MATLAB Homework

Ege Ozan Özyedek

MATH242

Answers

While answering the questions, some parts of the code will be shown explicitly to emphasize their importance/role in the code. The complete code can be found in the .m files. Since every question is contained in only one function, some general things are not continuously defined (for example, in the first question variables k and m are not defined at every step since they were already defined).

- Question 1

1. To find the general solution, first the equation was defined. Then the solution was solved with the dsolve function, and then displayed on the command prompt via the disp function.

```
eqn = diff(x, t, 2) == - k/m * x; %define the diff eqn
gsolution = dsolve(eqn); %solve the eqn
fprintf('\nThe General Solution: '); %display
disp(gsolution);
```

```
C1*exp((t*(-k*m)^(1/2))/m) + C2*exp(-(t*(-k*m)^(1/2))/m) %general soln
```

2. The process in finding the particular solution was similar, the only difference between the two was the definition of initial conditions and the variables k and m. The first derivative of x is also defined in order to set the initial condition. The particular solution in this case is the position of the mass, the derivative of the particular solution is the velocity of the mass, therefore the derivative of the particular solution is obtained as well. m = 2, k = 4

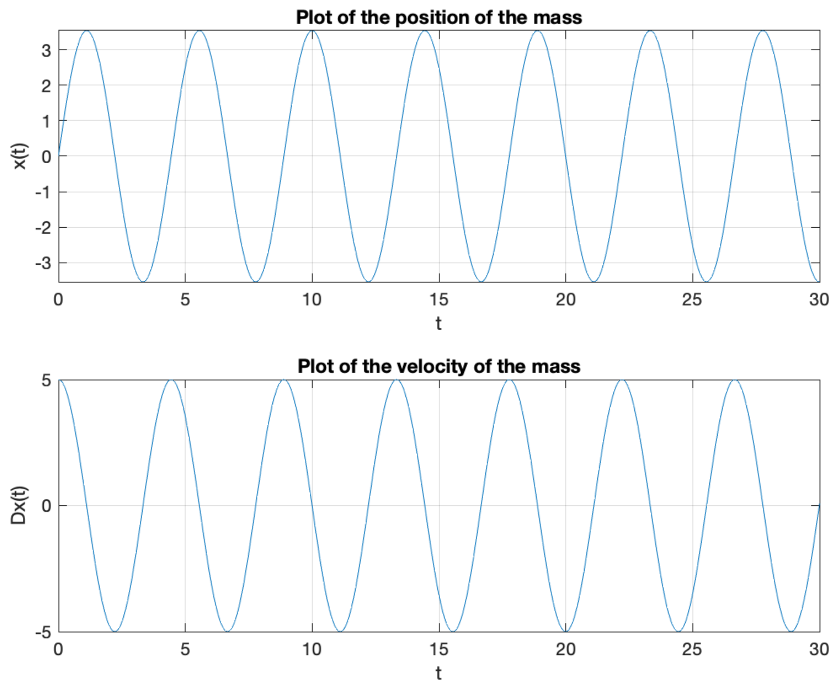
```
Dx = diff(x); %define the first derivative of x for init cond
m = 2;
k = 4;
eqn = diff(x, t, 2) == - k/m * x;
conds = [x(0) == 0, Dx(0) == 5]; %the initial conditions
psol1 = simplify(dsolve(eqn, conds)); %solve the eqn with conds
Dpsol1 = diff(psol1);
```

```
(5*2^(1/2)*sin(2^(1/2)*t))/2 %particular soln
```

3. The graph of the velocity and the position was plotted. The graph obtained is a sinusoidal function, and shows the continuous oscillation of the mass between two springs. Since there does not exist any forcing function or a perturbation term, the mass oscillates continuously and the velocity of it changes accordingly (the velocity of the mass is at maximum at position 0, its minimum at the final position)

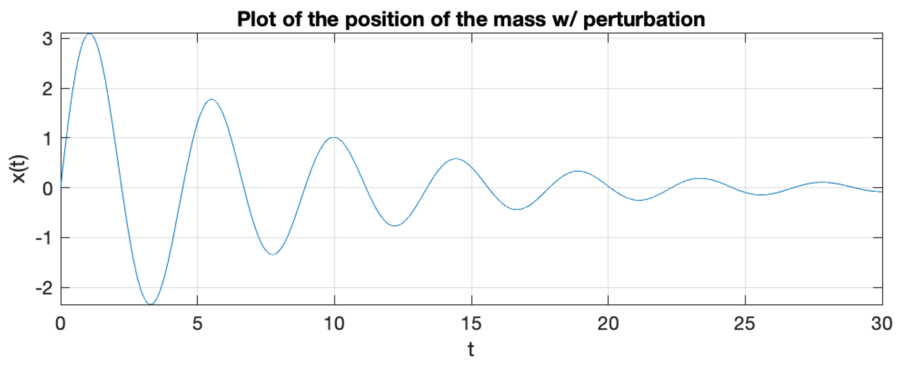
```
subplot(2,1,1);
```

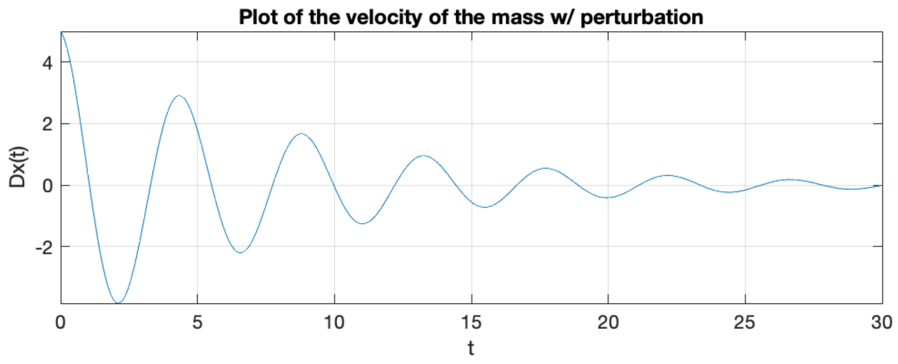
```
fplot(psol1, [0 25]);
title('Plot of the position of the mass');
xlabel('t');
ylabel('x(t)');
grid on;
subplot(2,1,2);
fplot(Dpsol1, [0 25]);
title('Plot of the velocity of the mass');
xlabel('t');
ylabel('Dx(t)')
grid on;
```



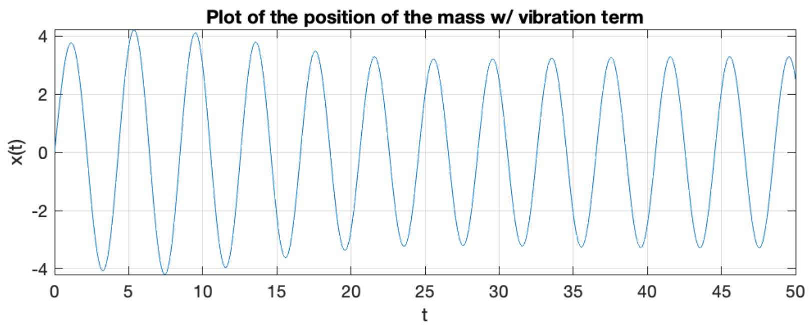
4. Similar steps were held in order to find the particular solution and in obtaining the plot of the particular solution. The perturbation term was chosen to be $\beta = 0.25$ so that it would not overrule the oscillation but would also show the effect of such term in the oscillation. The graphs now show a decaying sinusoidal signal, this is because of the effect of the perturbation term. When this term is added to the signal the new roots of the homogenous equation become complex, hence the particular equation includes decaying sinusoidal. When the perturbation term is a bigger value, say 2, it overrules the oscillation (which is the sinusoidal terms), in other words the roots become real roots instead of complex ones. This is much closer to what a real-life oscillation would look like with friction.

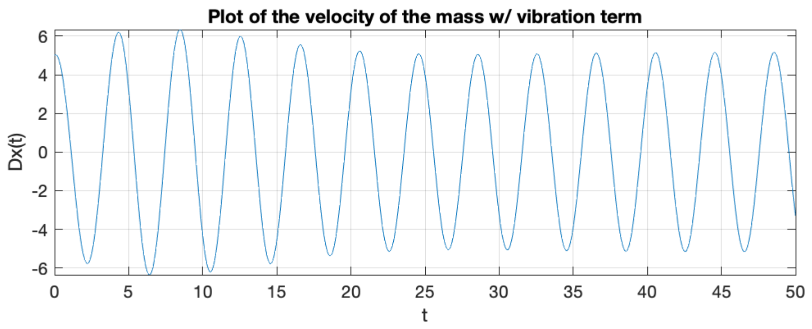
```
w = (k/m) ^(1/2);
b = 0.25;
eqn = diff(x, t, 2) + b * diff(x, t, 1) + w^2 * x == 0;
```





5. Again, the particular solution was found following the usual steps and the graphs were plotted as usual as well. The new variables were chosen as $\alpha = 2, \omega = \frac{\pi}{2}$. The addition of a sinusoidal forcing function, in addition to the perturbation term that was already existent, continues the oscillation. So, the particular solution is still a decaying sinusoidal, but now it contains another sinusoidal added to it, so it decays not to 0 but to the forcing function, compared to the equation which does not have a forcing function.



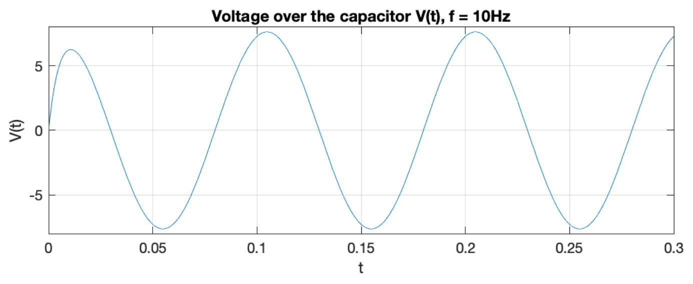


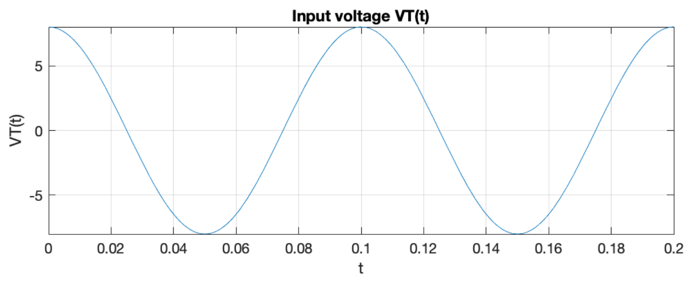
- Question 2

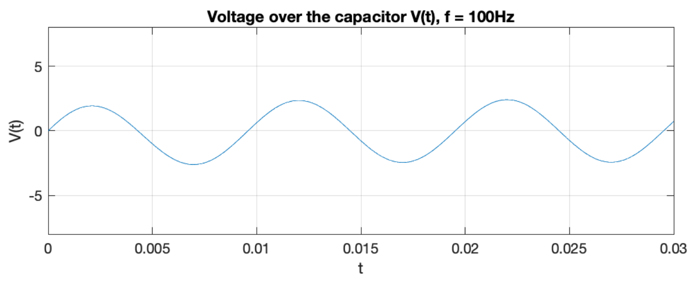
1. The general solution was found similarly to Question 1.

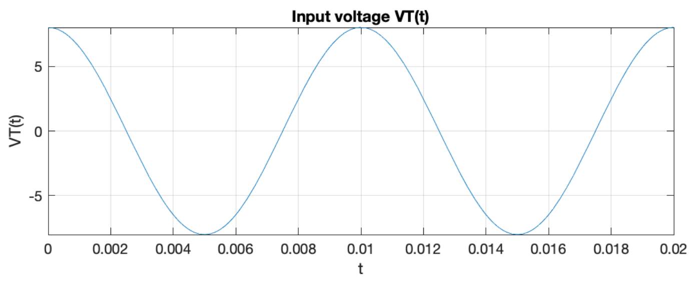
```
A = 8;
eqn = diff(V, t, 1) * RC + V == A * cos(2 * pi * f * t);
%define equation
gsol = dsolve(eqn); %find general soln
fprintf('\nGeneral Solution: ');
disp(gsol);
```

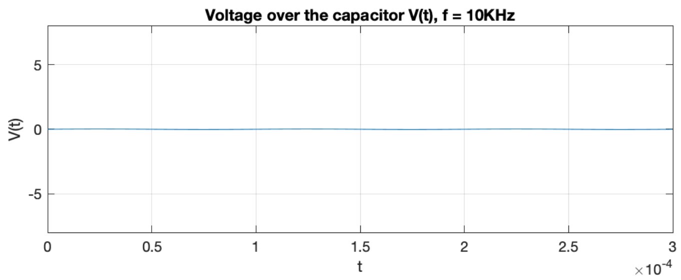
2. When the graphs are compared, it is clearly visible that the circuit only passes voltages (or rather signals) that are low frequency. It can be seen that while when f = 10Hz the amplitude of the voltage across the capacitor is nearly the same as the input voltage, as the frequency increases this amplitude lowers and at f = 10KHz it is 0. RC = $8 * 10^{-3}$

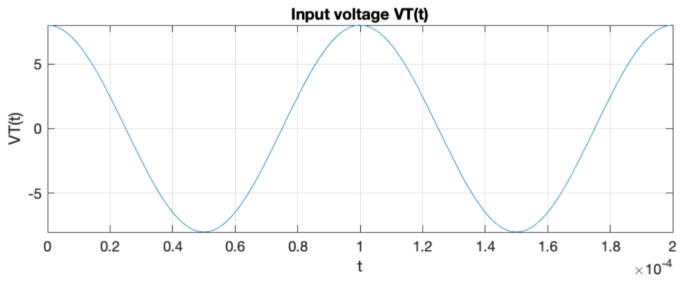




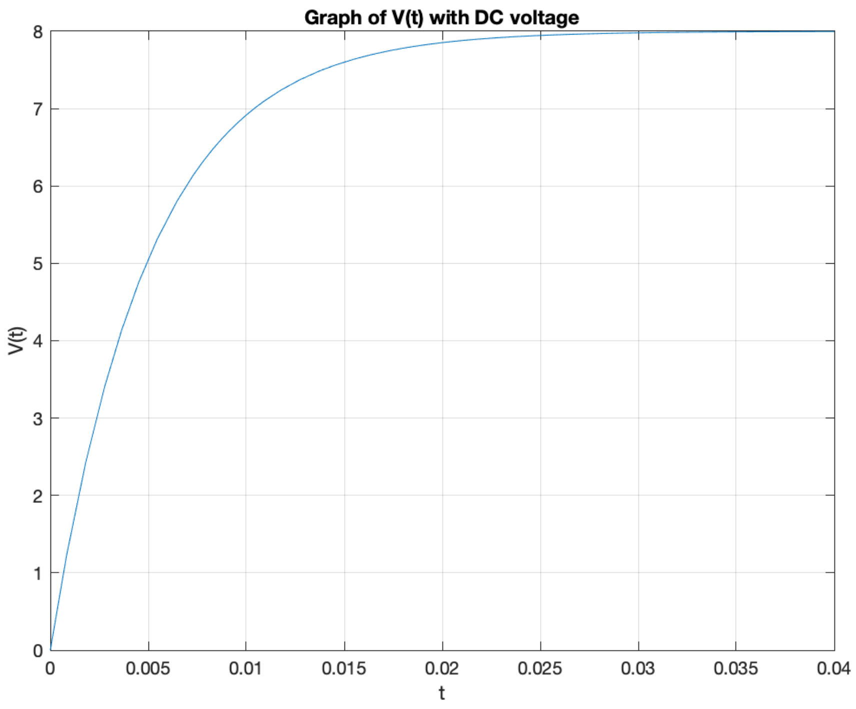




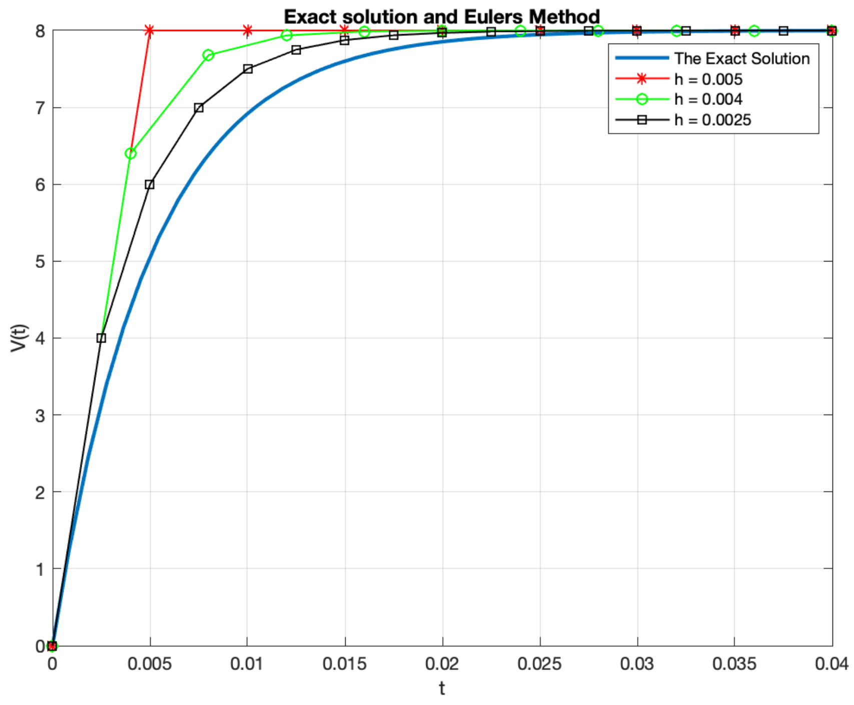




3. This time since the input voltage is not an AC signal and rather a constant DC signal, a converging exponential can be seen. It converges to the DC input voltage.

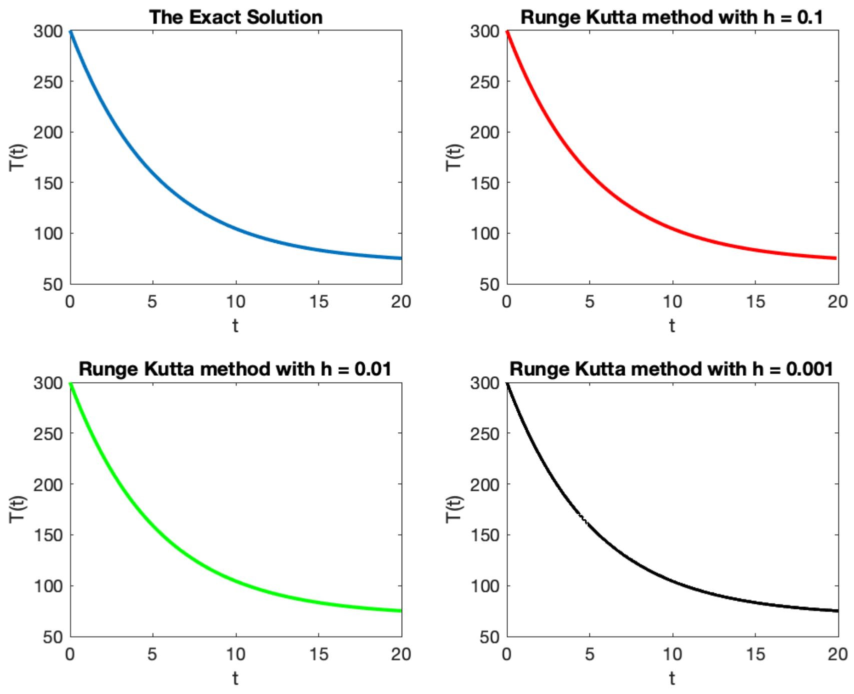


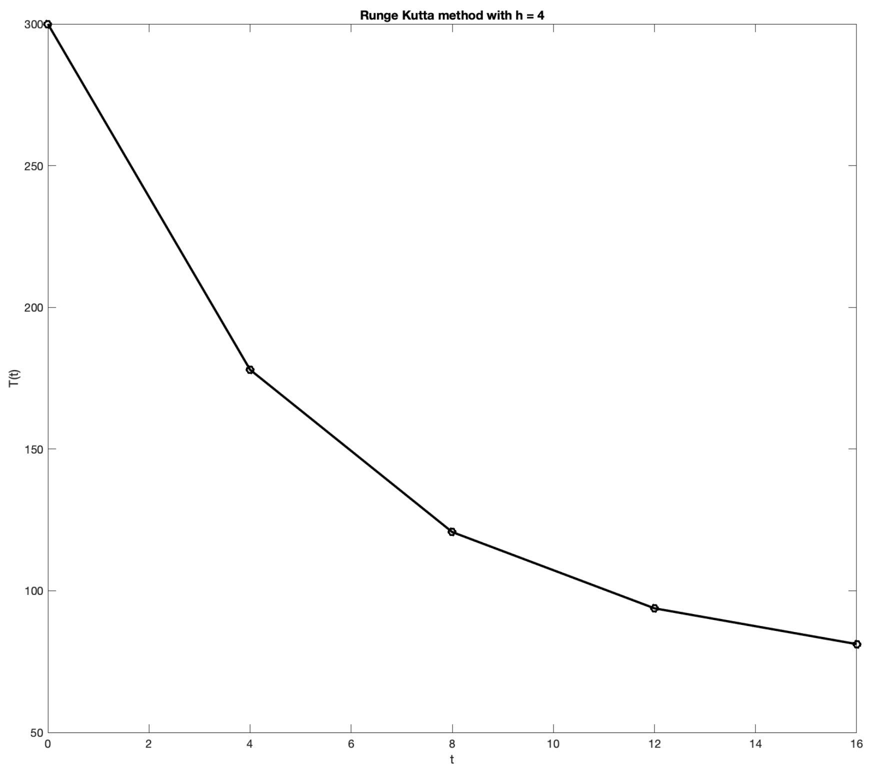
4. As we decrease the step size, Euler's Method becomes much more precise and looks a lot more like the exact solutions curve. At the biggest step size, it looks like multiple lines combined to create a graph which looks like an exponential but when the step size is decreased it is much closer to the actual exponential.



- Question 3

Similar to Euler's Method, Runge Kutta Method is more precise when the step size is at its lowest. However, since in this part the step sizes are fairly small, all of the step sizes give nearly perfect results. If all results are plotted in the same cell, no difference is visible. Instead, all graphs were plotted in the same figure. To show the difference a higher step size makes I decided to also include the graph where h = 4.





It can be seen that a higher step size lowers the credibility of the Runge Kutta method.