Assignment Part 1

Computational Engineering 1 MT2526

Group 6

Akhil Mora – 9507286954

Raghavendra Machipeddi – 9506066894

Anton Karlsson – 9303051230

Erik Ising – 9306020018

Abstract

This assignment is the first one of two parts in the course *Computational Engineering 1* and this is a group assignment, where the tasks are individually graded by how much effort each member put into the work. The tasks are solved by appointed groups where the grades also depends on the group performance. One part of the assignment are therefore to include a self-evaluation of all group members using rates to get the individually grades. The main purpose of the assignments in this course is to facilitate learning of the theory covered for the whole course, where most of the solutions are solved by Matlab.

The assignment was to analyze the given system representing a car moving along a rough road. Solving the task, we found that the Milne method was unstable. We were instructed to solve the problem with Runge-Kutta instead. The results were satisfying and we could adjust the suspension parameters to get really comfortable movement of the “car-body” described in the problem. To get even better results, we think that some sort of interpolation could have been made to increase the measurement points to get more smooth curves to integrate.

Table of Contents_Toc449627649

1 Notations

$m_1 = Mass\ of\ wheelset$

$m_2 = Mass\ of\ carbody$

$k_1 = Contact\ stiffness$

$k_2 = Suspension\ stiffness$

$c_1 = Contact\ damping$

$c_2 = Suspension\ damping$

$z_s = Displacement\ of\ the\ track$

$z_1 = Displacement\ of\ mass\ 1$

$z_2 = Displacement\ of\ mass\ 2$

2 Introduction

This assignment (*Part 1)* include a system that should represent a vehicle driving along a rough road. The whole system is analyzed as a vertical ride comfort where the height position (z_s) of the track is given by data in a separate text file given for the work. The system is represented by the 1D 2DOF model that can be seen in *fig1* below:

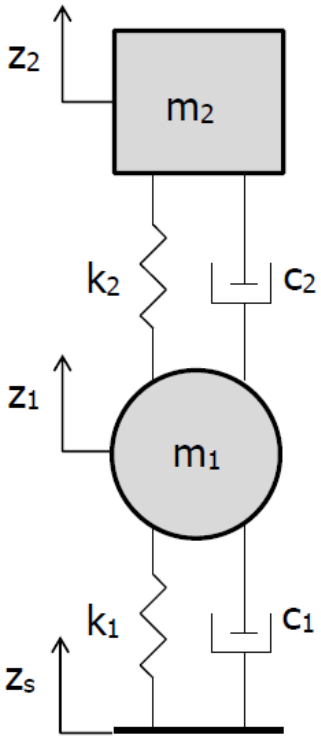

Figure 1: The 2DOF system with an excited base.

Seen from *figure 1*, mass 1 is the wheelset and mass 2 is the car-body. Those two masses are attached to each other with both suspension, consisting of both stiffness and damping. The wheelset (mass 1) have instead contact stiffness and contact damping against the road. This simplified vehicle is running with a constant velocity given as v = 72km/h and parameters values such as stiffness, damping and weight are given in *table 1* below:

Parameter	**Value**
Mass of wheelset, m_1	6000 kg
Mass of carbody, m_2	38200 kg
Contact stiffness, k_1	11200 kN/m
Contact damping, c_1	410 kNs/m
Suspension stiffness, k_2	2160 kN/m
Suspension damping, c_2	160 kNs/m

Table 1: Vehicle and contact parameters values.

3 Task 1

3.1 Problem statement

The first task of this assignment is very straight forward, including just to plot a graph of the given data according to how the ground height depending on the distance traveled. It's also asked from the task how this will be used as excitation according to the model, something that will be answered in the discussion part.

3.2 Method

To solve this task MATLAB was used. To create the graph we used the "plot-function".

3.3 Solution

From the description the track that the vehicle was driving is given as a text file called "*sz_data.txt"* there the roughness of the ground are given regarding to the distance. All rows from the first column are called "S" which are the steps values of the distance. The second column is the displacement of the track, called "Zs". Then it was just to plot the graph with all these given values, see the code down in the *appendix.*

3.4 Results

The height displacement of the road depending on the distance of the track is shown down here in *fig2*.

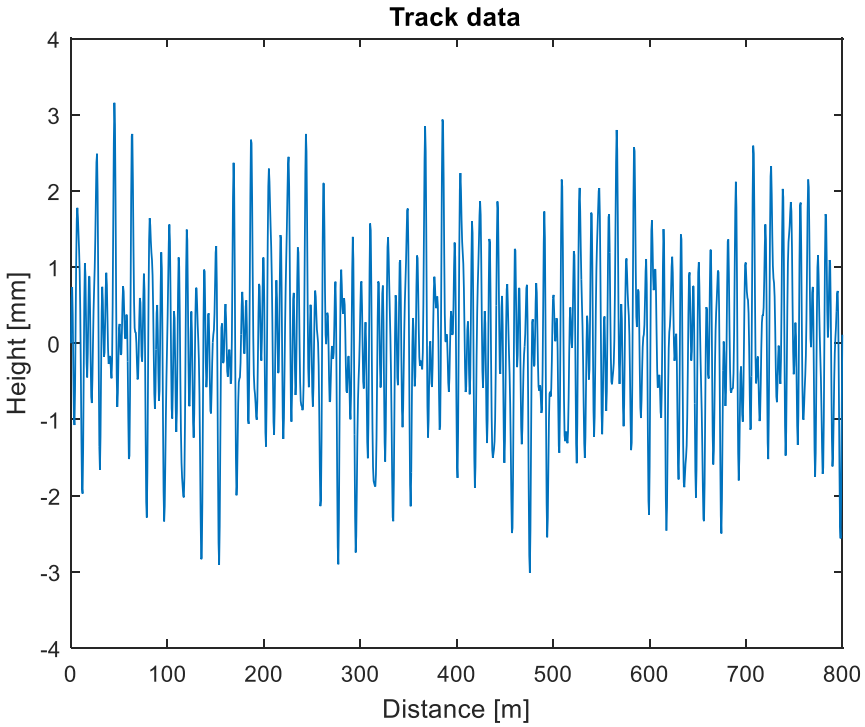

Figure 2: The roughness of the given track.

3.5 Discussion and Conclusions

The displacement of the road will be used in the model in such way that it excites the system by moving. This results in a compression or elongation of the spring. This leads to the wheelset moving, which later on leads to movement of both the suspension and car-body.

4 Task 2

4.1 Problem statement

The problem is to solve the system behavior numerically using the Milne method. It's also asked to show the response of the system in a graph. The variation of the height on the surface of the road along the route is given.

4.2 Method

The problem describes the vibrations of 2 DOF system (two masses). The behavior is represented by two equations (2nd order ODEs), one for each mass.

Overview of steps in the solution procedure:

- Two functions were defined, one for equations of motion of the system and other for Runge-Kutta method.
- The functions are called in the main script and the time response of the system is plotted.

4.3 Solution

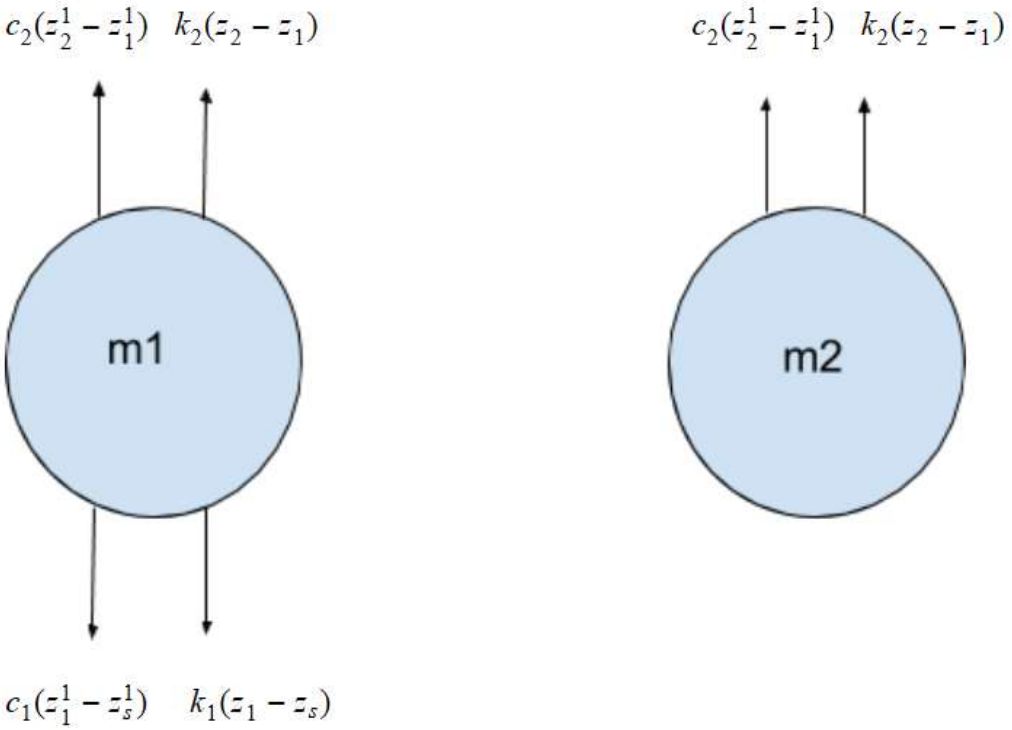

Figure 3: Free-body diagram of the system.

From the above diagrams, the equations of motion for the masses can be written as

$$m_1 z_1^{11} + c_1(z_1^1 - z_s^1) + c_2(z_1^1 - z_2^1) + k_1(z_1 - z_s) + k_2(z_1 - z_2) = 0$$

$$m_2 z_2^{11} - c_2(z_1^1 - z_2^1) - k_2(z_1 - z_2) = 0$$

As the equations of the motion are second order differential equations, they are transformed into two first order differential equations and are solved using Runge-Kutta method.

In the function, named "*vibrationeq*", the transformed equations of the system are given.

In the other function, named "*finalrungekutta*", the initial conditions are defined and a loop is created in which, the constants k_1, k_2, k_3 and k_4 are calculated for displacement and velocity equations of two masses. The iterations for displacements and velocities are calculated using the formulation of fourth order R-K method.

In the main script, the masses, spring constants and damping coefficients are defined. The step size is calculated. The function "*finalrungekutta*" is called to calculate the vertical displacements of two masses. Finally a graph is plotted between vertical displacements of two masses and time.

4.4 Results

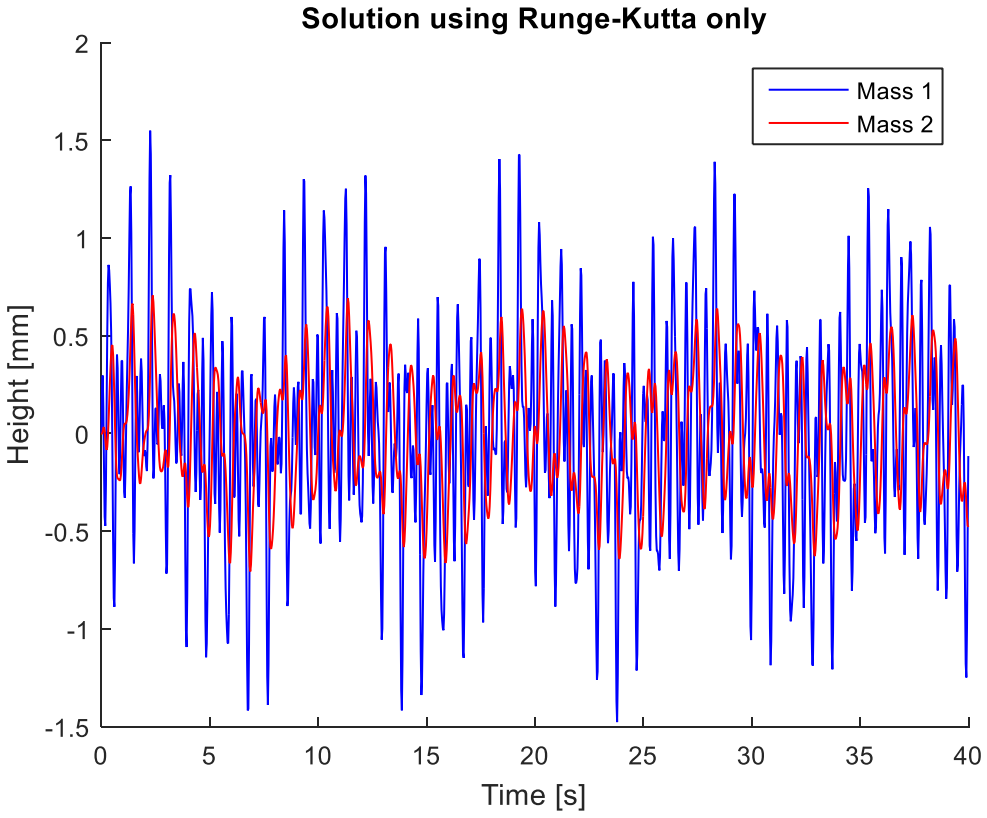

Figure 4: Time response of the system solved using Runge-Kutta method.

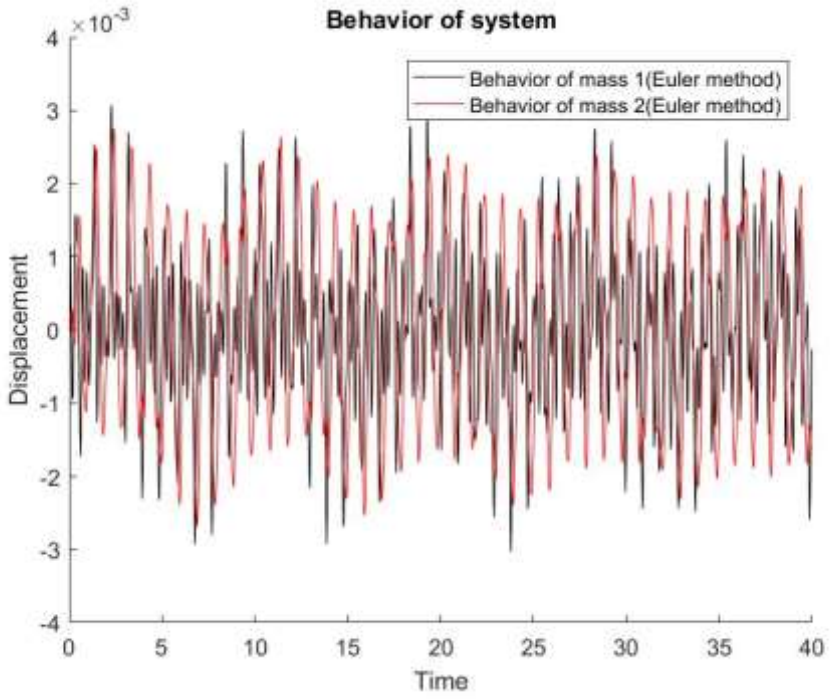

Figure 5: Time response of the system solved using Euler method.

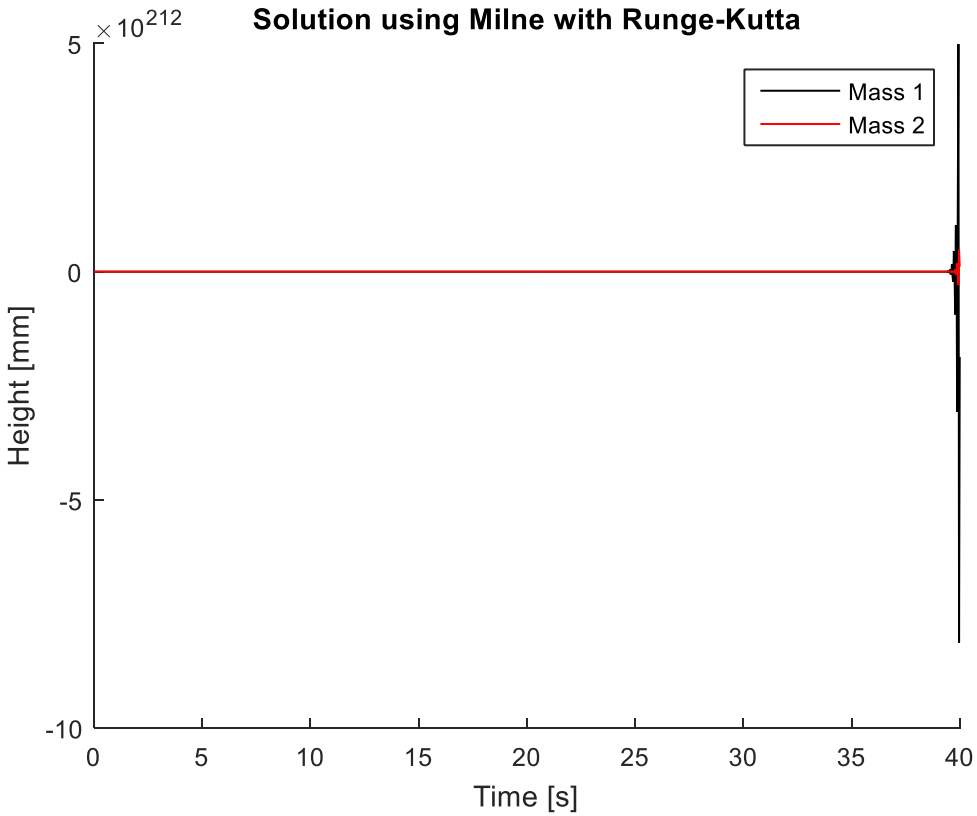

Figure 6: Time response of the system solved using Milne method

4.5 Discussion and Conclusions

The results obtained from the implementation of Runge-Kutta method were compared with that obtained from Euler and Milne methods. From the above graphs, the following conclusions were drawn.

- The maximum displacements of the masses were different when the problem was solved using Runge-Kutta and Euler methods.
- The results obtained from Milne method were observed to be unstable.

Masses	Runge-Kutta Method	Euler Method	Milne Method
m1	0.0027m	0.0031m	$8.125 \cdot 10^{212}$ m
m2	0.0007m	0.0028m	$4.8 \cdot 10^{211}$ m

Table 2: Maximum displacements of the masses:

Our choice of accuracy of the solution is Runge-Kutta method because it has truncation error of $O(h^4)$ whereas the Euler method has an error of O(h). Moreover, R-K method has better approximation because of interpretation of constants k_1, k_2 , k_3 and k_4 using Euler and Midpoint methods.

5 Task 3

5.1 Problem statement

This problem is to determine a value for the ride comfort. Something that is easily done just calculating the RMS-value (Root Mean Square) of the acceleration of the car-body (mass 2).

5.2 Method

To solve this task MATLAB's own function calculating RMS-values was used. A function where you simply write "rms()". Inside the brackets, you simply add the vector that you want to calculate the RMS-value for.

5.3 Solution

Within the Runge-Kutta method the acceleration signal could be obtained using the equation of motion for the given 2DOF-system.

5.4 Results

Our acceleration signal gives us the following result: **RMS-value ≈ 0.053 m/s^2.**

5.5 Discussion and Conclusions

Looking at *figure 7* we can see that the RMS-value seems to be correct. We obtained the acceleration from our equation of motion for every single measurement point, so the task was quite straight forward.

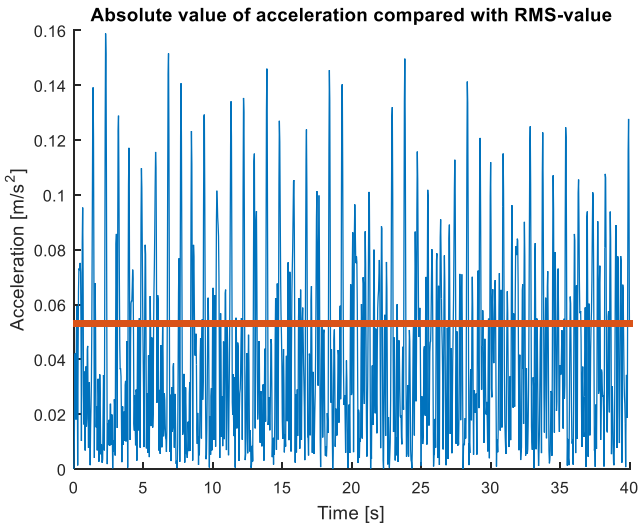

Figure 7, showing RMS-value of acceleration

6 Task 4

6.1 Problem statement

The fourth task was to suggest an alternative suspension parameter for the system (i.e. C_2 and K_2 should be changed). The suspension should be chosen to minimize the RMS-acceleration of the car-body (mass 2), and to keep the maximum displacement amplitude of the car-body within a 2 cm range.

6.2 Method

This problem was solved using MATLAB. The methodology used was that we created a loop that calculates the RMS-value by looping through different C_2 values. The program then determines the C_2 value that gives the smallest RMS-acceleration of the car-body for the current K_2 value.

6.3 Solution

With our method we obtained a lot of data. Some of that data is represented in the table below:

K2 [N/m]	C2 [Ns/m]	RMS [m/s^2]	Displ. [mm]
With the given data			
2160000	160000	0.053228205	0.71
With tested values			
2160000	101000	0.026009023	0.8
1500000	87000	0.020364022	0.7
500000	1000	0.005292629	0.22
1000	1000	1.47E-04	0.04
1	1	1.47E-07	2.20E-04

Table 3: Table showing how the suspension affects the RMS-value

6.4 Results

Studying the table above we can see that a RMS-value of about 0.0204 is obtained by using K_2 = 1500000 N/m and C_2 = 87000 Ns/m (marked as yellow in *table 3*). The displacement for all our measurements is lower than 2 cm which also can be seen in the table.

6.5 Discussion and Conclusions

From the table we can see that really low RMS-values can be obtained by decreasing both the spring and damping constants of the suspension. Being realistic, we can realize that the values in the lower half of *table 3* are too low values to be applied in reality. This because we would need a really long and thin spring which would break from the weight of the car-body. The value we found the most interesting was the one marked as yellow. There we managed to reduce the RMS value a lot without changing the spring and damping constants too much.

7 Discussion and Conclusions

The task was initially mainly about implementing the Milne method to solve the motion behavior of a simplified model of a car traveling on a rough road. After quite some time, we realized that the task was unsolvable, in the means that the method returns unstable results. This was something we really didn't expect since we could solve the problem with the Euler method alone that has an error of O(h), while the Milne method has an error of $O(h^4)$.

Because of the instability mentioned above, the task was changed to solving the problem by using the Runge-Kutta method that also has an error of $O(h^4)$. We managed to solve the problem with this method and got, as we expected, even more stable results than with the Euler method. Therefore we are satisfied with our obtained results. We also manage to get RMS-values that sounds reasonable. To get even better results, we think that some sort of interpolation could have been made to increase the measurement points to get more smooth curves to integrate.

8 References

To manage to solve these tasks the course literature and material from *It's learning* were used.

1. Broman, G (2003). *Computational Engineering.* Karlskrona: Blekinge Institute of Technology.

Appendix

Main script

```
clc
clear
close all

% TASK 1
load sz_data.txt
S = sz_data(:,1);   %distance
Zs = sz_data(:,2);  %displacement
plot(S,Zs*1000)
xlabel('Distance [m]')
ylabel('Height [mm]')
title('Track Data')

% TASK 2
steps=length(Zs);
velocity = 72/3.6;
time = S/velocity;
h = max(time)/(steps-1);
global K2 C2        %making those variables global to make them easy to
change for upcoming task 4.
C2 = 160e3;
K2 = 2160e3;



%The part trying to solve the method with the Milne method.
figure()
hold on
[t,s1,s2,v1,v2]=finalmilnewithrunge('vibrationeq',h,Zs,steps); %This method
is for our Milne solution. It's unstable.
plot(t, s1*1000,'k')
plot(t, s2*1000','r')
xlabel('Time [s]')
ylabel('Height [mm]')
title('Solution using Milne with Runge-Kutta')
legend('Mass 1','Mass 2')

%The part solving the method with the Runge-Kutta method.
figure()
hold on
[a,t,s1,s2,v1,v2]=finalrungekutta('vibrationeq',h,Zs,steps); %This method
is for our Runge-Kutta solution.
plot(t, s1*1000, 'b')
plot(t, s2*1000, 'r')
```

```
xlabel('Time [s]')
ylabel('Height [mm]')
title('Solution using Runge-Kutta only')
legend('Mass 1','Mass 2')


% TASK 3
RMSvalue = rms(a);
figure()
hold on
a(length(a)+1) = rms(a);
plot(t,abs(a))
plot([0 40], [RMSvalue RMSvalue], 'Linewidth', 4)
xlabel('Time [s]')
ylabel('Acceleration [m/s^2]')
title('Absolute value of acceleration compared with RMS-value')

% TASK 4

%THE FOLLOWING  WRAPPED CODE IS FOR SELECTING BEST C2 DEPENDING ON K.
%{
counter = 50000;                    %start value of the loop.
RMSvalueOLD = 1;
RMSvalue = 0.5;
K2 =2160e3;
while RMSvalueOLD > RMSvalue    %Loop stops when the RMS-value starts to
increase.
counter = counter + 1000        %Calculates RMS values with a given
interval.
C2 = counter;
[a,t,s1,s2,v1,v2]=finalrungekutta('vibrationeq',h,Zs,steps);
RMSvalueOLD = RMSvalue;
RMSvalue=rms(a);
end
figure()
plot(t, s2*1000, 'r')
%}


%With the code above we found out that a lower K gives a lower optimal C.
And a lower K gives a lower RMS value of the carbody.
[a,t,s1,s2,v1,v2]=finalrungekutta('vibrationeq',h,Zs,steps);
RMSvalueOLD = RMSvalue;
RMSvalue=rms(a);
```

Functions

Vibration-equation

```
function [sprime1, sprime2, vprime1, vprime2] =
vibrationeq(t,s1,s2,sz,v1,v2,vz)

%Assigning all variables.
global K2 C2;
M1 = 6000;
M2 = 38200;
C1 = 410e3;
K1 = 11200e3;

%Formulas aquired from hand calculations.
sprime1 = v1;
vprime1 = (-(C1+C2)*v1 + C2*v2 + C1*vz - (K1+K2)*s1 + K2*s2 + K1*sz)/M1;
sprime2 = v2;
vprime2 = (C2*v1 - C2*v2 + K2*s1 - K2*s2)/M2;
```

Runge-Kutta loop

```
function [accel2,t,s1,s2,v1,v2] = finalrungekutta(f,h,sz,nstep)

% Initial values
t(1)=0;
s1(1)=0;
v1(1)=0;
s2(1)=0;
v2(1)=0;

% The velocity of the height change of the ground.
vz = diff(sz)/h;
vz(length(sz)) = 0;

for i=2:nstep               %Loops through all measurement points of the
ground displacement.

    t(i)=t(i-1)+h;
    [sprime1, sprime2, vprime1, vprime2] = feval(f,t(i-1),s1(i-1),s2(i-
1),sz(i-1),v1(i-1),v2(i-1),vz(i-1)); %More or less the Euler method.
    k_11 = sprime1*h;       %k1 for velocity of mass 1.
    k_11v = vprime1*h;      %k1 for acceleration of mass 1.
    k_12 = sprime2*h;       %k2 for velocity of mass 2.
    k_12v = vprime2*h;      %k2 for acceleration of mass 2.
    accel2(i-1)=vprime2;    %For task 3, returns the acceleration of the
carbody.
    [sprime1, sprime2, vprime1, vprime2] = feval(f,t(i-1)+h/2,s1(i-
1)+k_11/2,s2(i-1)+k_12/2,(sz(i)-sz(i-1))/2,v1(i-1)+k_11v/2,v2(i-
1)+k_12v/2,(vz(i)-vz(i-1))/2);
    k_21 = sprime1*h;
    k_21v = vprime1*h;
    k_22 = sprime2*h;
    k_22v = vprime2*h;
    [sprime1, sprime2, vprime1, vprime2] = feval(f,t(i-1)+h/2,s1(i-
1)+k_21/2,s2(i-1)+k_22/2,(sz(i)-sz(i-1))/2,v1(i-1)+k_21v/2,v2(i-
1)+k_22v/2,(vz(i)-vz(i-1))/2);
    k_31 = sprime1*h;
    k_31v = vprime1*h;
```

```
    k_32 = sprime2*h;
    k_32v = vprime2*h;
    [sprime1, sprime2, vprime1, vprime2] = feval(f,t(i-1)+h,s1(i-
1)+k_31,s2(i-1)+k_32,sz(i-1),v1(i-1)+k_31v,v2(i-1)+k_32v,vz(i-1));
    k_41 = sprime1*h;
    k_41v = vprime1*h;
    k_42 = sprime2*h;
    k_42v = vprime2*h;

    %The weighting part of Runge-kutta.
    v1(i)= v1(i-1) + (1/6)*(k_11v + 2*k_21v + 2*k_31v + k_41v);
    v2(i)= v2(i-1) + (1/6)*(k_12v + 2*k_22v + 2*k_32v + k_42v);
    s1(i) = s1(i-1) + (1/6)*(k_11 + 2*k_21 + 2*k_31 + k_41);
    s2(i) = s2(i-1) + (1/6)*(k_12 + 2*k_22 + 2*k_32 + k_42);

end

end
```

Milne-Loop

```
function [t,s1,s2,v1,v2] = finalmilnewitheuler(f,h,sz,nstep)

% Initial values
t(1)=0;
s1(1)=0;
v1(1)=0;
s2(1)=0;
v2(1)=0;

% The velocity of the height change of the ground.
vz = diff(sz)/h;
vz(length(sz)) = 0;

for i=2:nstep

    t(i)=t(i-1)+h;


    if i<=4     %Runge-Kutta was used for the first 4 points.

    [sprime1, sprime2, vprime1, vprime2] = feval(f,t(i-1),s1(i-1),s2(i-
1),sz(i-1),v1(i-1),v2(i-1),vz(i-1)); %More or less the Euler method.
    k_11 = sprime1*h;       %k1 for velocity of mass 1.
    k_11v = vprime1*h;      %k1 for acceleration of mass 1.
    k_12 = sprime2*h;       %k2 for velocity of mass 2.
    k_12v = vprime2*h;      %k2 for acceleration of mass 2.
    accel2(i-1)=vprime2;    %For task 3, returns the acceleration of the
carbody.
    [sprime1, sprime2, vprime1, vprime2] = feval(f,t(i-1)+h/2,s1(i-
1)+k_11/2,s2(i-1)+k_12/2,(sz(i)-sz(i-1))/2,v1(i-1)+k_11v/2,v2(i-
1)+k_12v/2,(vz(i)-vz(i-1))/2);
    k_21 = sprime1*h;
    k_21v = vprime1*h;
    k_22 = sprime2*h;
    k_22v = vprime2*h;
```

```
    [sprime1, sprime2, vprime1, vprime2] = feval(f,t(i-1)+h/2,s1(i-
1)+k_21/2,s2(i-1)+k_22/2,(sz(i)-sz(i-1))/2,v1(i-1)+k_21v/2,v2(i-
1)+k_22v/2,(vz(i)-vz(i-1))/2);
    k_31 = sprime1*h;
    k_31v = vprime1*h;
    k_32 = sprime2*h;
    k_32v = vprime2*h;
    [sprime1, sprime2, vprime1, vprime2] = feval(f,t(i-1)+h,s1(i-
1)+k_31,s2(i-1)+k_32,sz(i-1),v1(i-1)+k_31v,v2(i-1)+k_32v,vz(i-1));
    k_41 = sprime1*h;
    k_41v = vprime1*h;
    k_42 = sprime2*h;
    k_42v = vprime2*h;

    %The weighting part of Runge-kutta.
    v1(i)= v1(i-1) + (1/6)*(k_11v + 2*k_21v + 2*k_31v + k_41v);
    v2(i)= v2(i-1) + (1/6)*(k_12v + 2*k_22v + 2*k_32v + k_42v);
    s1(i) = s1(i-1) + (1/6)*(k_11 + 2*k_21 + 2*k_31 + k_41);
    s2(i) = s2(i-1) + (1/6)*(k_12 + 2*k_22 + 2*k_32 + k_42);

    else    %The milne method (unstable in this case).
    %predictor
    [sprime1(i-1), sprime2(i-1), vprime1(i-1), vprime2(i-1)] = feval(f,t(i-
1),s1(i-1),s2(i-1),sz(i-1),v1(i-1),v2(i-1),vz(i-1));
    v1(i) = v1(i-4) + (4/3)*(2*vprime1(i-3)-vprime1(i-2)+2*vprime1(i-1))*h;
    s1(i) = s1(i-4) + (4/3)*(2*sprime1(i-3)-sprime1(i-2)+2*sprime1(i-1))*h;
    v2(i) = v2(i-4) + (4/3)*(2*vprime2(i-3)-vprime2(i-2)+2*vprime2(i-1))*h;
    s2(i) = s2(i-4) + (4/3)*(2*sprime2(i-3)-sprime2(i-2)+2*sprime2(i-1))*h;

    %corrector
    [sprime1(i), sprime2(i), vprime1(i), vprime2(i)] =
feval(f,t(i),s1(i),s2(i),sz(i),v1(i),v2(i),vz(i));
    v1(i) = v1(i-2) + (1/3)*(vprime1(i-2)+4*vprime1(i-1)+vprime1(i))*h;
    s1(i) = s1(i-2) + (1/3)*(sprime1(i-2)+4*sprime1(i-1)+sprime1(i))*h;
    v2(i) = v2(i-2) + (1/3)*(vprime2(i-2)+4*vprime2(i-1)+vprime2(i))*h;
    s2(i) = s2(i-2) + (1/3)*(sprime2(i-2)+4*sprime2(i-1)+sprime2(i))*h;
    end

end

end
```