{-e^{-x}}{x} - \frac{-3}{x}\int_0^1 ue^{-ux}du) \\ = \frac{x^4e^x}{6}(\frac{-e^{-x}}{x} - \frac{-3}{x}(\frac{-x^2e^{-x}-2xe^{-x}-2e^{-x}+2}{x^3}) \\ = \frac{x^4e^x}{6}(\frac{-e^{-x}}{x} - (\frac{3x^2e^{-x}+6xe^{-x}+6e^{-x}-6}{x^4}) \\ = \frac{x^4e^x}{6}(\frac{-x^3e^{-x}-3x^2e^{-x}-6xe^{-x}-6e^{-x}+6}{x^4}) \\ = \frac{-1}{6}x^3 - \frac{1}{2}x^2 - x - 1 + e^x\end{gathered}$$

3.2 Part B

The code to evaluate the analytically solved integrals I_n for $n = 0, 1, 2, 3$ is in the file called problem3.jl attached on stellar. The exact results for the analytical solutions at $x = 10^{-4}$ found by the code were:

$$\begin{gathered}I_0(10^{-4}) = 0.0001000050001667141 \\ I_1(10^{-4}) = 5.000166725110944e - 9 \\ I_2(10^{-4}) = 1.667554982986985e - 13 \\ I_3(10^{-4}) = 8.883163203173967e - 17\end{gathered}$$

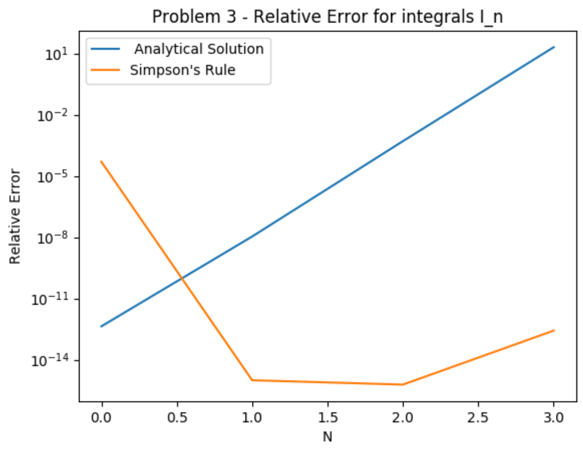
3.3 Part C

The code to evaluate the integrals using Simpson's Rule with $N = 1000$ when $x = 10^{-4}$ is in the file problem3.jl attached on stellar. The resulting values of estimating each integral for $n = 0, 1, 2, 3$ found by the code were:

$$I_0(10^{-4}) = 0.00010001000050001679$$
$$I_1(10^{-4}) = 5.000166670833422e - 9$$
$$I_2(10^{-4}) = 1.6667083341666796e - 13$$
$$I_3(10^{-4}) = 4.166750001388913e - 18$$

3.4 Part D

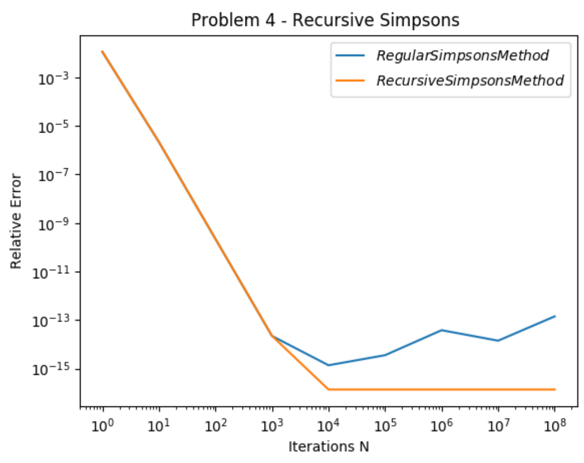
This is a plot comparing the relative errors of the analytical solutions and the simpson's rule solutions for $I_n(10^{-4})$ when $n = 0, 1, 2, 3$:



The simpson's rule estimates tend to have less relative error to compute then the exact analytical solution. This is because the analytical solution is susceptible to catastrophic loss of precision because it takes a single time step and subtracts a very small numbers, such as $(10^{-4})^n$, from relatively large numbers, such as 1, $e^{10^{-4}}$. The simpson's rule prevents this since each iteration only involves addition.

4 Problem 4

My implementation of the recursive simpson's method can be found in problem4.jl which is attached on stellar. I have evaluated the two methods on a test integral of $\int_1^2 \texttt{log}^3(x)dx$ and plotted each method's relative error when compared to the exact solution.



We can see that the recursive simpson's method works better (maintains a very low relative error) even for high values of N, whereas the regular simpson's method levels off at a slightly higher error for large values of N. The recursive simpson's method returns lower error for high values of N since it will continue to subdivide N and recall simpson's method which allows less error to accumulate since simpson's method is only ever being called for small subsets of N.

5 Problem 5

5.1 Part A

The calculation is $S_N = \sum_{n=1}^{N} s_n$ in exact terms using kahan summation. In general at time step k, we have

$$y_k = s_k$$

$$t_k = S_{k-1} + y_k = \sum_{i=1}^{k-1} s_i + s_k = s_1 + ... + s_k$$

$$c_k = (t_k - S_{k-1}) - y_k = (S_{k-1} + y_k - S_{k-1}) - y_k = 0$$

$$S_k = t_k = s_1 + ... + s_k$$

So when $k = N$ this will exactly compute the sum S_N.

5.2 Part B

5.2.1 Part i

The exact sum is $10000.0 + 4.14159 + 1.71828 = 10005.85987$, which when rounded to six digits will be 10005.9.

5.2.2 Part ii

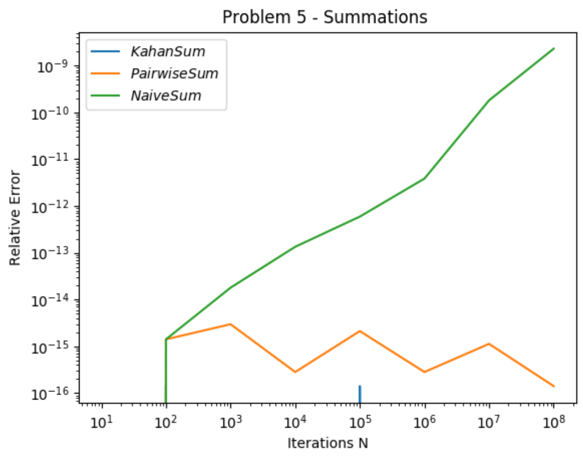
The rolling sum is computed as such (with rounding to six significant digits after each pair of sums):
$s_1 + s_2 = 10000.0 + 4.14159 = 10004.14159 = 10004.1$
$s_{1,2} + s_3 = 10004.1 + 1.71828 = 10005.81828 = 10005.8$
This value is different from the exact value computed in part i.

5.2.3 Part iii

The kahan summation computes the values in two iterations as such:
Iteration 1:
$y_1 = 4.14159 - 0$
$t_1 = 10000.0 + 4.14159 = 10004.1$
$c_1 = (10004.1 - 10000.0) - 4.14159 = 4.1 - 4.14159 = -.04159$
$S_1 = 10004.1$
Iteration 2:
$y_2 = 1.71828 + .04159 = 1.715987$
$t_2 = 10004.1 + 1.75987 = 10005.85987 = 10005.9$ $c_2 = (10005.9 - 10004.1) - 1.715987 = 1.8 - 1.75987 = .04013$
$S_2 = 10005.9$
The value of the sum of the three digits using the kahan sum exactly matches the exact solution from part i for six digits of precision. The kahan sum calculates the sum more accurately because it avoids loss of precision due to rounding errors since it keeps a running compensation sum of the lower digit decimals in the c_i term since these are not incorporated yet in the total sum in t_i/S_n.

5.3 Part C

My implementation of the kahan sum in julia can be found in problem5.jl which is attached on stellar. We test this sum on $P(N) = \sum_{n=1}^{N} \frac{\pi}{N}$ which has an exact solution of π. This plot shows the results of the kahan sum, a naive sum from problem set 1, and the pairwise sum from lecture for the test summation $P(N)$.



The kahan sum is the most accurate with a relative error at machine epsilon for all values of $N \in [10, 10^9]$.