Homework Checklist

1. Did you read every problem in its entirety?
   Yes
2. Dud you include the code you used to produce the results and plots?
   Yes
3. Did you write in complete English sentences including correct spelling and punctuation?
   Yes
4. Do your plots have readable labels on the axes?
   Yes
5. Did you use appropriate line styles in your plots?
   Yes
6. Are the lines visible on the plots you submitted?
   Yes
7. Do your plots have appropriate captions?
   Yes
8. Do your plots have appropriate titles?
   Yes
9. Do your plots have appropriate legends?
   Yes
10. Is the scale of the axes’ correct (plain or log10)?
    Yes
11. If you are plotting errors, are all the errors non-negative?
    Yes
12. If you have any handwritten notes, are these notes legible?
    I have no handwritten notes
13. Do your tables have appropriate column and/or row headings?
    Yes
14. Did you properly cite any sources you used in completing the homework?
    Yes
15. Did you properly identify and acknowledge collaborations with classmates?
    Yes

Signed, *Tanner O’Rourke*

Note: All code implementations are found in the appendix.

Question 1

Consider the differential equation

y'(t) + y(t) = sin(t) t ← [0, 8]

y(0) = 0

1a. Verify the solution is: y(t) = $\frac{1}{2}(e^{-t} + \sin(t) - \cos(t))$ (exp(-x) = e^{-x})

Differentiate:

$\Longrightarrow$ y'(t) = $\frac{1}{2}(-e^{-t} + \cos(t) + \sin(t))$

Plug y'(t) and y(t) into the differential equation:

$\Longrightarrow$ y'(t) + y(t) = sin(t)

$\Longrightarrow \frac{1}{2}(-e^{-t} + \cos(t) + \sin(t)) + \frac{1}{2}(e^{-t} + \sin(t) - \cos(t))$ = sin(t)

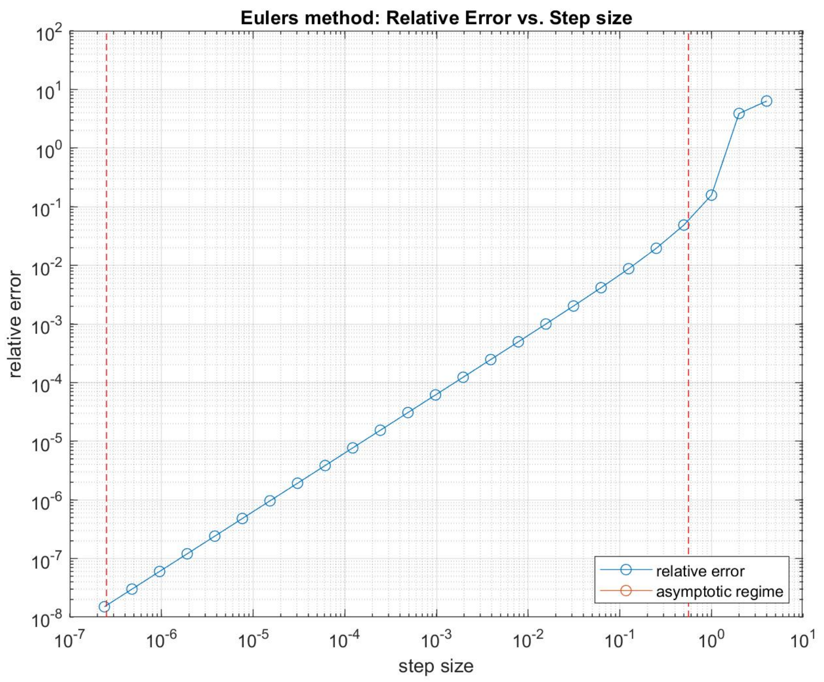
$\Longrightarrow -\frac{1}{2}e^{-t} + \frac{1}{2}\cos(t) + \frac{1}{2}\sin(t) + \frac{1}{2}e^{-t} + \frac{1}{2}\sin(t) - \frac{1}{2}\cos(t)$ = sin(t)

$\Longrightarrow \frac{1}{2}\sin(t) + \frac{1}{2}\sin(t)$ = sin(t)

$\Longrightarrow \sin(t)$ = sin(t)

Because both sides of the equation are equal, the solution is correct.

1b. Implement forward Euler to estimate the solution to the differential equation using step sizes $h = 2^{-k}$ with k = 1,....15. For each step, compute relative error at final time t = 8. Plot error vs. step size on a log-log scale. Identify the asymptotic regime and report observed convergence.



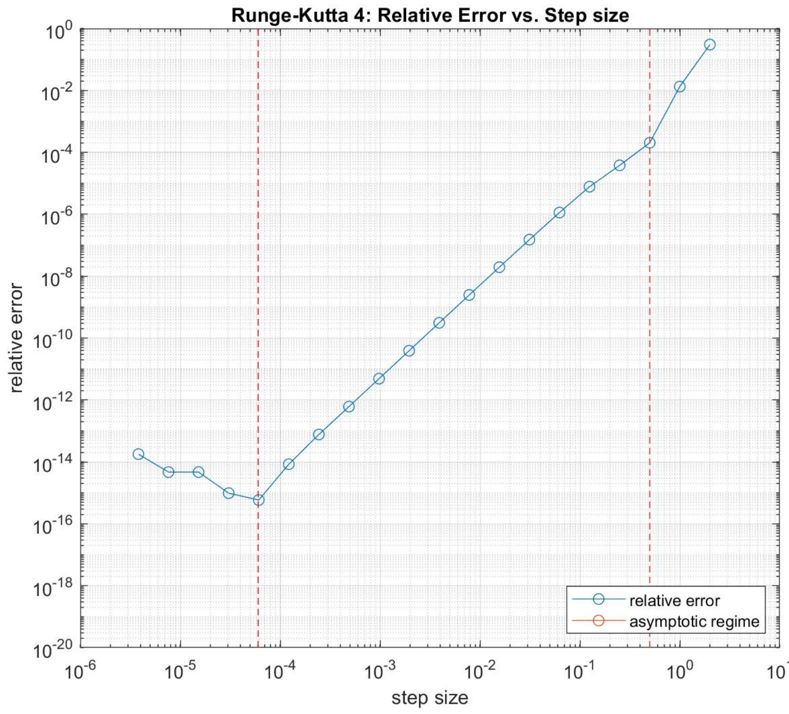
I chose to compute step sizes of h from h = 2^{-k} for k = -2,-1,0,1,....22. This gave me a better display of the range of the asymptotic regime. As shown, I cannot see a region dominated by ronud-off error in the asymptotic regime of the graph. This is because my computer couldn't compile the function for an h smaller than 2^{-22}.

Therefore I cannot define an exact interval for the asymptotic regime. Instead I define a lower-bound for the interval as the range of h's from **[log(0.5), Ω(log(2.38^{-7}))]**.

I can conclude that because Euler's method has a global error of O(*h*), our function has an observed rate of convergence of:

$$\textbf{C * O(h),} \text{ where C} = \frac{\Delta y}{\Delta x} = \frac{\log(.049) - \log(1.5^{-8})}{\log(0.5) - \log(2.38^{-7})} = \textbf{0.0423}$$

1c. Implement *4th order Runge-Kutta*to estimate the solution to the differential equation with step sizes *h* = 2^{-k} with k = 1,....15. For each step, compute relative error at final time t = 8. Plot error vs. step size h on a log-log scale. Identify the asymptotic regime and report observed convergence.



I chose to compute step sizes of h for h = 2^{-k} for k = 1,0,-1,...,-18, because it yielded a better display of the asymptotic regime and it's interval.

From the graph, we can conclude that the asymptotic regime resides on the interval of h's from **[log(0.5),log(6.1^{-5})].**

Because the asymptotic rate of error convergencefor Runge-Kutta 4 is O(h^2), we can conclude from the graph that the function's asymptotic rate of error convergence is:

$$\textbf{C * O}(h^2)\textbf{,} \text{ where C} = \frac{\Delta y}{\Delta x} = \frac{\log(2.04 * 10^{-4}) - \log(5.58 * 10^{-16})}{\log(0.5) - \log(6.1 * 10^{-5})} = \textbf{2.95}$$

Question 2

Consider the 3rd order differential equation:

$$y'''(t) + y''(t) + 4y'(t) + 4y(t) = 4t^2 + 8t - 10$$
$$y(0) = -3 \quad y'(0) = -2 \quad y''(0) = 2$$

2a. Verifty the solution is $y(t) = -\sin(2t) + t^2 - 3$

Differentiate 3 times:

$\Longrightarrow y'(t) = -2\cos(2t) + 2t$

$\Longrightarrow y''(t) = 4\sin(2t) + 2$

$\Longrightarrow y'''(t) = 8\cos(2t)$

Plug the $y^n(t)$'s into the differential equation.

$\Longrightarrow y'''(t) + y''(t) + 4y'(t) + 4y(t) = 4t^2 + 8t - 10$

$\Longrightarrow (8\cos(2t)) + (4\sin(2t) + 2) + 4(-2\cos(2t) + 2t) + 4(-\sin(2t) + t^2 - 3) = 4t^2 + 8t - 10$

$\Longrightarrow 8\cos(2t) + 4\sin(2t) + 2 - 8\cos(2t) + 8t - 4\sin(2t) + 4t^2 - 12 = 4t^2 + 8t - 10$

$\Longrightarrow 8\cos(2t) - 8\cos(2t) + 4\sin(2t) - 4\sin(2t) + 4t^2 + 8t - 10 = 4t^2 + 8t - 10$

$\Longrightarrow (8\cos(2t) - 8\cos(2t)) + (4\sin(2t) - 4\sin(2t)) = (4t^2 - 4t^2) + (8t - 8t)\ (-10 + 10)$

Because both sides of the equation are equal, the solution is correct.

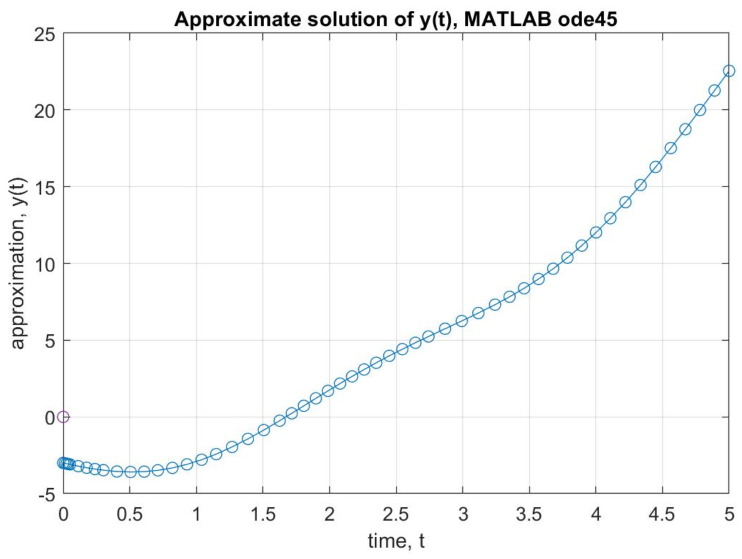
2b. Rewrite the problem as a first order system of the form *$u'(t) = f(u(t), t)$ where $u(t)$* $\leftarrow \mathbb{R}^3$. specify initial condition u(0) as a 3-vector.

$$\begin{array}{lcl} u_1(t) = y(t) & & u_1'(t) = y'(t) = u_2(t) \\ u_2(t) = y'(t) & \Longrightarrow & u_2'(t) = y''(t) = u_3(t) \\ u_3(t) = y''(t) & & u_3'(t) = y'''(t) = 4t^2 + 8t - 10 - y''(t) - 4y'(t) - 4y(t) \\ u_3(t) = y''(t) & & u_3'(t) = 4t^2 + 8t - 10 - u_3(t) - 4u_2(t) - 4u(t) \end{array}$$

$$\Longrightarrow u'(t) = \begin{pmatrix} u_1'(t) \\ u_2'(t) \\ u_3'(t) \end{pmatrix} = \begin{pmatrix} y'(t) \\ y''(t) \\ y'''(t) \end{pmatrix} = \begin{pmatrix} \boldsymbol{u_2(t)} \\ \boldsymbol{u_3(t)} \\ \boldsymbol{4t^2 + 8t - 10 - u_3(t) - 4u_2(t) - 4u(t)} \end{pmatrix}$$

$$u(0) = \begin{pmatrix} u_1(0) \\ u_2(0) \\ u_3(0) \end{pmatrix} = \begin{pmatrix} \mathbf{-3} \\ \mathbf{-2} \\ \mathbf{2} \end{pmatrix}$$

2c. Use a numerical differentiation solver (like MATLAB's *ode45*) to approximate the solution y(t) for time [0, 5]. Plot approximation vs. time for the ODE solver's solution.



2d. Experiment with the tolerance parameter of the solver. Choose tolerances of 10^{-k} with k=1,....10. For each tolerance, compute relative error at the final time t = 5. Make a table of tolerance vs. error.

<u>Function output:</u>

```
Tolerance: 1.0e-01 | Norm of rel error: 1.2963e-04
Tolerance: 1.0e-02 | Norm of rel error: 1.3415e-04
Tolerance: 1.0e-03 | Norm of rel error: 6.1121e-05
Tolerance: 1.0e-04 | Norm of rel error: 3.5799e-06
Tolerance: 1.0e-05 | Norm of rel error: 1.4055e-07
Tolerance: 1.0e-06 | Norm of rel error: 4.5269e-09
Tolerance: 1.0e-07 | Norm of rel error: 4.1510e-09
Tolerance: 1.0e-08 | Norm of rel error: 4.1774e-09
Tolerance: 1.0e-09 | Norm of rel error: 4.1805e-09
Tolerance: 1.0e-10 | Norm of rel error: 4.1805e-09
```

Appendix

1b: Euler Method

```
%NOTE - Used Prof. Constantine's Euler implementation in the euler_method call

y_0 = 0;
f_time = 8.0;
fun = @(t, y) sin(t) - y;
real_val = (1/2) * (exp(-8) + sin(8) - cos(8));

hvals = zeros(25);
approx = zeros(25);
error = zeros(25);
for k = 1:25
    hvals(k) = 2^(-k+3);
    num_steps = 8/hvals(k);

    [T, Y] = euler_method(fun, y_0, num_steps, f_time);
    approx(k) = Y(num_steps+1, 1);
    error(k) = (abs(real_val - approx(k)) / abs(real_val));
end

loglog(hvals, error, '-o');
xlabel('step size'), ylabel('relative error'),
title('Eulers method: Relative Error vs. Step size');
line([10^(-6.6)×10^(-6.6)], [10^(-8)×10^(2)], 'Color', 'red', 'LineStyle', '--')
line([10^(-0.25)×10^(-0.25)], [10^(-8)×10^(2)], 'Color', 'red', 'LineStyle', '--')
legend('relative error', 'asymptotic regime', 'Location', 'southeast');
grid on
```

1c: 4th order Runge-Kutta

```
fun = @(t, y) sin(t) - y;
real_val = (1/2) * (exp(-8) + sin(8) - cos(8));

hvals = zeros(20);
approx = zeros(20);
error = zeros(20);
for k = 1:20
    hvals(k) = 2^(-k+2);
    approx(k) = Runge_4Kutta(fun, 0, hvals(k));
    error(k) = (abs(real_val - approx(k)) / abs(real_val));
end

loglog(hvals, error, '-o');
xlabel('step size'), ylabel('relative error'),
title('Runge-Kutta 4: Relative Error vs. Step size');
line([10^(-4.22)×10^(-4.22)], [10^(-20)×10^(-0)], 'Color', 'red', 'LineStyle', '--')
line([2^(-1)×2^(-1)], [10^(-20)×10^(-0)], 'Color', 'red', 'LineStyle', '--')
legend('relative error', 'asymptotic regime', 'Location', 'southeast');
grid on
```

2c: Solving using ode45

```
%Implementation found on Mathworks site
%https://www.mathworks.com/help/matlab/ref/ode45.html
%Copyright MathWorks 2016

function dydt = y_prime(t, y)
    dydt = [y(2); y(3); 4*(t^2) + (8*t) - 10 - y(3) - (4*y(2)) - (4*y(1))];
end

tvals = zeros(53);
approx = zeros(53);
[t, y] = ode45(@y_prime, [0, 5], [-3; -2; 2]);
for i = 1:53
   tvals(i) = t(i);
   approx(i) = y(i);
   fprintf('tvals(i) = %d, approx(i) = %d\n', tvals(i), approx(i));
end

plot(tvals, approx, '-o')
xlabel('time, t'), ylabel('approximation, y(t)'),
title('Approximate solution of y(t), MATLAB ode45');
grid on
```

2d. Solving using ode45, RelTol

```
% Implementation found on Mathworks website
% https://www.mathworks.com/help/matlab/ref/odeset.html
% Copyright Mathworks 2016

function dydt = odefun(t, y)
    dydt = [y(2); y(3); 4*(t^2) + (8*t) - 10 - y(3) - (4*y(2)) - (4*y(1))];
end

real_val = -sin(2*5) + 5^2 - 3;
for k = 1:10
    tol = 10^(-k);
    options = odeset('RelTol', tol);
    [t, y] = ode45(@odefun, [0, 5], [-3; -2; 2], options);

    approx = y(length(y));
    error = abs(real_val - approx) / abs(real_val);
    fprintf('Tolerance: %.1e | Norm of rel error: %.4e\n', tol, error);
end
```