Table of Contents


```
clear all; close all; clc
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                               %
%                         Assignment 1                          %
%                   Modelling and simulation                    %
%        Written by Johannes Lundahl and Daniel Söderqvist      %
%                       14 september 2023                       %
%                                                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Define symbolic variables

```
syms p1 x1 y1 z1 theta phi l m1 m2 g real
syms x2 y2 z2 x2_dot y2_dot z2_dot x2_dotdot y2_dotdot z2_dotdot Z real
syms x1_dot y1_dot z1_dot theta_dot phi_dot real
syms x1_dotdot y1_dotdot z1_dotdot theta_dotdot phi_dotdot real
syms ux uy uz real
```

1. a)

Define positions and statevector

```
p1 = [x1; y1; z1];
```

```
p1_dot = [x1_dot; y1_dot; z1_dot];

p1_dotdot = [x1_dotdot; y1_dotdot; z1_dotdot];

q = [p1; theta; phi];

q_dot = [p1_dot; theta_dot; phi_dot];

q_dotdot= [p1_dotdot; theta_dotdot; phi_dotdot];

p2 = p1 + l*[sin(q(5))*cos(q(4)); sin(q(4))*sin(q(5)); -cos(q(5))];
```

Derivative of the positions using jacobian()

```
dp2_dq = jacobian(p2,q);

dp1_dq = jacobian(p1,q);
```

Potential and Kinetic energy & Lagrange

```
w = simplify(m1*dp1_dq'*dp1_dq + m2*dp2_dq'*dp2_dq); % Masses combined with
 p_dot

T = simplify((1/2)*q_dot'*w*q_dot); % Kinetic energy

V1 = m1*g*[0 0 1]*p1; % Potential energy mass 1
V2 = m2*g*[0 0 1]*p2; % Potential energy mass 2
V = simplify(V1 + V2); % Total Potential energy

L = T - V; %Lagrange
```

Euler-Lagrange

```
grad_q_dot_L = jacobian(L,q_dot)';

grad_q_L = jacobian(L,q)';

d_dt_gradq_dotL =  simplify(jacobian(grad_q_dot_L,q_dot)*q_dotdot +
 jacobian(grad_q_dot_L,q)*q_dot);

EL = simplify(jacobian(grad_q_dot_L,q_dot)*q_dotdot +
 jacobian(grad_q_dot_L,q)*q_dot - grad_q_L);
Q = [ux; uy; uz; 0; 0]; % External forces
```

On the Mq = b form and printing to command window

```
fprintf('Answer to question 1. a) = \n')
M = simplify(jacobian(grad_q_dot_L,q_dot))
b = simplify(Q + grad_q_L + jacobian(grad_q_dot_L,q)*q_dot)
```

```
Answer to question 1. a) =

M =

[                  m1 + m2,                              0,              0, -
l*m2*sin(phi)*sin(theta), l*m2*cos(phi)*cos(theta)]
[                        0,                        m1 + m2,              0,
 l*m2*cos(theta)*sin(phi), l*m2*cos(phi)*sin(theta)]
[                        0,                              0,        m1 + m2,
               0,            l*m2*sin(phi)]
[-l*m2*sin(phi)*sin(theta), l*m2*cos(theta)*sin(phi),              0,  -
l^2*m2*(cos(phi)^2 - 1),                        0]
[ l*m2*cos(phi)*cos(theta), l*m2*cos(phi)*sin(theta), l*m2*sin(phi),
               0,                  l^2*m2]


b =


                                       - l*m2*cos(theta)*sin(phi)*phi_dot^2
 - 2*l*m2*cos(phi)*sin(theta)*phi_dot*theta_dot -
 l*m2*cos(theta)*sin(phi)*theta_dot^2 + ux

                                       - l*m2*sin(phi)*sin(theta)*phi_dot^2
 + 2*l*m2*cos(phi)*cos(theta)*phi_dot*theta_dot -
 l*m2*sin(phi)*sin(theta)*theta_dot^2 + uy


                                           l*m2*cos(phi)*phi_dot^2 + uz - g*m1
 - g*m2

 -2*l*m2*(phi_dot*x1_dot*cos(phi)*sin(theta) +
 theta_dot*x1_dot*cos(theta)*sin(phi) + theta_dot*y1_dot*sin(phi)*sin(theta) -
 phi_dot*y1_dot*cos(phi)*cos(theta) - l*phi_dot*theta_dot*cos(phi)*sin(phi))
-l*m2*(g*sin(phi) - 2*phi_dot*z1_dot*cos(phi) +
 2*phi_dot*x1_dot*cos(theta)*sin(phi) + 2*theta_dot*x1_dot*cos(phi)*sin(theta)
 + 2*phi_dot*y1_dot*sin(phi)*sin(theta) - l*theta_dot^2*cos(phi)*sin(phi) -
 2*theta_dot*y1_dot*cos(phi)*cos(theta))
```

1. b)

Define new positions and statevector

```
p1 = [x1; y1; z1];

p1_dot = [x1_dot; y1_dot; z1_dot];

p1_dotdot = [x1_dotdot; y1_dotdot; z1_dotdot];

p2 = [x2; y2; z2];
```

```
p2_dot = [x2_dot; y2_dot; z2_dot];

p2_dotdot = [x2_dotdot; y2_dotdot; z2_dotdot];

q = [p1; p2];

q_dot = [p1_dot; p2_dot];

q_dotdot= [p1_dotdot; p2_dotdot];
```

Adding constraints

```
E = p1 - p2;
C = 1/2*(E'*E - l^2);
```

Derivative of the positions

```
dp2_dq = jacobian(p2, q);

dp1_dq = jacobian(p1, q);
```

Potential and Kinetic energy & Lagrange

```
w = simplify(m1*dp1_dq'*dp1_dq + m2*dp2_dq'*dp2_dq);

T = simplify((1/2)*q_dot'*w*q_dot);

V1 = m1*g*[0 0 1]*p1;
V2 = m2*g*[0 0 1]*p2;
V = simplify(V1 + V2);

L = T - V - Z*C;
```

Euler-Lagrange

```
grad_q_dot_L = jacobian(L,q_dot)';

grad_q_L = jacobian(L,q)';

d_dt_gradq_dotL =  simplify(jacobian(grad_q_dot_L,q_dot)*q_dotdot +
 jacobian(grad_q_dot_L,q)*q_dot);

EL = simplify(jacobian(grad_q_dot_L,q_dot)*q_dotdot +
 jacobian(grad_q_dot_L,q)*q_dot - grad_q_L);
Q = [ux; uy; uz; 0; 0; 0];
```

On the Mq = b form and printing to command window

```
fprintf('\n\n\nAnswer to question 1. b) = \n')
```

```
M = simplify(jacobian(grad_q_dot_L,q_dot))
b = simplify(Q + grad_q_L + jacobian(grad_q_dot_L,q)*q_dot)
fprintf('\n\n\nWe can see that by using constraints we get a much simpler and
 user-firendly expressions and matrices than if we dont.')
```

```
Answer to question 1. b) =

M =

[m1,  0,  0,  0,  0,  0]
[ 0, m1,  0,  0,  0,  0]
[ 0,  0, m1,  0,  0,  0]
[ 0,  0,  0, m2,  0,  0]
[ 0,  0,  0,  0, m2,  0]
[ 0,  0,  0,  0,  0, m2]


b =

       ux - Z*(x1 - x2)
       uy - Z*(y1 - y2)
uz - g*m1 - Z*(z1 - z2)
            Z*(x1 - x2)
            Z*(y1 - y2)
     Z*(z1 - z2) - g*m2



We can see that by using constraints we get a much simpler and user-firendly
 expressions and matrices than if we dont.
```

2. a)

Defining a and c implicit form of euler-lagrange equation using and printing to command window

```
fprintf('\n\n\nThe answer to question 2. a) =\n')
a = simplify(jacobian(C, q)')

d_w_q_qdot = simplify(jacobian(grad_q_dot_L,q)*q_dot);
b_second = simplify(-jacobian(jacobian(C, q)*q_dot, q)*q_dot);
c = simplify([Q - d_w_q_qdot + jacobian(T,q)' - jacobian(V,q)'; b_second])
```

```
The answer to question 2. a) =

a =

x1 - x2
y1 - y2
z1 - z2
x2 - x1
y2 - y1
z2 - z1


c =


                                                ux

                                                uy

                                       uz - g*m1

                                                 0

                                                 0

                                            -g*m2
- x1_dot^2 + 2*x1_dot*x2_dot - x2_dot^2 - y1_dot^2 + 2*y1_dot*y2_dot -
 y2_dot^2 - z1_dot^2 + 2*z1_dot*z2_dot - z2_dot^2
```

2. b)

Trying to take the inverse of the M_q matrix

```
fprintf('\n\n\nThe answer to question 2. b) =\nWe can see from the following
 inverse that its yielding a pretty complicated and complex expression that
\n takes alot of computional power. Thats why its better to use the explicit
 form.\n')
M_q = simplify([w, a; a', zeros(size(a', 1), size(w, 2) + size(a, 2) -
 size(a', 2))]);
inv_M_q = simplify(pinv(M_q))
```

```



The answer to question 2. b) =
We can see from the following inverse that its yielding a pretty complicated
 and complex expression that
 takes alot of computional power. Thats why its better to use the explicit
 form.

inv_M_q =
```

```
[(m1*x1^2 + m1*x2^2 + m1*y1^2 + m1*y2^2 + m2*y1^2 + m2*y2^2 + m1*z1^2
 + m1*z2^2 + m2*z1^2 + m2*z2^2 - 2*m1*x1*x2 - 2*m1*y1*y2 - 2*m2*y1*y2 -
 2*m1*z1*z2 - 2*m2*z1*z2)/(m1*(m1 + m2)*(x1^2 - 2*x1*x2 + x2^2 + y1^2 -
 2*y1*y2 + y2^2 + z1^2 - 2*z1*z2 + z2^2)),

                              -(m2*(x1 - x2)*(y1 - y2))/(m1*(m1 + m2)*(x1^2 -
 2*x1*x2 + x2^2 + y1^2 - 2*y1*y2 + y2^2 + z1^2 - 2*z1*z2 + z2^2)),

                                                      -(m2*(x1 - x2)*(z1 - z2))/
(m1*(m1 + m2)*(x1^2 - 2*x1*x2 + x2^2 + y1^2 - 2*y1*y2 + y2^2 + z1^2 - 2*z1*z2
 + z2^2)),

               (x1 - x2)^2/((m1 + m2)*(x1^2 - 2*x1*x2 + x2^2 + y1^2 - 2*y1*y2
 + y2^2 + z1^2 - 2*z1*z2 + z2^2)),

                           ((x1 - x2)*(y1 - y2))/((m1 + m2)*(x1^2 - 2*x1*x2 +
 x2^2 + y1^2 - 2*y1*y2 + y2^2 + z1^2 - 2*z1*z2 + z2^2)),

                                                   ((x1 - x2)*(z1 - z2))/((m1 +
 m2)*(x1^2 - 2*x1*x2 + x2^2 + y1^2 - 2*y1*y2 + y2^2 + z1^2 - 2*z1*z2 + z2^2)),
  (m2*(x1 - x2))/((m1 + m2)*(x1^2 - 2*x1*x2 + x2^2 + y1^2 - 2*y1*y2 + y2^2 +
 z1^2 - 2*z1*z2 + z2^2))]
[
                                                                  -(m2*(x1 -
 x2)*(y1 - y2))/(m1*(m1 + m2)*(x1^2 - 2*x1*x2 + x2^2 + y1^2 - 2*y1*y2 + y2^2
 + z1^2 - 2*z1*z2 + z2^2)), (m1*x1^2 + m1*x2^2 + m2*x1^2 + m2*x2^2 + m1*y1^2
 + m1*y2^2 + m1*z1^2 + m1*z2^2 + m2*z1^2 + m2*z2^2 - 2*m1*x1*x2 - 2*m2*x1*x2
 - 2*m1*y1*y2 - 2*m1*z1*z2 - 2*m2*z1*z2)/(m1*(m1 + m2)*(x1^2 - 2*x1*x2 + x2^2
 + y1^2 - 2*y1*y2 + y2^2 + z1^2 - 2*z1*z2 + z2^2)),

                                          -(m2*(y1 - y2)*(z1 - z2))/(m1*(m1 +
 m2)*(x1^2 - 2*x1*x2 + x2^2 + y1^2 - 2*y1*y2 + y2^2 + z1^2 - 2*z1*z2 + z2^2)),

                                                                       ((x1 -
 x2)*(y1 - y2))/((m1 + m2)*(x1^2 - 2*x1*x2 + x2^2 + y1^2 - 2*y1*y2 + y2^2 +
 z1^2 - 2*z1*z2 + z2^2)),

                              (y1 - y2)^2/((m1 + m2)*(x1^2 - 2*x1*x2 + x2^2
 + y1^2 - 2*y1*y2 + y2^2 + z1^2 - 2*z1*z2 + z2^2)),

                                                  ((y1 - y2)*(z1 - z2))/((m1 +
 m2)*(x1^2 - 2*x1*x2 + x2^2 + y1^2 - 2*y1*y2 + y2^2 + z1^2 - 2*z1*z2 + z2^2)),
  (m2*(y1 - y2))/((m1 + m2)*(x1^2 - 2*x1*x2 + x2^2 + y1^2 - 2*y1*y2 + y2^2 +
 z1^2 - 2*z1*z2 + z2^2))]
[
                                                                  -(m2*(x1 -
 x2)*(z1 - z2))/(m1*(m1 + m2)*(x1^2 - 2*x1*x2 + x2^2 + y1^2 - 2*y1*y2 + y2^2
 + z1^2 - 2*z1*z2 + z2^2)),

                 -(m2*(y1 - y2)*(z1 - z2))/(m1*(m1 + m2)*(x1^2 - 2*x1*x2 +
 x2^2 + y1^2 - 2*y1*y2 + y2^2 + z1^2 - 2*z1*z2 + z2^2)), (m1*x1^2 + m1*x2^2 +
 m2*x1^2 + m2*x2^2 + m1*y1^2 + m1*y2^2 + m2*y1^2 + m2*y2^2 + m1*z1^2 + m1*z2^2
 - 2*m1*x1*x2 - 2*m2*x1*x2 - 2*m1*y1*y2 - 2*m2*y1*y2 - 2*m1*z1*z2)/(m1*(m1 +
 m2)*(x1^2 - 2*x1*x2 + x2^2 + y1^2 - 2*y1*y2 + y2^2 + z1^2 - 2*z1*z2 + z2^2)),
```

```
                                                                                ((x1 -
 x2)*(z1 - z2))/((m1 + m2)*(x1^2 - 2*x1*x2 + x2^2 + y1^2 - 2*y1*y2 + y2^2 +
 z1^2 - 2*z1*z2 + z2^2)),

                    ((y1 - y2)*(z1 - z2))/((m1 + m2)*(x1^2 - 2*x1*x2 + x2^2
 + y1^2 - 2*y1*y2 + y2^2 + z1^2 - 2*z1*z2 + z2^2)),

                                                                  (z1 - z2)^2/((m1 +
 m2)*(x1^2 - 2*x1*x2 + x2^2 + y1^2 - 2*y1*y2 + y2^2 + z1^2 - 2*z1*z2 + z2^2)),
  (m2*(z1 - z2))/((m1 + m2)*(x1^2 - 2*x1*x2 + x2^2 + y1^2 - 2*y1*y2 + y2^2 +
 z1^2 - 2*z1*z2 + z2^2))]
[

  (x1 - x2)^2/((m1 + m2)*(x1^2 - 2*x1*x2 + x2^2 + y1^2 - 2*y1*y2 + y2^2 + z1^2
 - 2*z1*z2 + z2^2)),

             ((x1 - x2)*(y1 - y2))/((m1 + m2)*(x1^2 - 2*x1*x2 + x2^2 + y1^2
 - 2*y1*y2 + y2^2 + z1^2 - 2*z1*z2 + z2^2)),

                                      ((x1 - x2)*(z1 - z2))/((m1 + m2)*(x1^2 -
 2*x1*x2 + x2^2 + y1^2 - 2*y1*y2 + y2^2 + z1^2 - 2*z1*z2 + z2^2)), (m2*x1^2 +
 m2*x2^2 + m1*y1^2 + m1*y2^2 + m2*y1^2 + m2*y2^2 + m1*z1^2 + m1*z2^2 + m2*z1^2
 + m2*z2^2 - 2*m2*x1*x2 - 2*m1*y1*y2 - 2*m2*y1*y2 - 2*m1*z1*z2 - 2*m2*z1*z2)/
(m2*(m1 + m2)*(x1^2 - 2*x1*x2 + x2^2 + y1^2 - 2*y1*y2 + y2^2 + z1^2 - 2*z1*z2
 + z2^2)),
                                                                               -
(m1*(x1 - x2)*(y1 - y2))/(m2*(m1 + m2)*(x1^2 - 2*x1*x2 + x2^2 + y1^2 - 2*y1*y2
 + y2^2 + z1^2 - 2*z1*z2 + z2^2)),

                    -(m1*(x1 - x2)*(z1 - z2))/(m2*(m1 + m2)*(x1^2 - 2*x1*x2
 + x2^2 + y1^2 - 2*y1*y2 + y2^2 + z1^2 - 2*z1*z2 + z2^2)), -(m1*(x1 - x2))/
((m1 + m2)*(x1^2 - 2*x1*x2 + x2^2 + y1^2 - 2*y1*y2 + y2^2 + z1^2 - 2*z1*z2 +
 z2^2))]
[
                                                                                ((x1 -
 x2)*(y1 - y2))/((m1 + m2)*(x1^2 - 2*x1*x2 + x2^2 + y1^2 - 2*y1*y2 + y2^2 +
 z1^2 - 2*z1*z2 + z2^2)),

                              (y1 - y2)^2/((m1 + m2)*(x1^2 - 2*x1*x2 + x2^2
 + y1^2 - 2*y1*y2 + y2^2 + z1^2 - 2*z1*z2 + z2^2)),

                                                       ((y1 - y2)*(z1 - z2))/((m1 +
 m2)*(x1^2 - 2*x1*x2 + x2^2 + y1^2 - 2*y1*y2 + y2^2 + z1^2 - 2*z1*z2 + z2^2)),

                                                                           -(m1*(x1 -
 x2)*(y1 - y2))/(m2*(m1 + m2)*(x1^2 - 2*x1*x2 + x2^2 + y1^2 - 2*y1*y2 + y2^2
 + z1^2 - 2*z1*z2 + z2^2)), (m1*x1^2 + m1*x2^2 + m2*x1^2 + m2*x2^2 + m2*y1^2
 + m2*y2^2 + m1*z1^2 + m1*z2^2 + m2*z1^2 + m2*z2^2 - 2*m1*x1*x2 - 2*m2*x1*x2
 - 2*m2*y1*y2 - 2*m1*z1*z2 - 2*m2*z1*z2)/(m2*(m1 + m2)*(x1^2 - 2*x1*x2 + x2^2
 + y1^2 - 2*y1*y2 + y2^2 + z1^2 - 2*z1*z2 + z2^2)),

                                                 -(m1*(y1 - y2)*(z1 - z2))/(m2*(m1 +
 m2)*(x1^2 - 2*x1*x2 + x2^2 + y1^2 - 2*y1*y2 + y2^2 + z1^2 - 2*z1*z2 + z2^2)),
```

```
 -(m1*(y1 - y2))/((m1 + m2)*(x1^2 - 2*x1*x2 + x2^2 + y1^2 - 2*y1*y2 + y2^2 +
 z1^2 - 2*z1*z2 + z2^2))]
[
                                                                        ((x1 -
 x2)*(z1 - z2))/((m1 + m2)*(x1^2 - 2*x1*x2 + x2^2 + y1^2 - 2*y1*y2 + y2^2 +
 z1^2 - 2*z1*z2 + z2^2)),

                    ((y1 - y2)*(z1 - z2))/((m1 + m2)*(x1^2 - 2*x1*x2 + x2^2
 + y1^2 - 2*y1*y2 + y2^2 + z1^2 - 2*z1*z2 + z2^2)),

                                                            (z1 - z2)^2/((m1 +
 m2)*(x1^2 - 2*x1*x2 + x2^2 + y1^2 - 2*y1*y2 + y2^2 + z1^2 - 2*z1*z2 + z2^2)),

                                                                   -(m1*(x1 -
 x2)*(z1 - z2))/(m2*(m1 + m2)*(x1^2 - 2*x1*x2 + x2^2 + y1^2 - 2*y1*y2 + y2^2
 + z1^2 - 2*z1*z2 + z2^2)),

                 -(m1*(y1 - y2)*(z1 - z2))/(m2*(m1 + m2)*(x1^2 - 2*x1*x2 +
 x2^2 + y1^2 - 2*y1*y2 + y2^2 + z1^2 - 2*z1*z2 + z2^2)), (m1*x1^2 + m1*x2^2 +
 m2*x1^2 + m2*x2^2 + m1*y1^2 + m1*y2^2 + m2*y1^2 + m2*y2^2 + m2*z1^2 + m2*z2^2
 - 2*m1*x1*x2 - 2*m2*x1*x2 - 2*m1*y1*y2 - 2*m2*y1*y2 - 2*m2*z1*z2)/(m2*(m1 +
 m2)*(x1^2 - 2*x1*x2 + x2^2 + y1^2 - 2*y1*y2 + y2^2 + z1^2 - 2*z1*z2 + z2^2)),
 -(m1*(z1 - z2))/((m1 + m2)*(x1^2 - 2*x1*x2 + x2^2 + y1^2 - 2*y1*y2 + y2^2 +
 z1^2 - 2*z1*z2 + z2^2))]
[

  (m2*(x1 - x2))/((m1 + m2)*(x1^2 - 2*x1*x2 + x2^2 + y1^2 - 2*y1*y2 + y2^2 +
 z1^2 - 2*z1*z2 + z2^2)),

                           (m2*(y1 - y2))/((m1 + m2)*(x1^2 - 2*x1*x2 + x2^2
 + y1^2 - 2*y1*y2 + y2^2 + z1^2 - 2*z1*z2 + z2^2)),

                                                         (m2*(z1 - z2))/((m1 +
 m2)*(x1^2 - 2*x1*x2 + x2^2 + y1^2 - 2*y1*y2 + y2^2 + z1^2 - 2*z1*z2 + z2^2)),


 -(m1*(x1 - x2))/((m1 + m2)*(x1^2 - 2*x1*x2 + x2^2 + y1^2 - 2*y1*y2 + y2^2 +
 z1^2 - 2*z1*z2 + z2^2)),

                         -(m1*(y1 - y2))/((m1 + m2)*(x1^2 - 2*x1*x2 + x2^2 +
 y1^2 - 2*y1*y2 + y2^2 + z1^2 - 2*z1*z2 + z2^2)),

                                                  -(m1*(z1 - z2))/((m1 + m2)*(x1^2
 - 2*x1*x2 + x2^2 + y1^2 - 2*y1*y2 + y2^2 + z1^2 - 2*z1*z2 + z2^2)),
 -(m1*m2)/((m1 + m2)*(x1^2 - 2*x1*x2 + x2^2 + y1^2 - 2*y1*y2 + y2^2 + z1^2 -
 2*z1*z2 + z2^2))]
```

Published with MATLAB® R2022b