Institutionen för Fysik
Umeå Universitet

Modeling with particle-spring systems

Anton Lindqvist (anli0516@student.umu.se)

January 31, 2017

Abstract

In this computer lab we try to simulate a simple particle system using MATLAB. To begin whit we take a look at only two particles connected by a single spring and then move on to more complex system involving both more particles and more springs. We also look at the difference between simulating a macroscopic and a microscopic system.

Kursens Namn
Handledare: Namn separerade med komma.

Contents

1 Resultat

1.1 Exercise 2: Simpler particle-spring system

In this exercise we are asked to simulate a simple system whit only two particles connected by a single spring. Both particles have the same mass $m = 1$ and at $t = 0$ the distance between the particles is 1.8 m. The spring has the the constant $\kappa_d = 10$.

Figure 1 – Two particle connected by a single spring

Question (a):

Answer:

The two particles moves back and fourth as can be sen in figure (2).

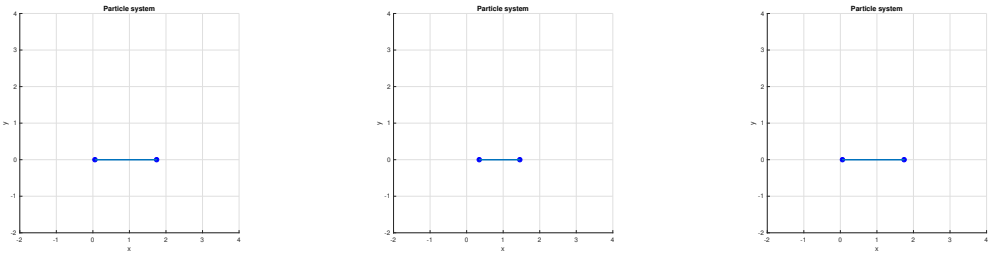

Figure 2 – Graphs showing the two particle spring system.

Question (b):

Here we are asked to calculate the spring, potential, kinetic and total energy. These quantities are denoted by E_{spring}, E_p, E_k, E_{tot} respectively. The aim is to find out if the total energy is constant. We also want to test how large Δt can be before the total energy changes more than 1% per oscillation.

Answer:

As can be seen in figure (3) the total energy for the undamped system is constant. The larges step size that restrict the energy from chaining more than 1% is $\Delta t = 0.004$.

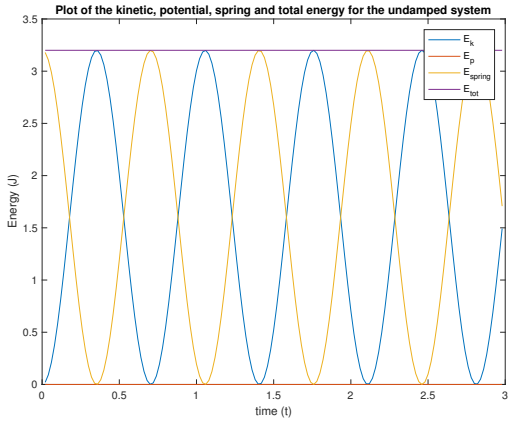

Figure 3 – Graph showing the different energy quantities plotted against time.

Question (c):

Now we want to look at how the system behave whit dampening. If we let $\kappa = 0.5$, how long will it take for the amplitude to decrees to less then 10% of its original value. To confirm this we are asked to solve equation (1) analytically and compare the result with the numerical one.

$$\ddot{r} + 2(\kappa_d/m)\dot{r} + 2(\kappa_s/m) = 0, \qquad r = |x_1 - x_2| - L \tag{1}$$

Answer:

The time it will take for the amplitude to decrees to less then 10 % of the original value is 4.64 s. The time it takes is however dependent of the chosen Δt. It takes longer for the amplitude to decrees if Δt is larger. The graph of the analyst solution to equation (1) can be sen in figure (4). This is the spring extension plotted against time and confirms that it takes about 4.6 s for the amplitude to be less then 10 % of the original value.

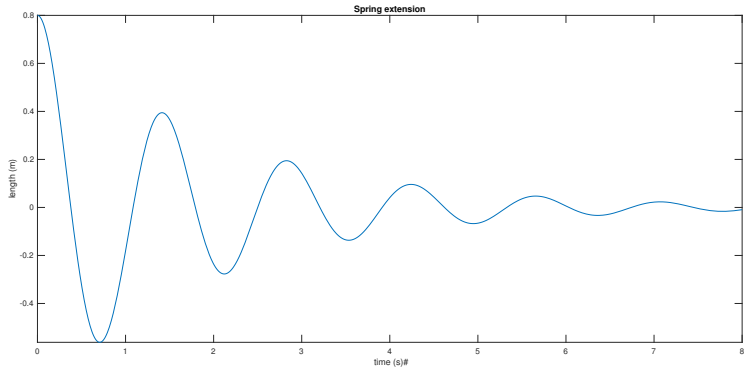

Figure 4 – Graph of the spring extension plotted against time.

Question (d):

The aim is to make the system rotate around the point $(x, y) = (0.9, 0)$. By giving the two particles the same speed $v = 5$, both parallel to the y-axis but in opposite direction, the system should now be rotating. The dampening should be set to zero before calculating the angular momentum $\mathcal{L} = \sum m(xv_y - yv_x) = I\omega$ relative to the center of mass as a function of time. Furthermore a plot of the spring length should be presented.

Answer:

As can be seen in figure (5) the equilibrium length of the spring is around 2.4. This differs from the rest length ($L = 1$) because of the rotation of the system. As the particles increase in speed the force required to keep the particle in the circular trajectory also becomes grater, resulting in an outstretch of the spring. As can be seen in figure (6) the angular momentum is constant. This correspond to what would be expected.

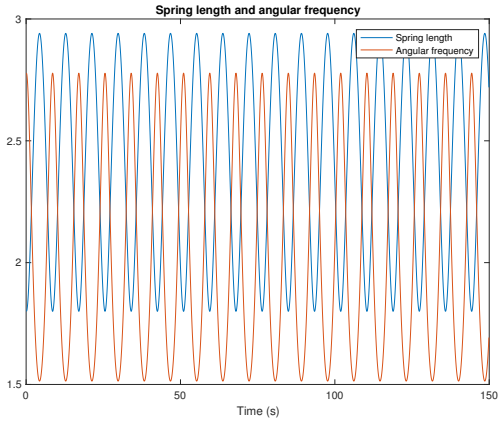

Figure 5 – Graph showing the spring length and angular frequency as a function of time. Note that the equilibrium seems to be about 2.4.

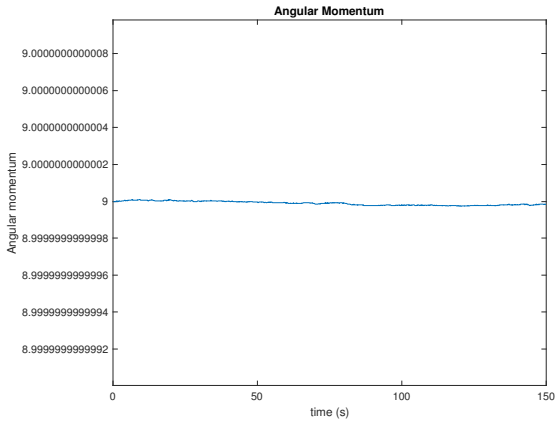

Figure 6 – Graph showing the angular momentum as a fuction of time.

1.2 Exercise 3: Modelling a microscopic 2D object

Question (a):

The model is a rectangular object consistent of 32 particles. these particles should be arranged in a $4X8$ rectangle. As in the precious exercise the particle mass is $m = 1$ and the springs connecting the particle should have the length $L = 1$ (vertical and horizontal) and $L = \sqrt{2}$ (diagonal). The spring constant is $\kappa_s = 100$ and the dampening $\kappa_d = 0$. Gravitation is set to $g = 1$.

Answer:

The model of a rectangular particle system can be seen in figure (7).

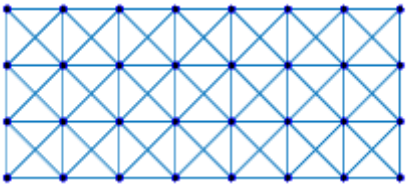

Figure 7 – Graph of the rectangular particle system.

Question (b):

Now the aim is to ad a surface simulating a floor. It will be made up by 16 circles whit the radius $R = 1$. The particle should after contact with the surface of the circles be given a new velocity v_{new} as to simulate an elastic collision.

Answer:

Figure (8) shows the same system as in figure (7) but now resting on the "floor" made up by randomly placed circles along the x-axis.

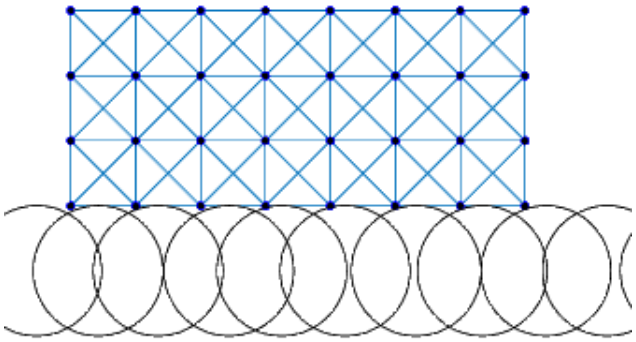

Figure 8 – Graph of the rectangular particle system resting on a "floor".

Question (c):

Now all particles are given an initial velocity $v = 5$ in the positive x-direction. The vertical initial velocity is set to zero.

Answer:

The particle spring system travels along the x-axis as can be seen in figure (9). The springs allow the object to deform as it interacts whit the surface.

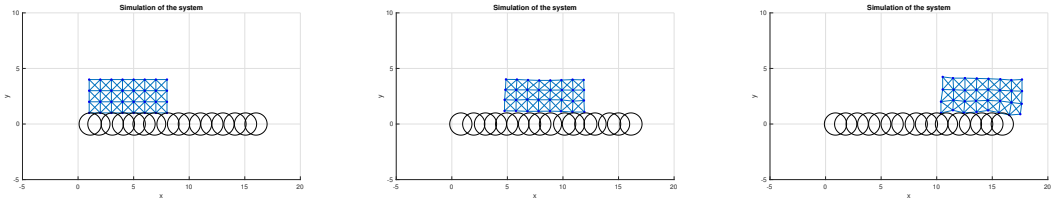

Figure 9 – Graphs showing the rectangular model for three executive iterations

Question (d):

The aim is to calculate the energy quantities as a function of time. The quantities are: E_{spring}, E_p, E_k, E_{tot}.

Answer:

Figure (10) shows the graph of the different energy quantities for the undamped system. As expected the total and the potential energy of the system is constant. The kinetic energy is decreasing as time goes on while the energy in the springs is increasing.

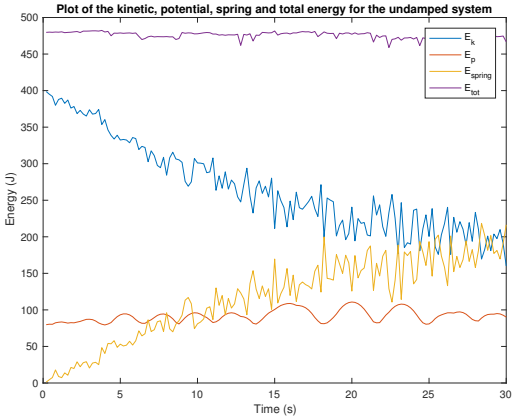

Figure 10 – Graph showing the energy quantities for the undamped system.

Question (e):

Finely the velocity of the object is to be calculated. The question to be answered is whether or not the velocity changes linearly over time. This also leads to the question of friction and whether or not it is dependent of the initial velocity.

Answer:

As can be seen in figure (11) the velocity of the objects center of mass is decreasing linearly as time goes on. This shows that the the model simulates a constant friction. The frictional coefficient μ is given by equation (2) and its value dose not depend seriously on the the initial velocity as can be seen in table (1).

$$F_f = \mu N \tag{2}$$

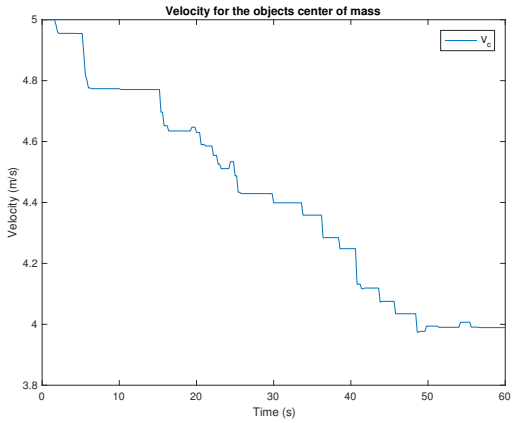

Figure 11 – Graph showing velocity for the objects center of mass.

Table 1 – Friction coefficient for different initial velocities.

Friction coefficients	
Velocity	Friction coefficient
3	0.057845
5	0.060793
7	0.049066

1.3 Exercise 3: Modelling a macroscopic 2D object

This time the goal is to simulate a macroscopic object sliding on an rugged surface. The program from the precious exercise is still being used because it still is a satisfying model even though the what earlier represented the particles now represents a much larger part of the object witch in turn is made up by millions and millions of particles.

Question (a):

The aim is to repeat the simulation from Exercise 3 but now changing the dampening to $\kappa_d = 5$.

Answer:

Figure (12) shows that the E_{tot} is no longer constant. This is because there is no longer anything keeping track of the heat.

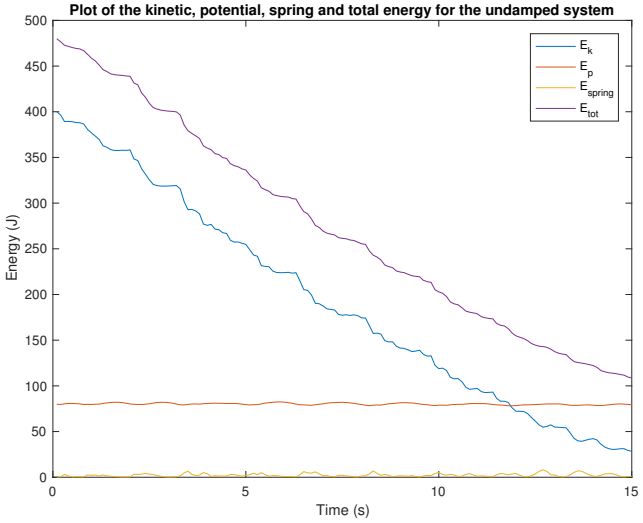

Figure 12 – Graph showing the energy quantities for the undamped system with $\kappa_d = 5$.

Question (b):

Finally the goal is to produce a simulation whit the following parameter values; $m = 1$, $g = 10$, $\kappa_s = 1000$, $\kappa_d = 50$, $L = 1$, $R = (3/4)L$, $v = 10$ and $\Delta t = 0,005$.

Answer:

As can be seen in figure (14) the velocity of the object is decreasing as time goes on. Because of the dampening the kinetic energy is transformed in to heat and so the object is losing speed as time goes on. The fictional coefficient with initial velocity $v = 5$ is $\mu = 0.033285$.

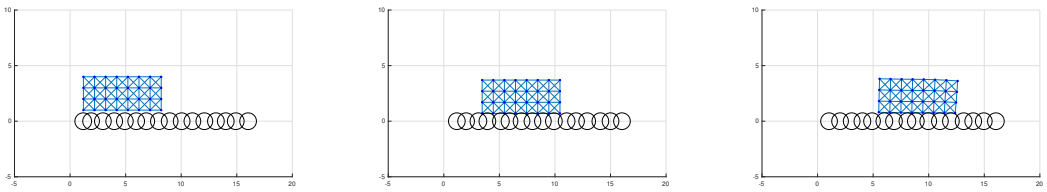

Figure 13 – Snap shots of the macroscopic object for three different time steps.

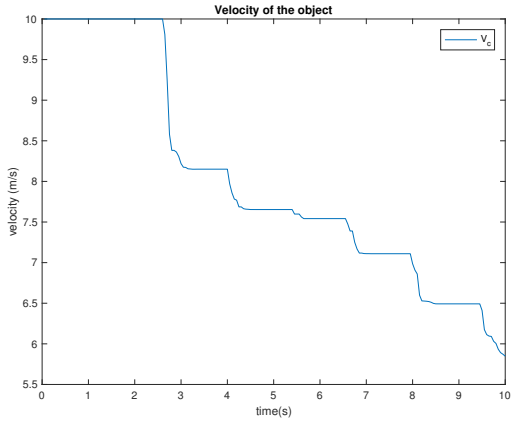

Figure 14 – Graph showing the objects velocity plotted against time.

A Appendix

Ecercise 2c

Matlab Code

```
close all; clear all;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
L = 1;                          % Spring length
c = 3;                          % Columns

steps = 1500;                   % Number of steps
modulus = 10;                   % Get energy every "modulus" step
deltat = 0.004;                 % Step size

%%% Particles %%%
NP = 2;                         % Number of particles
m = 1; M = eye(NP);             % Particle masses
g = 0;                          % Gravity

%%% Pre-allocate memory %%%
X = zeros(NP,c, steps);
V = zeros(NP,c,steps);
BOLL_a = zeros(1,NP);
E_k = zeros(1,steps/modulus - 1);
E_p = zeros(1,steps/modulus - 1);
E_s = zeros(1,steps/modulus - 1);
E_tot = zeros(1,steps/modulus - 1);
springLength = zeros(1,steps/modulus-1);

%%% Inital position %%%
X1 = [0, 0, 0];
X2 = [1.8, 0, 0];
X(:,:,1) = [X1; X2];

%%% Initial velocity %%%
V1 = [0, 0, 0];
V2 = [0, 0, 0];
V(:,:,1) = [V1; V2];


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% Spring struct %%%
numsprings = 1;                          % Number of springs
KS = 10;                          % Spring coeficcient
KD = 0.5;                         % Damping coeficcient

spring_number = 0;
for k = 1:numsprings                                                    %
    Loop over springs
     particle_number = k;
     spring_number = spring_number + 1;
     spring(spring_number).from = particle_number;    % Number of
        hte "from" particle
     spring(spring_number).to = particle_number + 1; % Number of
        hte "to" particle
     spring(spring_number).length = L;                % Spring
        rest length
     spring(spring_number).KS = KS;                   % Spring
        coefficient
     spring(spring_number).KD = KD;                   % Damping
        coefficient
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Parametervalues for the circle (ball)
x = [-0.5  0]; y = [0  0.5]; % inital position
Ra = 0.06;                   % Radius of the circle

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

a = spring(1).from; b = spring(1).to;

r = X(a,:,1) - X(b,:,1);              % Relative distance
v = V(a,:,1) - V(b,:,1);               % Relative velocity
KS = spring(1).KS; KD = spring(1).KD;

dis_normA = r/norm(r);
dis_normB = -dis_normA;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Half Euler-step
F_old = -(KS*(norm(r)-L) + KD*dot(v,r)/norm(r)) * [dis_normA;
    dis_normB];
V = V(:,:,1) - F_old*deltat/2;          % Half Euler-step

Ve_a = V(:,:,1) - F_old*deltat/2;       % Half Euler-step
Ve_b = V(:,:,1) - F_old*deltat/2;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Graphic-window
daspect([1,1,1]);                   % Scale the axis equal
axis([-2,4,-2,4]);                  % Interval of the axis
hold on; grid on                    % Lock the installations

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Particles as complete circles

for i = 1:NP
BOLL_a(i)=rectangle('Position',[x(i)-Ra,y(i)-Ra,2*Ra,2*Ra],...
    'Curvature',[1,1],'EdgeColor','b','FaceColor','b');
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

xl = [X(:,1,1)', X(1,1,1)];
yl = [X(:,2,1)', X(1,2,1)];
lines = line(xl,yl);
points = plot(X(:,1,1), X(:,2,1));

index = 1;

%%% Main loop %%%

for n = 1:steps-1
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Patricels (a and b) that are connected by spring number 1:
    a = spring(1).from; b = spring(1).to;
```

```
r  = X(a,:,n) - X(b,:,n);      % Relative distance
v = V(a,:,n) - V(b,:,n);       % Relative velocity
KS = spring(1).KS;
KD = spring(1).KD;

dis_normA = r/norm(r);
dis_normB = -dis_normA;

%%% Spring length %%%
L = spring(1).length;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
F = -(KS*(norm(r)-L) + KD*dot(v,r)/norm(r)) * [dis_normA;
    dis_normB];
V(:,:,n+1) = V(:,:,n) + deltat*inv(M)*F;              % Update
    the velocity
X(:,:,n+1) = X(:,:,n) + deltat*V(:,:,n+1);            % Update
    the position
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% Plot %%%
pause(0.001);
xl = [X(:,1,n)',X(1,1,n)];
yl = [X(:,2,n)', X(1,2,n)];

%%% Update of the circles every step %%%
for j = 1:NP
set(BOLL_a(j),'Position',[X(j,1,n)-Ra,X(j,2,n)-Ra,2*Ra,2*Ra
    ]);
end

set(lines, 'XData', xl, 'YData', yl);
set(points, 'XData', X(:,1,n), 'YData', X(:,2,n));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% Update energy every "modulus" step %%%
if ~mod(n,modulus)
    Ve_a = 1/2*(V(1,:,n+1) + V(1,:,n));
    Ve_b = 1/2*(V(2,:,n+1) + V(2,:,n));
```

```
        E_k(index) = (1/2)*M(1,1)*(norm(Ve_a))^2 + (1/2)*M(2,2)
            *(norm(Ve_b))^2;
        E_p(index) = g*m*norm(r);
        E_s(index) = (1/2)*KS*(norm(r) - L)^2;
        E_tot(index) = E_k(index) + E_p(index) + E_s(index);

        springLength(index) = norm(r);
        %%% calcuata spring extention (amplitude) %%%
        springAmplitude(index) = norm(r)-L;

        index = index + 1;
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

end

%%% Time steps %%%
timet = 1:index-1;
time = timet.*10*deltat;

%%% Calculet the time to the energy amplitude is less then 10%
    %%%
E_max = max(E_tot);
for i = 1:index-1
    if abs(E_tot(i)/E_max) <= 0.01
        fprintf('Time = %.2f\n',i*deltat*10);
        break
    end
end


%%% Ploting energies %%%
figure
plot(time, E_k, time, E_p, time, E_s, time, E_tot)
legend('E_k', 'E_p', 'E_{spring}','E_{tot}')
xlabel('Time (s)')
ylabel('Energy (J)')
figure
title('Plot of the kinetic, potential, spring and total energy
    for the damped system')
plot(time, E_tot)
legend('E_{tot}')
title('Total Energy for the damped system')
```

```
xlabel('Time (s)')
ylabel('Energy (J)')

%%% Plotting spring length %%%
figure
xlabel('time (s)'); ylabel('springlength (m)')
plot(time, springLength)
```

Ecercise 2d

Matlab Code

```
close all; clear all;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
L = 1;                          % Spring length
c = 3;                          % Columns

steps = 1500;                   % Number of steps
modulus = 1;                    % Get energy every "modulus" step
deltat = 0.01;                  % Step size

%%% particles %%%
NP = 2;                         % Number of particles
m = 1; M = eye(NP);             % Particle masses
g = 0;                          % Gravity

%%% Pre-allocate memory %%%
X = zeros(NP,c, steps);
V = zeros(NP,c,steps);
BOLL_a = zeros(1,NP);
E_k = zeros(1,steps/modulus - 1);
E_p = zeros(1,steps/modulus - 1);
E_spring = zeros(1,steps/modulus - 1);
E_tot = zeros(1,steps/modulus - 1);
springL = zeros(1,steps/modulus-1);
ang_m = zeros(1,steps/modulus-1);
angMom = zeros(1,steps);
angFreq = zeros(1,steps-1);

%%% Initial postition %%%
X1 = [0, 0, 0];
X2 = [1.8, 0, 0];
X(:,:,1) = [X1; X2];


%%% Initial velocity %%%
V1 = [0, -5, 0];
V2 = [0, 5, 0];
V(:,:,1) = [V1; V2];
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% Struct spring %%%
NS = 1;                         % Number of springs
KS = 10;                        % Spring coeficcient
KD = 0;                         % Damping coeficcient

spring_number = 0;
for k = 1:NS                                                % Loop over
    springs
     particle_number = k;
     spring_number = spring_number + 1;
     spring(spring_number).from = particle_number;          % Number of
        hte "from" particle
     spring(spring_number).to = particle_number + 1;        % Number of
        hte "to" particle
     spring(spring_number).length = L;                      % Spring
        rest length
     spring(spring_number).KS = KS;                         % Spring
        coefficient
     spring(spring_number).KD = KD;                         % Damping
        coefficient
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% Parametervalues for the circle (ball)

x = [-0.5  0]; y = [0  0.5]; % inital position of the center
Ra = 0.06;                   % Radius of the circle

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
a = spring(1).from;
b = spring(1).to;

xa = X(a,:,1);
ya = X(a,:,1);
xb = X(b,:,1);
yb = X(b,:,1);
```

```
va = V(a,:,1);
vb = V(b,:,1);

r  = xa - xb;     % Relative distance
v = va - vb;      % Relative velocity
KS = spring(1).KS;
KD = spring(1).KD;

dis_normA = r/norm(r);
dis_normB = -dis_normA;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Half Euler-step
F_old = -(KS*(norm(r)-L) + KD*dot(v,r)/norm(r)) * [dis_normA;
    dis_normB];  % Force without damping
V = V(:,:,1) - F_old*deltat/2;        % Half Euler-step
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Ve_a = V(:,:,1) - F_old*deltat/2;        % Half Euler-step
Ve_b = V(:,:,1) - F_old*deltat/2;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Graphic-window
daspect([1,1,1]);              % Scale the axis equal
axis([-2,4,-2,4]);             % Interval of the axis
hold on; grid on               % Lock the installations
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Particles as complete circles

for i = 1:NP
BOLL_a(i)=rectangle('Position',[x(i)-Ra,y(i)-Ra,2*Ra,2*Ra],...
     'Curvature',[1,1],'EdgeColor','b','FaceColor','b');
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

xl = [X(:,1,1)', X(1,1,1)];
yl = [X(:,2,1)', X(1,2,1)];
lines = line(xl,yl);
points = plot(X(:,1,1), X(:,2,1));
```

```
index = 1;

% loop = [1:steps]*deltat;

for n = 1:steps-1
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Patricels (a and b) that are connected by spring number 1:
    a = spring(1).from; b = spring(1).to;

    r  = X(a,:,n) - X(b,:,n);      % Relative distance
    v = V(a,:,n) - V(b,:,n);       % Relative velocity
    KS = spring(1).KS;
    KD = spring(1).KD;

    dis_normA = r/norm(r);
    dis_normB = -dis_normA;

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    F = -(KS*(norm(r)-L) + KD*dot(v,r)/norm(r)) * [dis_normA;
        dis_normB];
    V(:,:,n+1) = V(:,:,n) + deltat*inv(M)*F;                % Update
        the velocity
    X(:,:,n+1) = X(:,:,n) + deltat*V(:,:,n+1);              % Update
        the position

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%% Plot %%%
    pause(0.001);
    xl = [X(:,1,n)',X(1,1,n)];
    yl = [X(:,2,n)', X(1,2,n)];

    %%% Update circles every step %%%
    for j = 1:NP
    set(BOLL_a(j),'Position',[X(j,1,n)-Ra,X(j,2,n)-Ra,2*Ra,2*Ra
        ]);
    end

    set(lines, 'XData', xl, 'YData', yl);
    set(points, 'XData', X(:,1,n), 'YData', X(:,2,n));
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %%% Update energy every "modulus" step %%%
```

```
    if ~mod(n,modulus)
        Ve_a = 1/2*(V(a,:,n+1) + V(a,:,n)); Ve_b = 1/2*(V(b,:,n
            +1) + V(b,:,n));

        E_k(index) = (1/2)*M(a,a)*(norm(Ve_a))^2 + (1/2)*M(b,b)
            *(norm(Ve_b))^2;
        E_p(index) = g*m*norm(r);
        E_spring(index) = (1/2)*KS*(norm(r) - L)^2;
        E_tot(index) = E_k(index) + E_p(index) + E_spring(index)
            ;
        springL(index) = norm(r);
        index = index + 1;
    end

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   %%% Calculate spring length %%%
    springL(n) = norm(r);

    %%% Calculate angular momentum %%%
    xOrb = 0.9; yOrb = 0; I = 0;
    for j = 1:NP
        angMom(n) = angMom(n) + M(j,j)*((X(j, 1, n)-xOrb)*V(j,
            2, n) - (X(j, 2, n)-yOrb)*V(j, 1, n));
        I = I + M(j,j) *(norm(X(j, 1, n) - xOrb)^2 + norm(X(j,
            2, n) - xOrb)^2);
    end

     %%% Calculating angular frequency %%%
    angFreq(n) = angMom(n)/I;

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% time steps %%%
time_t = 1:index-1;
time = time_t.*10*deltat;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% Plots %%%

%%% Plotting energys %%%
figure
plot(time, E_k)
hold on
plot(time, E_p)
hold on
plot(time, E_spring)
hold on
plot(time, E_tot)
legend('E_k', 'E_p', 'E_{spring}','E_{tot}')
title('Plot of the kinetic, potential, spring and total energy
    for the undamped system')


%%% Plotting total energy %%%
figure
plot(time, E_tot)
legend('E_{tot}')
title('Total Energy for the undamped system')

%%% Plotting spring length and angular frequency %%%
figure
plot(time, springL(time_t))
hold on
plot(time, angFreq(time_t))
xlabel('Time (s)');
legend('Spring length', 'Angular frequency');
title('Spring length and angular frequency')
hold off

%%% Plotting angular momentum %%%
figure
plot(time, angMom(time_t))
xlabel('time (s)'); ylabel('Angular momentum');
title('Angular Momentum')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Ecercise 3

Matlab Code

```
close all; clear all;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

L = 1;                                  % Spring length
dim = 3;
Nr = 4;                                 % Columns
Nc = 8;                                 % Rows

steps = 1500;                           % Number of steps
modulus = 10;                           % Get energy every "modulus"
    step
deltat = 0.01;                          % Step size

%%% particles %%%
NP = Nr*Nc;                             % Number of particles
m = 1; M = eye(NP);                     % Particle masses
g = 1;                                  % Gravity

%%% springs %%%
numsprings = 4*Nr*Nc - 3*(Nr+Nc) + 2;   % Number of springs
KS = 100;                               % Spring coeficcient
KD = 5;                                 % Damping coeficcient

%%% circels %%%
NB = 100;                               % Number of big circles
R = L;
Rsmall = 0.06;                          % Radius small circles
Rbig = R;                               % Radius big circles

%%% Pre-allocate memory %%%
X = zeros(NP,dim, steps);
V_update = zeros(1,steps);
V = zeros(NP,dim,steps);
Vcenter = zeros(1,steps);
lines=zeros(1,numsprings);
BOLL_a = zeros(1,NP); BOLL_b = zeros(1,NB); c = zeros(NB,3);
E_k = zeros(1,steps/modulus - 1);
E_p = zeros(1,steps/modulus - 1);
```

```
E_s = zeros(1,steps/modulus - 1);
E_tot = zeros(1,steps/modulus - 1);

%%% Initial velocity %%%
V0 = 5;
V(:,1,1) = V0;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% Position and velocity small circles %%%
particle_number = 1;
for row = 1:Nr
    for col = 1 : Nc
        X(particle_number, 1, 1) = col*L;
        X(particle_number, 2, 1) = row*L;
        particle_number = particle_number + 1;
    end
end

%%% Position and velocity big circles %%%
particle_number = 1;
for position = 1:NB
        Xb(particle_number, 1, 1) = (0.1*R*randn) + position;
        Xb(particle_number, 2, 1) = 0;
        particle_number = particle_number + 1;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

spring = springs(KS,KD,L,Nr,Nc, numsprings);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for s = 1:numsprings
    a = spring(s).from; b = spring(s).to;
    lines(s) = line([X(a,1,1),X(b,1,1)],[X(a,2,1),X(b,2,1)]);
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%% Half Eulerstep (leapfrog) %%%
F_old = force(spring,X(:,:,1), V(:,:,1), NP, g, dim);
V(:,:,1) = V(:,:,1) - M*F_old*deltat/2;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%% Graphics %%%%
daspect([1,1,1]);
axis([-5,100,-5,10]);
title('Simulation of the system'); xlabel('x'); ylabel('y');
hold on;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%% Circles (small) %%%%
for i = 1:NP
    BOLL_a(i) = rectangle('Position',[X(i,1,1)-Rsmall,X(i,2,1)-
        Rsmall,2*Rsmall,2*Rsmall],...
         'Curvature',[1,1],'EdgeColor','b','FaceColor','k');
end

for i = 1:NB
    c(i) = (i-1)+0.1*R*randn;
    BOLL_b(i) = rectangle('Position',[Xb(i,1,1)-Rbig,Xb(i,2,1)-
        Rbig,2*Rbig,2*Rbig],...
         'Curvature',[1,1],'EdgeColor','k');
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


index = 1;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% main loop %%%
for n = 1:steps

    %%% Calculate force %%%
    F = force(spring,X(:,:,n), V(:,:,n), NP, g, dim);

    %%% Velocity uppdate %%%
    V(:,:,n+1) = V(:,:,n) + deltat*inv(M)*F;

    %%% Position uppdate %%%
    X(:,:,n+1) = X(:,:,n) + deltat*V(:,:,n+1);

    %%% Calculate the new velocity after hitting the floor %%%
    [X(1:NP,:,n+1),V(:,:,n+1)] = floorHit(X(1:NP,:,n+1), V(:,:,n
        +1), c, R, NB, NP);
    V_update = (1/2)*(V(:,:,n+1) + V(:,:,n));

    %%% Center of mass %%%
    mass_sum = sum(M*ones(NP,1));
    Vcenter(n) = sum(M*V(:,1,n))/mass_sum;

    % Plot
    pause(0.00001);

    %%% Update circles (small) %%%
    for j = 1:NP
        set(BOLL_a(j),'Position',[X(j,1,n)-Rsmall,X(j,2,n)-
            Rsmall,2*Rsmall,2*Rsmall]);
    end

    for s = 1:numsprings
        a = spring(s).from;
        b = spring(s).to;

        set(lines(s),'XData', [X(a,1,n),X(b,1,n)], 'YData', [X(a
            ,2,n),X(b,2,n)]);
    end
```

```
    %%% Update energy every "modulus" step %%%
    if ~mod(n,modulus)
        [E_k(index), E_p(index), E_s(index)] = energy(X(:,:,n),
            V_update, spring, M, g, NP, numsprings);
        E_tot(index) = E_k(index) + E_p(index) + E_s(index);
        index = index + 1;
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Time steps %%%
timet = 1:index-1;
time = (timet.*10)*deltat;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% calculate frictenal coeficient %%%
tot_mass = sum(sum(M));
F_mass = tot_mass*(Vcenter(end) - Vcenter(1))/time(end);
my = F_mass/(tot_mass*-g);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% Plotting energys %%%
figure
plot(time, E_k(timet))
hold on
plot(time, E_p(timet))
hold on
plot(time, E_s(timet))
hold on
plot(time, E_tot(timet))
legend('E_k', 'E_p', 'E_{spring}','E_{tot}')
title('Plot of the kinetic, potential, spring and total energy
    for the undamped system')
xlabel('Time (s)'); ylabel('Energy (J)')
```

```
%%% Plotting total energy %%%
figure
plot(time, E_tot(timet))
legend('E_{tot}')
title('Total Energy for the undamped system')

%%% Plotting velocity of center of mass %%%
figure
plot(time, Vcenter(timet))
legend('V_c')
title('Velocity for the objects center of mass')
xlabel('Time (s)'); ylabel('Velocity (m/s)')

%%% print of the frictional coeficient %%%
fprintf('Friction coefficient with %.1f as initial velocity: %f\
    n', V0, my);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```