Name: Jingsi Zhou
Date: 11/30/2020
Section: 17460
Assignment: Lab 10

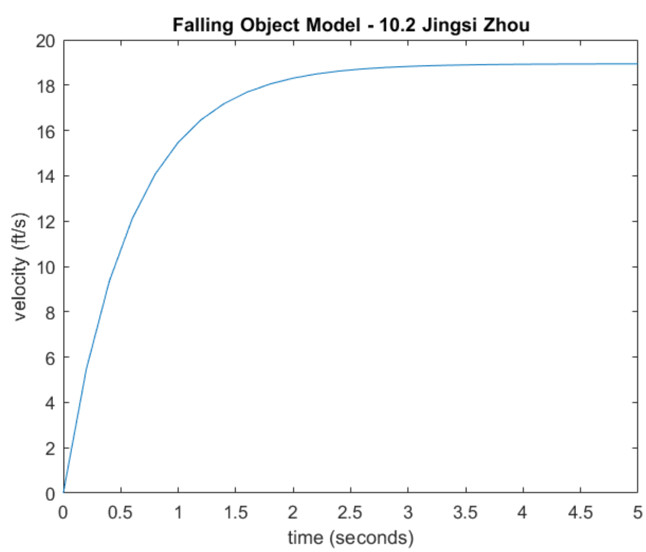
Lab 10

Problem 10.1 *Use the Composite Simpson's rule and the velocity data in the mat file to numerically calculate the position of the car at time $t = 24$ seconds. Assume that at $t = 0$, $v = 0$.*

Using the Composite Simpson's rule and the data, I calculated the position of the car at t = 24 seconds to be **682.2926 units.**

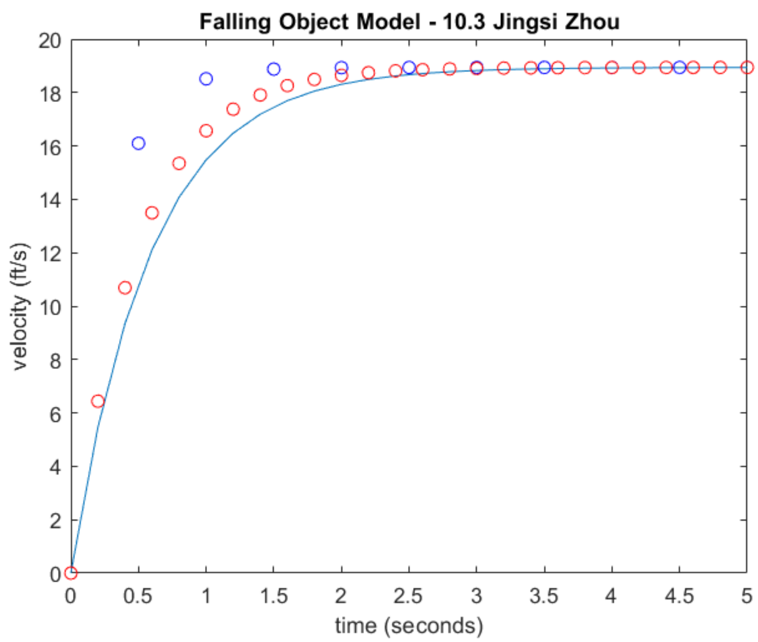
```
>> t_evens = t(3:2:5);
>> t_0 = t(1);
>> t_odds = t(2:2:6);
>> t_evens = t(3:2:5);
>> v_0 = v(1);
>> v_odds = v(2:2:6);
>> v_evens = v(3:2:5);
>> t_final = t(7);
>> v_final = v(7);
>> b_minus_a = t_final - t_0;
>> BigSimp = (t_final - t_0)*((v_0 + 4*sum(v_odds) + 2 * sum(v_evens) +
v_final)/(3*7))

BigSimp =

  682.2926
```

Problem 10.2 *Enter the necessary commands in an m-file to create a smooth line plot of Eqn. (10.5) for $t = 0$ to 5 seconds with $\rho = 1.7$ and $g = 32.2 feet/s^2$. The x-axis should have units of seconds, the y-axis should have units of feet per second, and the curve should be a smooth line. Label the x and y axes appropriately and create a title for the plot that includes your name.*



```
t = [0:.2:5];
g = 32.2;
p = 1.7;
y = (-32.2/1.7).*exp(-p.*t) + g/p;
plot(t,y);
xlabel('time (seconds)')
ylabel('velocity (ft/s)')
title('Falling Object Model - 10.2 Jingsi Zhou')
```

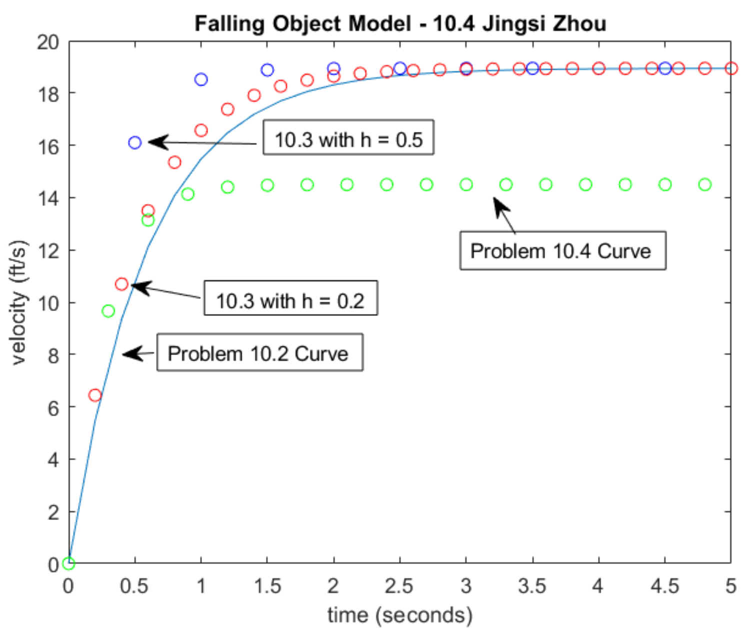
Problem 10.3 *Write a script to numerically calculate the velocity using Euler's method with* $t = 0$ *to* 5 *secs,* $h = 0.5$ *and* 0.2*,* $g = 32.2$ *feet/*s^2*,* $\rho = 1.7$ *for* $p = 1$ *and overlay a plot of your numerically calculated values of velocity on top of the exact solution you plotted in earlier. To distinguish your numerical solution from the exact solution, plot your numerical solution using circular markers (instead of a line). Make sure your plot is well labeled.*




```
t = [0:0.2:5];
g = 32.2;
rho = 1.7;
y = (-32.2/1.7).*exp(-rho.*t) + g/rho;
plot(t,y);
xlabel('time (seconds)')
ylabel('velocity (ft/s)')
title('Falling Object Model - 10.3 Jingsi Zhou')
hold on
%h = 0.5
euler_5 = [0];
p = 1;
y_i = 0; %first deriv for t=0
for t = [0:0.5:5]
    deriv_i =  -rho*y_i^p + g;
    y_ip1 = y_i + deriv_i*0.5;
    euler_5(length(euler_5)+1) = y_ip1;
    y_i = y_ip1;
```

```
end
t2 = [0:0.5:5];
plot(t2, euler_5(1:end-1), 'ob');
%h = 0.2
euler_2 = [0];
p = 1;
y_i = 0; %first deriv for t=0
for t = [0:0.2:5]
    deriv_i =  -rho*y_i^p + g;
    y_ip1 = y_i + deriv_i*0.2;
    euler_2(length(euler_2)+1) = y_ip1;
    y_i = y_ip1;
end
t3 = [0:0.2:5];
plot(t3, euler_2(1:end-1), 'or');
```

Problem 10.4 *Using the drag coeefficient you were given in the first problem, generalize the result of the previous section for* $p = 1.1$ *in Eqn. (10.4). Assure that your results have converged by gradually reducing the step size (h) until the shape of the curve no longer changes. Overlay the* $p = 1.1$ *result with the results of the previous problems. Label each curve with the text tool.*



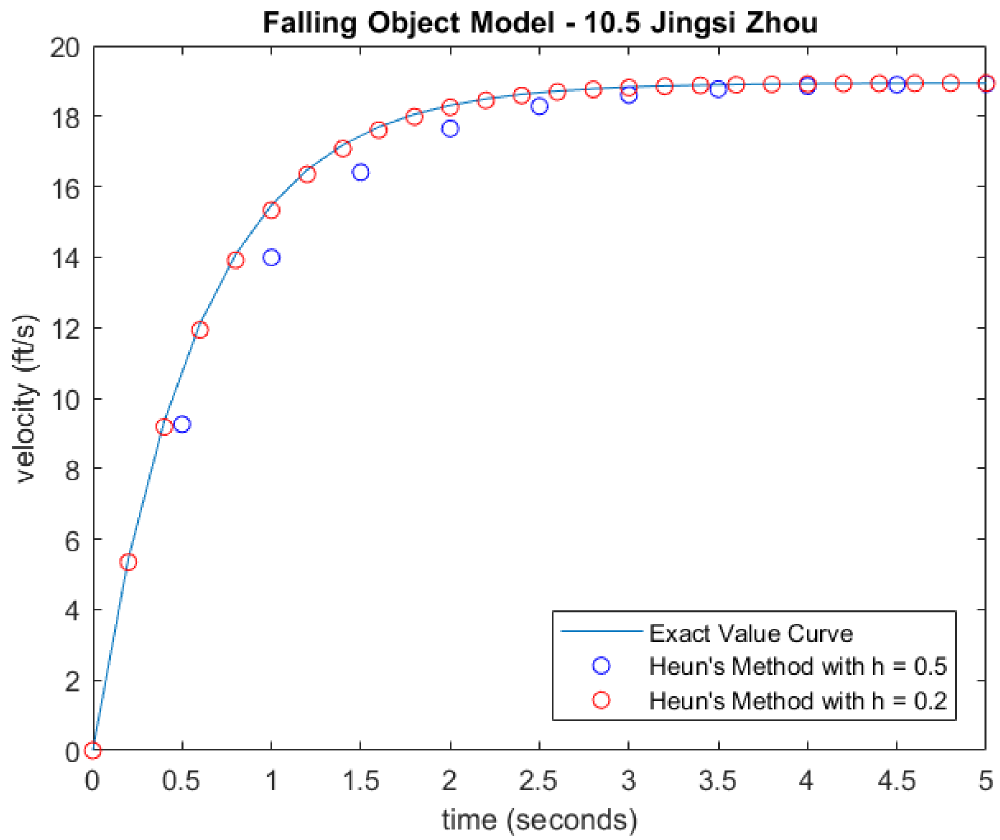

```
t = [0:0.2:5];
g = 32.2;
rho = 1.7;
y = (-32.2/1.7).*exp(-rho.*t) + g/rho;
plot(t,y);
xlabel('time (seconds)')
ylabel('velocity (ft/s)')
title('Falling Object Model - 10.3 Jingsi Zhou')
hold on
%h = 0.5
euler_5 = [0];
p = 1;
y_i = 0; %first deriv for t=0
```

```
for t = [0:0.5:5]
    deriv_i =  -rho*y_i^p + g;
    y_ip1 = y_i + deriv_i*0.5;
    euler_5(length(euler_5)+1) = y_ip1;
    y_i = y_ip1;
end
t2 = [0:0.5:5];
plot(t2, euler_5(1:end-1), 'ob');
%h = 0.2
euler_2 = [0];
p = 1;
y_i = 0; %first deriv for t=0
for t = [0:0.2:5]
    deriv_i =  -rho*y_i^p + g;
    y_ip1 = y_i + deriv_i*0.2;
    euler_2(length(euler_2)+1) = y_ip1;
    y_i = y_ip1;
end
t3 = [0:0.2:5];
plot(t3, euler_2(1:end-1), 'or');
%p= 1.1
euler_p4 = [0];
p = 1.1;
y_i = 0; %first deriv for t=0
for t = [0:0.3:5]
    deriv_i =  -rho*y_i^p + g;
    y_ip1 = y_i + deriv_i*0.3;
    euler_p4(length(euler_p4)+1) = y_ip1;
    y_i = y_ip1;
end
t3 = [0:0.3:5];
plot(t3, euler_p4(1:end-1), 'og');
```

Problem 10.5 *1.) Write a script to numerically calculate the velocity using Heun's method with $t = 0$ to 5 secs, $h = 0.5$ and 0.2, $g = 32.2$ feet/s^2, $\rho = 1.7$ for $p = 1$ and overlay a plot of your numerically calculated values of velocity on top of the exact solution you plotted in earlier. To distinguish your numerical solution from the exact solution, plot your numerical solution using dots (instead of a line). Make sure your plot is well labeled.*

2.) Find the root mean square error between Euler and Heun methods and the analytical method, using the equation:

$$(R.M.S.\ error) = \sqrt{\frac{\sum_{i=1}^{N}(y_{i,numerical} - y_{i,analytical})^2}{N}} \tag{10.11}$$

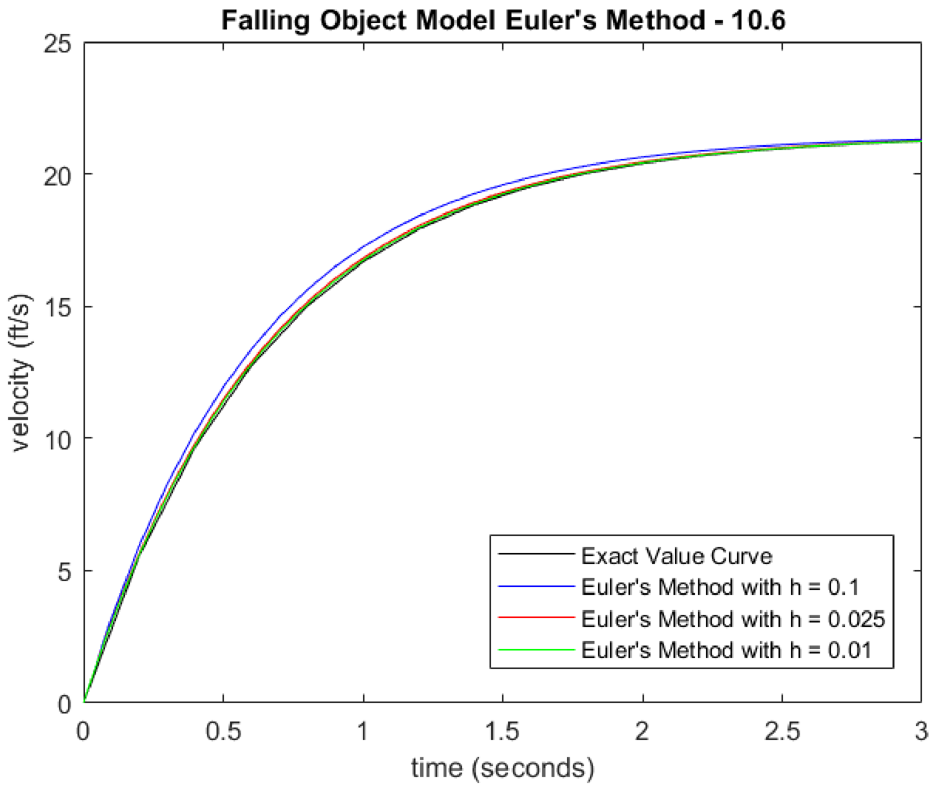


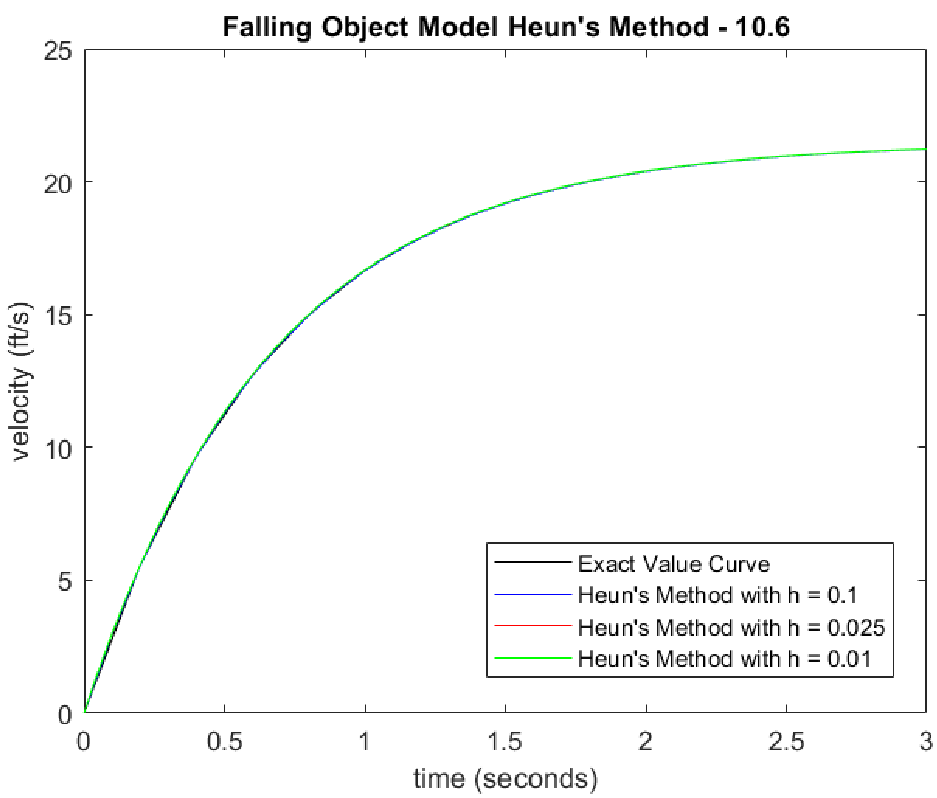

```
t = [0:0.2:5];
g = 32.2;
rho = 1.7;
y = (-32.2/1.7).*exp(-rho.*t) + g/rho;
a1 = plot(t,y);
xlabel('time (seconds)')
ylabel('velocity (ft/s)')
title('Falling Object Model - 10.5 Jingsi Zhou')
hold on
p = 1;
% h = 0.5;
heun_5 = [0];
y_i = 0;
for t = [0:0.5:5]
    yp_begin = -rho*y_i^p + g;
    y_nextpredict = y_i + yp_begin*0.5;
    yp_end = -rho*y_nextpredict^p + g;
    y_avgnext = y_i + ((yp_begin + yp_end)/2)*0.5;
    heun_5(length(heun_5)+1) = y_avgnext;
    y_i = y_avgnext;
end
t2 = [0:0.5:5];
a2 = plot(t2, heun_5(1:end-1), "ob")
% h = 0.2;
heun_2 = [0];
y_i = 0;
for t = [0:0.2:5]
    yp_begin = -rho*y_i^p + g;
```

```
    y_nextpredict = y_i + yp_begin*0.2;
    yp_end = -rho*y_nextpredict^p + g;
    y_avgnext = y_i + ((yp_begin + yp_end)/2)*0.2;
    heun_2(length(heun_2)+1) = y_avgnext;
    y_i = y_avgnext;
end
t3 = [0:0.2:5];
a3 = plot(t3, heun_2(1:end-1), "or")
M1 = "Exact Value Curve";
M2 = "Heun's Method with h = 0.5";
M3 = "Heun's Method with h = 0.2";
legend([a1,a2,a3], [M1, M2, M3]);

>> euler_2real = euler_2(1:end-1);
>> heun_2real = heun_2(1:end-1);
>> diffyeuler = euler_2real - y;
>> diff_eulerSQUARED = diffyeuler.^2;
>> RMS_euler = sqrt((sum(diff_eulerSQUARED))/26)

RMS_euler =

    0.6067

>> diffyheun = heun_2real - y;
>> diff_heunSQUARED = diffyheun.^2;
>> RMS_heun = sqrt((sum(diff_heunSQUARED))/26)

RMS_heun =

    0.0805
```

Problem 10.6 *Using $\rho = 1.5$, change the time step to (a) 0.1, (b) 0.025, and (c) 0.01 seconds, and run both Euler and Heun's method programs to obtain plots for $t = 0$ to 3. Overlay each case on two plots (one plot for Euler's and one plot for Heun's method), such that each plot has three plot lines corresponding to the various time steps. Explain what you notice with the solution. Did the numerical solution change? What does this tell you about convergence of both methods? Is one better than the other in terms of convergence? Also, would you decrease the time step further? Why or why not?*

Both the Heun and Euler Methods converge to the analytical curve. Thus, the numerical solution essentially remains unchanged except for a few inaccuracies. Both methods are convergent toward the analytical curve. However, Euler seems to be better than Heun in terms of convergence because as step size decreases, the curve converges. For the Heun method, the smaller step size of h = 0.01 seems to begin to diverge from the analytical curve. I would not decrease the time step further for the Heun model because of this. However, for the Euler method, I would decrease the time step to see if the next graph continues to converge.


Falling Object Model Euler's Method - 10.6
velocity (ft/s)
time (seconds)
0
5
10
15
20
25
0
0.5
1
1.5
2
2.5
3
Exact Value Curve
Euler's Method with h = 0.1
Euler's Method with h = 0.025
Euler's Method with h = 0.01


Falling Object Model Heun's Method - 10.6
velocity (ft/s)
time (seconds)
0
5
10
15
20
25
0
0.5
1
1.5
2
2.5
3
Exact Value Curve
Heun's Method with h = 0.1
Heun's Method with h = 0.025
Heun's Method with h = 0.01

Problem 10.7 *Consider the following 2nd order differential equation:*

$$\frac{d^2y}{dt^2} + 0.5\frac{dy}{dt} + 3y = 0 \tag{10.12}$$

At t=0, y=0.5 and dy/dt=0.2.

1.) Formulate the given second order equation as two first order equations.

2.) Write a MATLAB program to solve for y and dy/dt using Heuns method over the interval t=[0,10] with a step size of 0.1.

3.) Plot your solutions on a single plot (use plot command and **hold on** *trick; plot y and dy/dt with different colors and enclose a legend).*

1)

$$\frac{d^2y}{dt^2} + 0.5(0.2) + 3(0.5) = 0$$

$$y'' + 0.5y' + 3y = 0 \qquad y(0) = 0.5 \quad y'(0) = 0.2$$

$$x_1(t) = y(t)$$
$$x_2(t) = y'(t) \qquad x_1(0) = y(0) = 0.5$$
$$x_2 = y'(0) = 0.2$$
$$x_1' = y' = x_2$$
$$x_2' = y'' = -0.5y' - 3y = -0.5x_2 - 3x_1$$

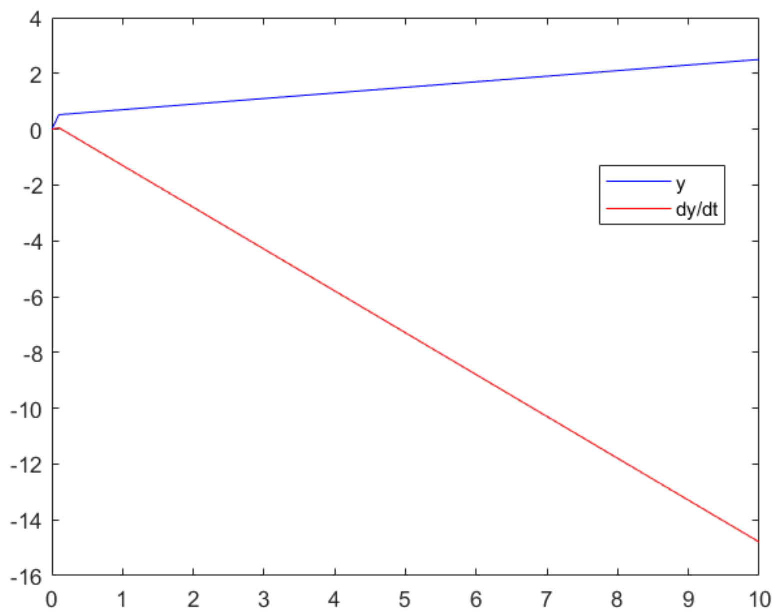
$$x_1' = x_2 \qquad x_1(0) = 0.5$$
$$x_2' = -0.5x_2 - 3x_1 \qquad x_2(0) = 0.2$$

2)

```
heun_1 = [0];
y_i = 0.5;
yp_begin1 = 0.2;
for t = [0:0.1:10]
    yp_begin = yp_begin1;
    y_nextpredict = y_i + yp_begin*0.1;
    yp_end = (y_nextpredict-y_i)/0.1;
    y_avgnext = y_i + ((yp_begin + yp_end)/2)*0.1;
    heun_1(length(heun_1)+1) = y_avgnext;
    yp_begin1 = (y_avgnext-y_i)/0.1;
    y_i = y_avgnext;
end
t2 = [0:0.1:10];
a2 = plot(t2, heun_1(1:end-1), "-b");
hold on
heun_2 = [0];
y_n = 0.2;
```

```
yp_begin1 = -0.5*heun_1(1)-3*(0.5);
for t = [0:0.1:10]
    yp_begin = yp_begin1;
    y_nextpredict = y_n + yp_begin*0.1;
    yp_end = (y_nextpredict-y_n)/0.1;
    y_avgnext = y_n + ((yp_begin + yp_end)/2)*0.1;
    heun_2(length(heun_2)+1) = y_avgnext;
    yp_begin1 = (y_avgnext-y_n)/0.1;
    y_n = y_avgnext;
end
t3 = [0:0.1:10];
a3 = plot(t3, heun_2(1:end-1), "-r");
M2 = "y"
M3 = "dy/dt"
legend([a2,a3], [ M2, M3]);
```

3)



Problem 10.8 *After identifying your state variable, perform the following:*

1.) Create a function for the state equation of the problem (Eq. 10.13). Assume $R = 0.1, C = 1.0$ *and that there is no voltage being supplied from the voltage source* ($v(t) = 0$)*. Your function should return the value of the derivative of the state variable evaluated with the value of the state variable as input.*

2.) Write a program that prompts a user for the initial condition $y(0)$, t_0, h, *and* t_{end}, *and uses the RK4 Method to determine the value* $y(t)$ *at each time step. Use the following values to test your program:* $R = 0.1$, $C = 1.0$, $t_0 = 0$ *s*, $h = 0.05$ *s*, $t_{end} = 1.0$ *s. Assume that the capacitor has a charge of 4 volts as the initial condition* ($y(0) = 4$)*.*

3.) Plot the voltage $y(t)$ *vs. time. Be sure and label your graph.*

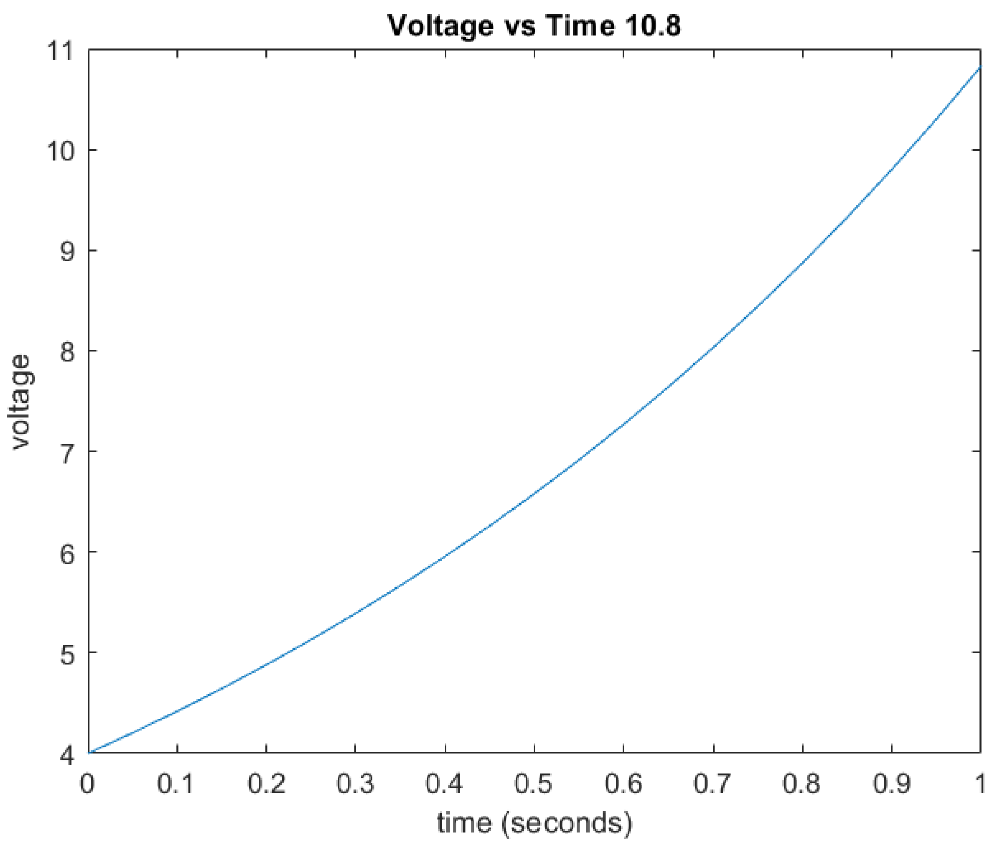
1) y' = -10y

2)

```
y_0 = input('Initial condition: ');
t_0 = input('t initial? ');
```

```
h = input('step? ');
t_end = input('end value of t? ');
vval = [];
xni = -10*y_0;
for t = [t_0:h:t_end]
    xn = xni;
    disp(['At t = ', num2str(t), ', y'' = ', num2str(xn)])
    k1n = [xn, t];
    k2n = [xn + (k1n* h/2), t + h/2];
    k3n = [xn + (k2n* h/2), t + h/2];
    k4n = [xn + (k3n* h/2), t + h/2];
    xnp1 = xn + (h/6)*(k1n(1) + 2*k2n(1) + 2*k3n(1) + k4n(1));
    v = 0.1*xn + y_0;
    vval(length(vval)+1) = v;
    xni = xnp1;
end
t1 = [t_0:h:t_end];
plot(t1, vval)

xlabel('time (seconds)')
ylabel('voltage')
title('Voltage vs Time 10.8')
```

3)



Problem 10.9 *Modify your state equation to include $v(t)$ as given in Eq. 10.14 and perform the following :*
1.) Assume $R = 0.1$, $C = 1.0$, and $v(t)$ is described by Eq. 10.14. Create a new function for the state equation of the problem.

2.) Determine the value $y(t)$ at each time step using the RK4 method. Use the following values to test your program: $R = 0.1$, $C = 1.0$, $t_0 = 0$ s, $h = 0.05$ s, $t_{end} = 1.0$ s. Assume that the capacitor has a charge of 4 volts as the initial condition ($y(0) = 4$). In your own words, explain how this problem differs from the previous problem in terms of the RK4 algorithm and how did you modify your previous code to handle this change.

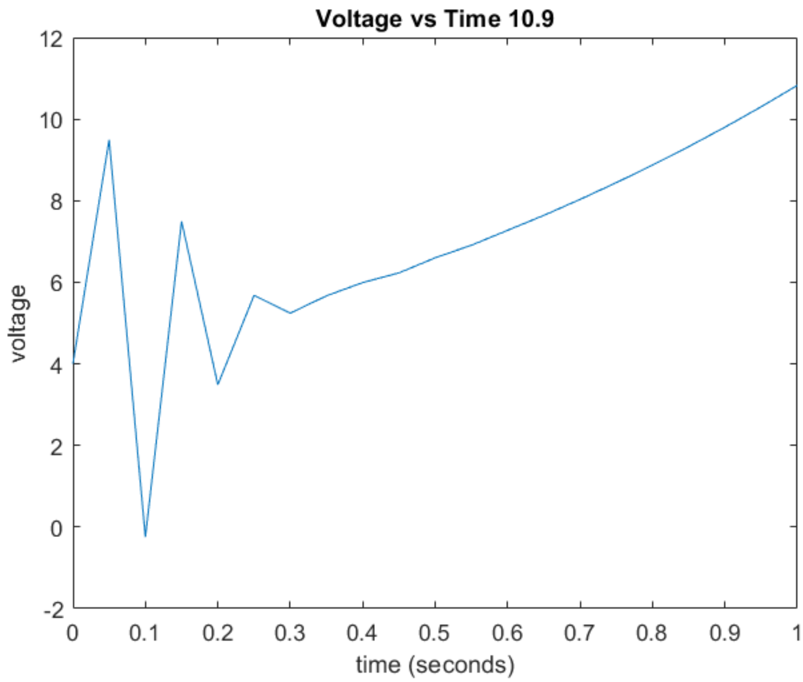
3.) Plot the voltage $y(t)$ vs. time. Be sure and label your graph.

4.) Repeat part 3.) for $h = 0.01$ seconds. Was this plot different than the one you got in part 3? Which plot would you consider more accurate and why? What did you learn from this?

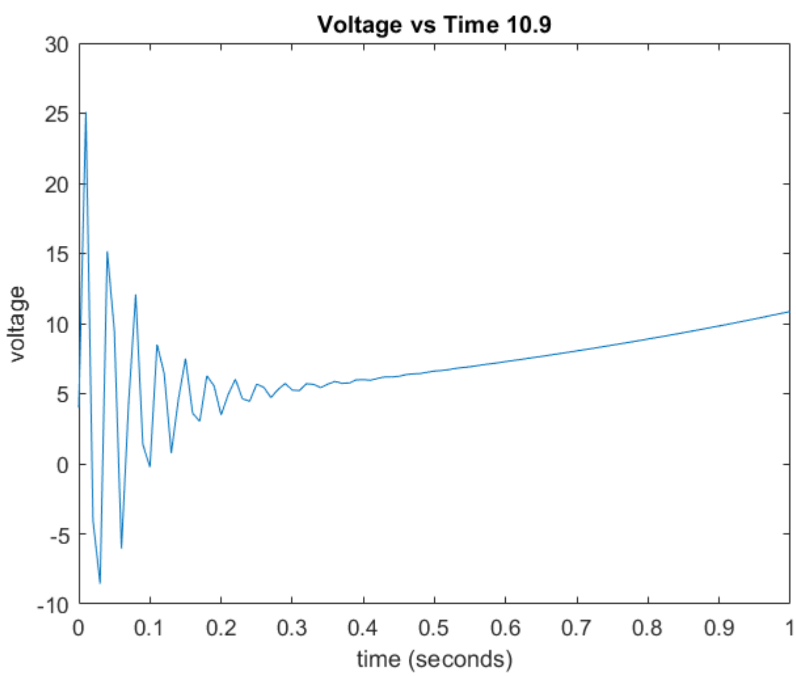
1) dy/dt = -10y + 249e^(-t/0.07)*sin(2pi*t/0.035)
2) At t = 0, y' = -40
   At t = 0.05, y' = -42.0423
   At t = 0.1, y' = -44.1889
   At t = 0.15, y' = -46.445
   At t = 0.2, y' = -48.8164
   At t = 0.25, y' = -51.3088
   At t = 0.3, y' = -53.9285
   At t = 0.35, y' = -56.682
   At t = 0.4, y' = -59.576
   At t = 0.45, y' = -62.6178
   At t = 0.5, y' = -65.8149
   At t = 0.55, y' = -69.1753
   At t = 0.6, y' = -72.7072
   At t = 0.65, y' = -76.4194
   At t = 0.7, y' = -80.3212
   At t = 0.75, y' = -84.4222
   At t = 0.8, y' = -88.7326
   At t = 0.85, y' = -93.263
   At t = 0.9, y' = -98.0248
   At t = 0.95, y' = -103.0297
   At t = 1, y' = -108.2901

   In terms of the RK4 algorithm, this problem is the same. However, to account for the new sinusoidal v(t) function, I edited the equation that calculates the voltage.

3)



4)



This plot is different and more accurate. This plot is more accurate because it considers more "cycles" because the sine function is periodic and oscillating. I learned that a smaller step time allows for a more accurate model graph.
