**Introduction:**

Problem Statement 12.1

1. The first part of the problem asked us to write a code for the state function dy/dt by using equation 1.
2. For the second part, you have to write a code that asks the user to input variables needed to solve the function created in the first part. The program uses the input values and the Runge kutta method to approximate future values of y, the charge in the capacitor.
3. Third part asks us to plot the solutions found in the second part.

**Unforced System**

Code –

function dy = capacitor(x)

dy = -x/.1; % 1/RC = 1/.1 … dy/dt = 1/.1 \* (-y)

end

The state variable is solved for to create the dy/dx relationship needed to solve the first order ode. In the unforced system, the value for vt is = 0 and R times C is .1. Using these numbers and equation 1 we solve for dy/dx.

Problem Statement 12.2

1. First part asks is to modify the function created in part one of 12.1. The function in this question needs to account for a nonzero v(t), equation.
2. For the second part, you have to write a code that asks the user to input variables needed to solve the function created in the first part. The program uses the input values and the Runge kutta method to approximate future values of y, the charge in the capacitor. This question is different than the second part of 12.1 due to the fact the v(t) is not zero.
3. Third part asks us to plot the solutions found in the second part.
4. Fourth part asks you to repeat part three with a different step size.

**Forced System**

function dy = capacitor2(t,x)

u = 24.9\*exp(-t/.07)\*sin((2\*pi())\*t/.035); % the v(t) is used instead of zero

dy = (-x+u)/.1; % 1/RC = 1/.1 … dy/dt = 1/.1 \* (-y+v(t))

end

The state variable is solved for to create the dy/dx relationship needed to solve the first order ode. In the forced system, the value for vt is shown in equation 2 and R times C is .1. Using these numbers and equation 1 we solve for dy/dx.

Eq. 1

Eq. 2

Eq. 3 Eq. 4 Eq. 5 Eq. 6 Eq. 7 Eq. 8

**Methods:**

The Runge-Kutta (RK) Method is an numerical approximation method the uses a combination of linear approximation slopes and calculates a weighted average slope to find future points from an initial values for x and y. Equation 3,4,5 and 6 shows how the slopes for the RK4 are calculated. Equation 7 shows how the slopes are integrated in an equation with the initial values to predict the future values.

**Unforced:**

Script

function [] = Unforced(y,j,h,z)

g = j:h:z; % time array

y1=zeros(1,length(g)); % creating voltage array

y1(1)=y; %intial voltage value

for ii=1:length(g)-1

kc1= capacitor(y1(ii)); % slope equations

kc2= capacitor(y1(ii)+(h/2)\*kc1);

kc3= capacitor(y1(ii)+(h/2)\*kc2);

kc4= capacitor(y1(ii)+h\*kc3);

slopec = (kc1+2\*kc2+2\*kc3+kc4);

y1(ii+1)=y1(ii)+(1/6)\*slopec\*h; % approximation equation

end

figure;

hold on;

grid on;

plot(g,y1);

xlabel('time');

ylabel('voltage');

title(‘unForced System’)

end

Algorithm

1. Started with creation of a function that takes in four variables from the user
2. Created a time array g with a start value j and end value z with a given step size.
3. Initialized voltage array as big as the time array with zeros
4. Initialized the first value of the voltage array with y
5. Created a loop to loop rk4 method to get multiple values to fill the voltage array
6. Used capacitor function and slope equations 3 4 5 and 6 to calculate a weighted slope
7. Used weighted slope and equation 7 get y values
8. Plotted the y values

**Forced:**

Script

function [] = forced2(y,j,h,z)

g = j:h:z; % time array

y1=zeros(1,length(g)); % creating voltage array

y1(1)=y; %intial voltage value

for ii=1:length(g)-1

kc1= capacitor2(g(ii),(y1(ii))); % slope equation

kc2= capacitor2((g(ii)+h/2),(y1(ii)+(h/2)\*kc1));

kc3= capacitor2((g(ii)+h/2),(y1(ii)+(h/2)\*kc2));

kc4= capacitor2((g(ii)+h),(y1(ii)+h\*kc3));

slopec = (kc1+2\*kc2+2\*kc3+kc4);

y1(ii+1)=y1(ii)+(1/6)\*slopec\*h; % approximation equation

end

figure;

hold on;

grid on;

plot(g,y1);

xlabel('time');

ylabel('voltage');

title(‘Forced System’)

end

Algorithm

1. Started with creation of a function that takes in four variables from the user
2. Created a time array g with a start value j and end value z with a given step size.
3. Initialized voltage array as big as the time array with zeros
4. Initialized the first value of the voltage array with y
5. Created a loop to loop rk4 method to get multiple values to fill the voltage array
6. Used capacitor2 function and slope equations 3 4 5 and 6 to calculate a weighted slope, this step is different than the function above because capacitor accounts for v(t) equation 2.
7. Used weighted slope and equation 7 get y values
8. Plotted the y values

**Results:**

**12.1**

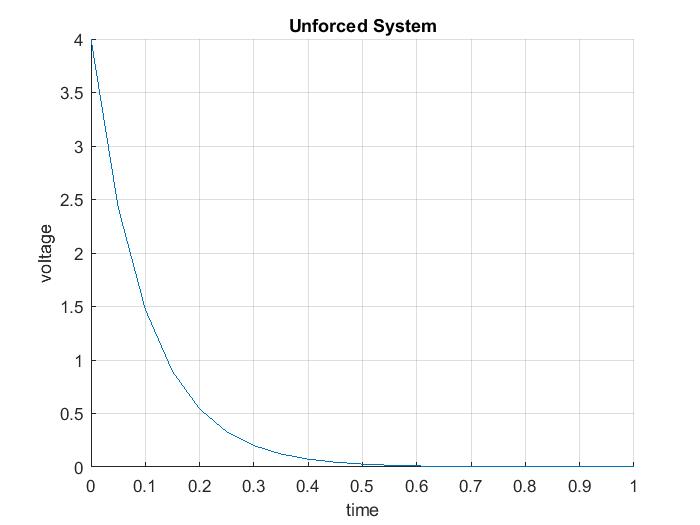


Figure 1 Charge in the Capacitor over time in an unforced system

Figure 1 shows the voltage across the capacitor over time. Initially voltage starts at 4 and drops quickly as the time goes on.

12.2

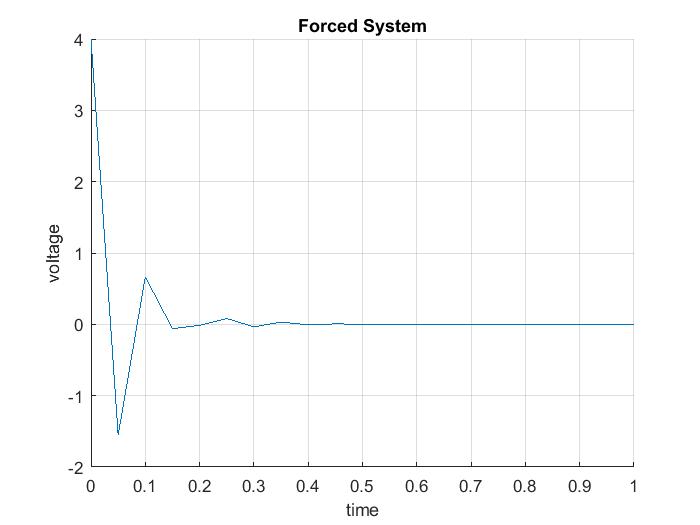


Figure 2 Charge in the Capacitor over time in a forced system with a step size of .05

Figure 2 shows the voltage across the capacitor over time. Initially voltage starts at 4 and drops quickly to a negative number and fluctuates up and down but goes to zero as the times goes on.

**Discussion:**

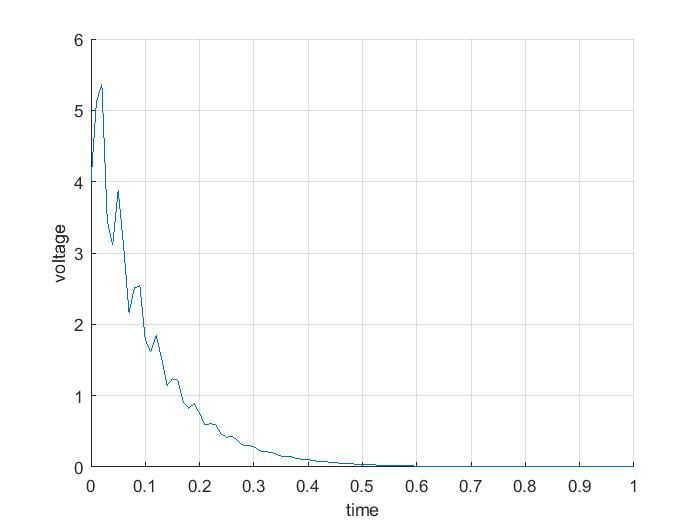


Figure 3 Charge in the Capacitor over time in a forced system with a step size of .01

Discussion

The figure 3 shows the voltage across the capacitor over time. Initially voltage starts at 4 and drops quickly and mimics the unforced graph but doesn’t create a smooth curve due to the v(t) not being 0. This graph mimics the unforced system more accurately thus this graph with stepsize of .01 is more accurate compared to the graph with a stepsize of .05. I learned that you can model first order ode’s using runge kutta method and also that a smaller step size help with modeling more complicated functions.

12.1-1

function dy = capacitor(x)

dy = -x/.1;

end

12.1-2

function [] = Unforced(y,j,h,z)

g = j:h:z;

y1=zeros(1,length(g));

y1(1)=y;

for ii=1:length(g)-1

kc1= capacitor(y1(ii));

kc2= capacitor(y1(ii)+(h/2)\*kc1);

kc3= capacitor(y1(ii)+(h/2)\*kc2);

kc4= capacitor(y1(ii)+h\*kc3);

slopec = (kc1+2\*kc2+2\*kc3+kc4);

y1(ii+1)=y1(ii)+(1/6)\*slopec\*h;

end

figure;

hold on;

grid on;

plot(g,y1);

xlabel('time');

ylabel('voltage');

title(‘unForced System’)

end

12.2-1

function dy = capacitor2(t,x)

u = 24.9\*exp(-t/.07)\*sin((2\*pi())\*t/.035);

dy = (-x+u)/.1;

end

12.2-2

function [] = forced2(y,j,h,z)

g = j:h:z;

y1=zeros(1,length(g));

y1(1)=y;

for ii=1:length(g)-1

kc1= capacitor2(g(ii),(y1(ii)));

kc2= capacitor2((g(ii)+h/2),(y1(ii)+(h/2)\*kc1));

kc3= capacitor2((g(ii)+h/2),(y1(ii)+(h/2)\*kc2));

kc4= capacitor2((g(ii)+h),(y1(ii)+h\*kc3));

slopec = (kc1+2\*kc2+2\*kc3+kc4);

y1(ii+1)=y1(ii)+(1/6)\*slopec\*h;

end

figure;

hold on;

grid on;

plot(g,y1);

xlabel('time');

ylabel('voltage');

title(‘Forced System’)

end