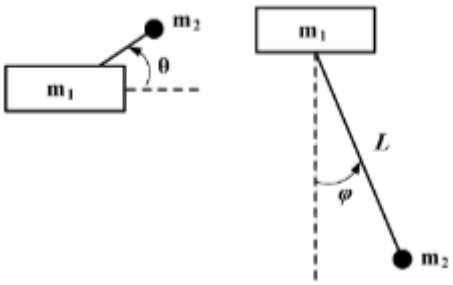
Hand in Assignment 1- Lagrange Modelling

Fikri Farhan Witjaksono *Oliver Wallin*

September 27, 2019

1. Introduction

Hand in 1 is about investigating and modelling a complex mechanical system based on the Lagrange approach. The system a helicopter with an assumed rigid rod connecting it to a hanging mass is the mechanical system to modell.



2. Question 1.a Model system with "minimal coordinates"

$$P_1 = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \quad P_2 = \begin{bmatrix} x \\ y \\ z \end{bmatrix} + L \begin{bmatrix} sin\varphi + cos\theta \\ sin\varphi + sin\theta \\ -cos\varphi \end{bmatrix}$$

$$q = \begin{bmatrix} x \\ y \\ z \\ \theta \\ \varphi \end{bmatrix}, \quad \dot{q} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\theta} \\ \dot{\varphi} \end{bmatrix}, \quad \dot{P}_1 = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix}$$

P1 is the position of Mass 1 (the helicopter) and is defined as the center of the system. Then the position of the hanging mass is calculated with spherical coordinates with regards to the angles and the length of the rigid rod. "q" is the generalized coordinates given in the question and "qdot" is the derivative of "q".

$$\dot{p_2} = \frac{\partial p_2}{\partial q} \cdot \dot{q}$$

To calculate the derivative of P2 which is expressed as $\dot{p}_2$, matlab function "Jacobian" is used to calculate $\frac{\partial p2}{\partial q}$,the result is then multiplied with $\dot{q}$ as the internal derivative.

$$T_1 = \frac{m_1 \cdot (\dot{p}_1^T \cdot \dot{p}_1)}{2}, \quad T_2 = \frac{m_2 \cdot \dot{p}_2(q,\dot{q})^T \cdot \dot{p}_2(q,\dot{q}))}{2}$$

$$T = T_1 + T_2;$$

Then the kinetic energy is calculated with the given function from the assignment. T_1 is an expression of the kinetic energy for the helicopter and T_2 is the expression of the kinetic energy for the hanging mass. The kinetic energies are then summed up to calculate it as a system

$$P_1 = m_1 g \cdot [0\ 0\ 1] \cdot p_1$$

$$P_2 = m_2 g \cdot [0\ 0\ 1] \cdot p_2$$

$$P = P1 + P2$$

The potential energy is added in the same way as the kinetic energy, P1 is for the helicopter and P2 is for the hanging mass.

$$\mathcal{L} = T(q, \dot{q}) - P(q)$$

The Lagrange summation is then ready to be calculated above.

$$U = \begin{bmatrix} F_x \\ F_y \\ F_z \\ 0 \\ 0 \end{bmatrix}$$

Let us introduce variable U which is the defined external forces acting on the system, in this case the forces are applied on the helicopter.

Next, we derive the Euler-Lagrangian by the equation below.

$$\nabla_q \mathcal{L} = \nabla_q T - \nabla_q V$$

$$\mathrm{U} = \frac{d}{dt}\nabla_{\dot{q}}\mathcal{L} - \nabla_q \mathcal{L} = \frac{\partial}{\partial q}(W(q)\dot{q})\dot{q} + \frac{\partial}{\partial \dot{q}}(W(q)\dot{q})\ddot{q} - \nabla_q \mathcal{L}$$

The final form of the Euler-Lagrange equation is shown by the equation below.

$$\dot{q} = v$$

$$M(q)\dot{v} = b(q, \dot{q}, u)$$

The final form consists of the terms from the initial Euler-Lagrangian equation. Rewriting b(q,q̇,u) and M(q) gives us the equation below.

$$b(q, \dot{q}, u) = \mathrm{U} - \frac{\partial}{\partial q}(W(q)\dot{q})\dot{q} + \nabla_q \mathcal{L}$$

$$M(q)\dot{v} = \frac{\partial}{\partial \dot{q}}(W(q)\dot{q})\ddot{q}$$

where W(q) is represented by the equation shown below. W(q) is considered as the equal form of M(q) in the final form

$W(q)$ = Please refer to the Matlab Code in the Appendix

Hence, after calculation in Matlab, the final form $b(q, \dot{q}, u)$ equation is obtained as shown below

$b(q, \dot{q}, u)$ = Please refer to the Matlab code in the Appendix

3. Question 1.b Model system with the Lagrange approach

In this part, we will introduce the constrained Lagrange approach. By this approach, we use the scalar constraint 'C'. Moreover, the generalized coordinates (q) will be transformed to

$$P_{1,const} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \quad \mathrm{p}_{2,const} = \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} \quad , \quad q = \begin{bmatrix} x \\ y \\ z \\ x_2 \\ y_2 \\ z_2 \end{bmatrix}, \qquad \dot{P}_{1,const} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{y} \end{bmatrix}$$

$$C = \frac{e^T e - L^2}{2} \quad , \quad e = p1 - p2$$

U is the defined external forces acting on the system, in the case of constrained system, it is 6x1 matrix compared to previous approach in 1(a) due to additional constraint as shown below.

$$U = \begin{bmatrix} F_x \\ F_y \\ F_z \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

The Constrained Lagrange equation is changed due to additional term C*z. The result is shown below.

$$\mathcal{L} = \mathrm{T}(q, \dot{q}) - P(q) - C \cdot \mathrm{z}$$

Next, the derivation of Euler-Lagrange equation for Constrained case will result in the equation below with additional term of gradient of Cz as well as additional state C(q).

$$\nabla_q \mathcal{L} = \nabla_q T - \nabla_q V - \nabla_q C \cdot \mathrm{z}$$

$$\mathrm{U} = \frac{d}{dt}\nabla_{\dot{q}}\mathcal{L} - \nabla_q \mathcal{L} = \frac{\partial}{\partial q}(W(q)\dot{q})\dot{q} + \frac{\partial}{\partial \dot{q}}(W(q)\dot{q})\ddot{q} - \nabla_q \mathcal{L} - \nabla_q C \cdot \mathrm{z}$$

$$C(q) = 0$$

where W(q) is represented by the equation shown below.

$$W(q) = \begin{bmatrix} m_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & m_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & m_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & m_1 + m_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & m_1 + m_2 & 0 \\ 0 & 0 & 0 & 0 & 0 & m_1 + m_2 \end{bmatrix}$$

Hence, after calculation in Matlab, the final general form equation is obtained as shown below.

$$\dot{q} = v$$

$$M(q)\dot{v} = b(q, z, u)$$

$$C(q) = 0$$

where M(q) and b are the general solution of the system. Here, $M(q)$ is considered to be the same with the state solution W(q). Due to additional term in the constraint, b are expressed as function of (q, z, u) instead of $(q, \dot{q}, u)$. $M(q)$ and $b(q, z, u)$ are expressed below.

$$b(q, z, u) = \begin{bmatrix} F_x - z(x - x_2) \\ F_y - z(y - y_2) \\ F_z - gm_1 - z(z - z_2) \\ z(x - x_2) \\ z(y - y_2) \\ z(z - z_2) - gm_2 \end{bmatrix}$$

$$M(q) = \begin{bmatrix} m_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & m_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & m_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & m_1 + m_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & m_1 + m_2 & 0 \\ 0 & 0 & 0 & 0 & 0 & m_1 + m_2 \end{bmatrix}$$

When the model made by "minimal coordinates" and the one made with the "Lagrange approach" is compared there are clearly some differences. In the model for "minimal coordinates" there are plenty of trigonometric functions included in the W-matrix. Trigonometric functions that isn't to be found in the Lagrange matrix. The "minimal coordinates" approach leads to a more complex answer where the "Lagrange" approach makes it a more pleasant algebraic equation.

3. Question 2.a

To use the previous model in the form that is stated we need to differentiate our constraint

$$\begin{bmatrix} M & a(q) \\ a(q)^T & 0 \end{bmatrix} \begin{bmatrix} \ddot{q} \\ z \end{bmatrix} = c(q, \dot{q}, u),$$

$$\dot{C}(q, \dot{q}, \ddot{q}) = \frac{\partial C(q)}{\partial q} \dot{q},$$

$$\ddot{C}(q, \dot{q}, \ddot{q}) = \frac{\partial C(q)}{\partial q} \ddot{q} + \frac{\partial}{\partial q} \left(\frac{\partial C(q)}{\partial q} \dot{q} \right) \dot{q},$$

$$a(q) = \frac{\partial C(q)}{\partial q} \; and \; c(q, \dot{q}, u) = \begin{bmatrix} b + \nabla_q C(q) z \\ -\frac{\partial}{\partial q} \left(\frac{\partial C(q)}{\partial q} \dot{q} \right) \dot{q} \end{bmatrix}$$

After the calculation using the simulation, the result of the states $a(q)$, $c(q, \dot{q}, u)$, M which shows the general solution for the implicit method is shown below in the next page.

$$a(q) = \begin{bmatrix} x - x_2 \\ y - y_2 \\ z - z_2 \\ x_2 - x \\ y_2 - y \\ z_2 - z \end{bmatrix}$$

$$c(q, \dot{q}, u) = \begin{bmatrix} F_x - z(x - x_2) \\ F_y - z(y - y_2) \\ F_z - m_1 g - z(z - z_2) \\ z(x - x_2) \\ z(y - x_2) \\ z(z - z_2) - g m_2 \\ \left(-\dot{x}^2 - \dot{x}_2^2 + 2\dot{x}\dot{x}_2\right) + \left(-\dot{y}^2 - \dot{y}_2^2 + 2\dot{y}\dot{y}_2\right) + \left(-\dot{z}_2{}^2 + 2\dot{z}\dot{z}_2 - \dot{z}^2\right) \end{bmatrix}$$

$$M(q) = \begin{bmatrix} m_1 & 0 & 0 & 0 & 0 & 0 & x - x_2 \\ 0 & m_1 & 0 & 0 & 0 & 0 & y - y_2 \\ 0 & 0 & m_1 & 0 & 0 & 0 & z - z_2 \\ 0 & 0 & 0 & m_1 + m_2 & 0 & 0 & x_2 - x \\ 0 & 0 & 0 & 0 & m_1 + m_2 & 0 & y_2 - y \\ 0 & 0 & 0 & 0 & 0 & m_1 + m_2 & z_2 - z \\ x - x_2 & y - y_2 & z - z_2 & x_2 - x & y_2 - y & z_2 - z & 0 \end{bmatrix}$$

4. Question 2.b

The explicit form of the matrixes can be written as:

$$\begin{bmatrix} \ddot{q} \\ z \end{bmatrix} = \begin{bmatrix} M(q) & a(q) \\ a(q)^T & 0 \end{bmatrix} c(q, \dot{q}, u),$$

Which can be done by multiply the inverse of the M matrix with the c constant.

$$Explicit = M^{-1} \cdot c$$

We see that this explicit form is better used in the simulation of models rather than the implicit method due to its less complexity in computation. For numerical simulation in explicit method e.g in Finite Element Car crash simulation, we could use small time step to iterate solution to highly non-linear problems. On the other hand, the benefits of implicit method is due to its stability during simulation when the small time step is used, hence allow for larger time step size.

Appendix (MATLAB Full code)

```
% Matlab Code for Assignment 1

clear all
close all
clc

syms x y z Theta Phi L xDot yDot zDot ThetaDot PhiDot m1 m2 g Fx
Fy Fz e z x2 y2 z2 x2Dot y2Dot z2Dot

% Positions and Generalized coordinates
p1 = [x;y;z]; %Define position of Mass 1
p1Dot = [xDot;yDot;zDot];
p2 = [(x+L*sin(Phi)+L*cos(Theta));(y+L*sin(Phi)+L*sin(Theta));(z-
L*cos(Phi))]; % Define position of Mass 2
q = [x;y;z;Theta;Phi]; % Define Generalized Coordinates
qDot = [xDot;yDot;zDot;ThetaDot;PhiDot]; % Define derivative of
Generalized coordinates

% Jacobian Calculation for p2Dot
p2Dot =jacobian(p2,q)*qDot;

% Kinetic Energy summation
T1 = (1/2)*(m1)*(p1Dot.')*(p1Dot);
T2 = (1/2)*m2*(p2Dot.')*p2Dot;
T = T1+T2;

%Potential Energy
P1 = m1*g*[0 0 1]*p1;
P2 = m2*g*[0 0 1]*p2;
P = P1+P2;

% Lagrange Summation
Lagrange = T-P;

%Define External Force
U = [Fx;Fy;Fz;0;0];

% Euler-Lagrange Equation

Tdot = jacobian(T,qDot);
W = jacobian(Tdot,qDot);

W_qdot = W*qDot;

diff_W_qdot = jacobian(W_qdot,q);

diff_W_qdot_qdot = diff_W_qdot*qDot;

diff_T = jacobian(T,q).'; % Gradient of Kinetic Energy
diff_P = jacobian(P,q).'; % Gradient of Kinetic Energy

b = U - diff_W_qdot_qdot + diff_T - diff_P; %b Matrix of the
final form

% Constrained Position, Generalized Coordinates and and Scalar
Constraints
p1Dot_const = [xDot;yDot;zDot];
p2_const = [x2;y2;z2]; % Constrained p2 position
q_const = [p1;p2_const]; % Constrained Generalized coordinates
qDot_const = [xDot;yDot;zDot;x2Dot;y2Dot;z2Dot];
e = p1-p2_const;
C = 0.5*((e.')*e-(L^2));
```

```
% Jacobian Calculation for constrained p2Dot

p2Dot_const = jacobian(p2_const,q_const)*qDot_const;

% Constrained Kinetic Energy,Potential Energy and Lagrange

T1_const = (1/2)*(m1)*(p1Dot_const.')*(p1Dot_const);
T2_const = (1/2)*m2*(p2Dot_const.')*p2Dot_const;
T_const = T1_const+T2_const; % Constrained Kinetic Energy

P1_const = m1*g*[0 0 1]*p1;
P2_const = m2*g*[0 0 1]*p2_const;
P_const = P1_const+P2_const; % Constrained Potential Energy

Lagrange_const = T_const-P_const-C*z; % Constrained Lagrange Equation

U_const = [Fx Fy Fz 0 0 0]; % External Force to the system

% Constrained Euler-Lagrange
Tdot_const = jacobian(T_const,qDot_const);
W_const = jacobian(Tdot_const,qDot_const);% W(q) matrix

W_qdot_const = W_const*qDot_const;

diff_W_qdot_const =  jacobian(W_qdot_const,q_const);

diff_W_qdot_qdot_const = diff_W_qdot_const*qDot_const;

diff_W_qdot_qdot_const_final = diff_W_qdot_qdot_const.'; % Transposed in order
to get Column Matrix

diff_T_const = jacobian(T_const,q_const); %Gradient q of T
diff_P_const = jacobian(P_const,q_const); %Gradient q of P

grad_q_C = jacobian(C,q_const);
Cz = grad_q_C*z; %Gradient q of Cz

b_const = U_const-diff_W_qdot_qdot_const_final+diff_T_const-diff_P_const-Cz;
b_const_final = b_const.'; %Transpose b_const so that we get the column matrix

% Implicit Model Matrix

%Left Matrix M(q)
a = grad_q_C;
a_final = a.'; %transpose a so that it is a column matrix
conA = [W_const a_final];
conB = [a_final.' 0];

M = [conA;conB];

%Right Matrix c(q,qdot,u)
diff_q_C = jacobian(C,q_const);
diff_q_C_2 = diff_q_C*qDot_const;
diff_q_C_3 =jacobian(diff_q_C_2,q_const);
diff_q_C_4 = diff_q_C_3*(-1)*qDot_const;
diff_q_C_4_final = diff_q_C_4.’;

c = [b_const_final;diff_q_C_4]; % First row is constrained Euler-Lagrange
Equation

%Explicit Model
Explicit = inv(M)*c;
```