10

I used [1] in making this assignment when finished I compared initial values with Prajish Kumar (4743873).	**Rick Staa, #4511328** **HW set 3 due 13/03/2018** **ME41060**

1 Statement of entregity

my homework is completely in accordance with the Academic Integrity

2 Problem Statement

In this assignment, we were asked to add passive elements, active elements and an impulsive contact to the model we created in assignment 1 and 2. The mass, length and inertia parameters are still those depicted in assignment 1. Further the initial state is still $x_0 = [\phi_1\ \phi_2\ \dot{\phi}_1\ \dot{\phi}_2]$. Only the state changes depending on the type of constraints.

2.1 Addition of a passive element

We were asked to add a spring (passive element) to the model that was attached to the ground in D, with coordinates (−L/2, 0) and connected to bar 1 in point E, being at 2/3 of the length measured from point A (Figure 1: Double pendulum with spring). The free length of this spring was said to be $l_o = \frac{2L}{3}$ and its linear stiffness k = (15/2)(mg/l). We were following asked to calculate the accelerations of the center of mass of the two bodies together with the LaGrangian multipliers for the case when the two bars are at $x_0 = [0.5\pi\ 0.5\pi\ 0\ 0]$.

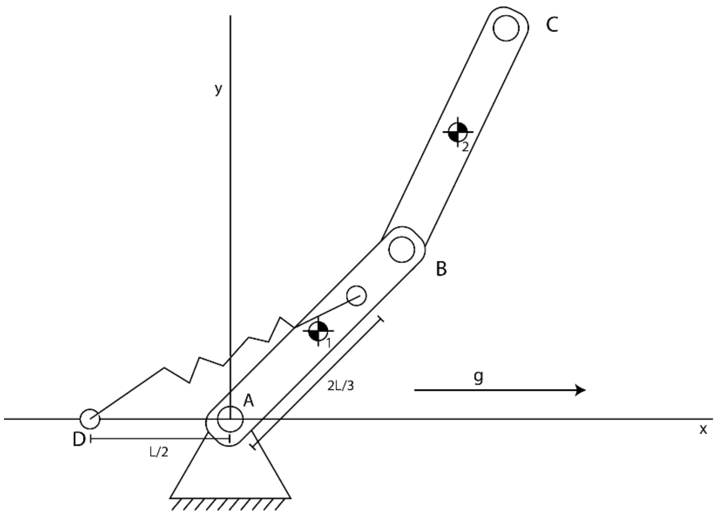

Figure 1: Double pendulum with spring

2.2 Adding the passive element

The addition of a passive element is very straight forward since most of the ingredients have already been calculated in the previous assignment. The passive (and active) elements can be added to the virtual work equations in a similar way as we would add the regular constrains. The virtual power for a passive element is:

$$\delta P = \sigma_l C_{l,i} \delta \dot{x}_i \tag{1}$$

Together with the regular constraints derived in assignment 2 the full virtual power equation now becomes:

$$\delta P = \delta \dot{x}_i \left(f_i - M_{ij} \ddot{x}_j - \lambda_k C_{k,i} - \sigma_v C_{v,i} \right) \tag{2}$$

If we rewrite this virtual power equation in matrix vector form we get the following system:

$$\begin{bmatrix} M_{\boldsymbol{i},\boldsymbol{j}} & C_{k,i}^{\boldsymbol{T}} \\ C_{k,i} & 0_{kk} \end{bmatrix} \begin{bmatrix} \ddot{x}_i \\ \lambda_k \end{bmatrix} = \begin{pmatrix} F_{\boldsymbol{i}} - C_{v,j}\sigma_v \\ -D_{k,ij}\dot{x}_i\dot{x}_j \end{pmatrix} \tag{3}$$

Except σ_v and $C_{v,j}$ all the terms have already been calculated in assignment 2 and 1. From these two σ_v depicts the constitutive behaviour or the linear elastic spring force and $C_{v,j}$ the transformations from the element forces to the forces at the cm coordinates. As a result, $C_{v,j}$ can be obtained by taking the elongation of the spring as an element constraint. This constraint needs to be expressed in the coordinates of the cm's of the body and differentiated to get it in the right form. The spring constraint expressed can be derived from the elongation of the spring:

$$C_v = l_v - l_0 = \sqrt{(x_E - x_D)^2 + (y_E - y_D)^2} - l_0 \tag{4}$$

In this l_v is the spring length and l_o is the rest length which was said to be $\frac{2}{3}L$. The spring constrained expressed in the Cm's of the body then becomes:

$$C_v = \sqrt{\left(x_1 + \frac{l}{6}\cos(\phi_1) + \frac{l}{2}\right)^2 + \left(y_1 + \frac{l}{6}\sin(\phi_1) - 0\right)^2} - \frac{2}{3}L$$

By differentiating C_v w.r.t. the state variables $\phi_1, \phi_2, \dot{\phi}_1 and\ \dot{\phi}_1$ we get the following constraint equations.

$$C_{v,i}(\boldsymbol{x}) = \frac{1}{l_s} \begin{bmatrix} x_1 + \frac{1}{6} l cos(\phi_1) + \frac{l}{2} \\ y_1 + \frac{1}{6} l sin(\phi_1) \\ \frac{1}{6} l \left(y_1 \cos(\phi_1) - x_1 \sin(\phi_1) - \frac{l}{2}\sin(\phi_1) \right) \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{5}$$

The constitutive relationship is as follows:

$$\sigma_v = F_v = k\Delta \mathrm{L} = k(L_v - L_0) = k\left(L_v - \frac{2}{3}L\right) \tag{6}$$

We now have all the ingredients to calculate the accelerations for the case when we have a passive element. The MATLAB code implementing this calculation can be found in Appendix A – Passive element

2.3 Calculate accelerations

With $x_0 = [0.5\,\pi\ 0.5\pi\ 0\ 0]$ we get the following result:

$\ddot{x}_1$	$2.1021\ \frac{m}{s^2}$
$\ddot{y}_1$	$0.00\ \frac{m}{s^2}$
$\ddot{\phi}_1$	$-7.6442\ \frac{rad}{s^2}$
$\ddot{x}_2$	$8.4086\,\frac{m}{s^2}$
$\ddot{y}_2$	$0.00\frac{m}{s^2}$
$\ddot{\phi}_2$	$-15.2883\ \frac{rad}{s^2}$
λ_1	0.2274 N
λ_2	-1.2733 N
λ_3	0.1819 N
λ_4	0.00 N

Since this situation is equal to the situation in Assignment 2 and we know the relationship between LaGrange multipliers we can compare the results. The biggest difference apart from scaling effects is element 1 now also has a vertical force component added to it. This is expected as we look at the attachment place of the spring.

3 Adding active element

We are now asked to add a motor to the pendulum this setup is shown in Figure 2: Double pendulum with motor.

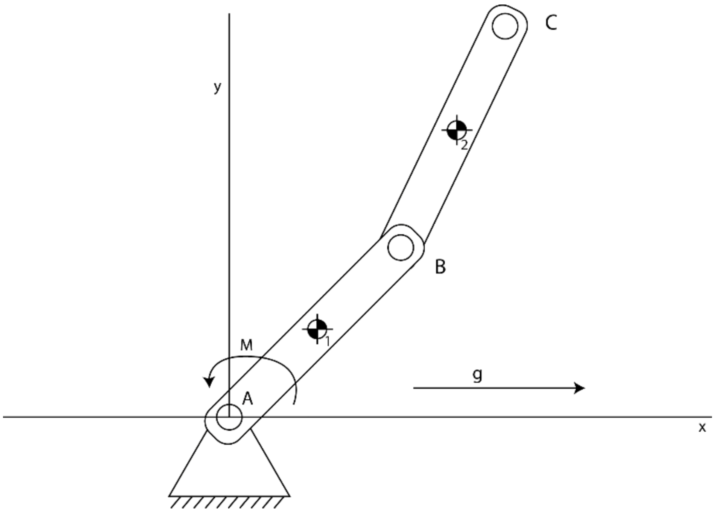

Figure 2: Double pendulum with motor

The active element (motor force) can be added to the already in assignment 1 and 2 derived model equations in a similar way as this was done for the passive element. To do this we now add the following additional constraint to the constraint vector C_k:

$$C_l(\boldsymbol{x}, t) = \phi_1 - \omega t = 0 \tag{7}$$

In this ω depicts the given constant speed of the motor and t the time. The new constraint now becomes a kinetic constrained and thus we need a full (convective) derivative with respect to both the state and time.

$$C_l(\boldsymbol{x}) = \begin{bmatrix} x_1 - \frac{1}{2}Lcos(\phi_1) \\ y_1 - \frac{1}{2}Lsin(\phi_1) \\ x_2 - x_1 - \frac{1}{2}Lcos(\phi_1) - \frac{1}{2}Lcos(\phi_2) \\ y_2 - y_1 - \frac{1}{2}Lsin(\phi_1) - \frac{1}{2}Lsin(\phi_2) \\ \phi_1 - \omega t \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{8}$$

This results in the following model equations:

$$\begin{bmatrix} M & C_{l,x}^T \\ C_{l,x} & 0 \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{x}} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} F \\ -C_{li,j}\dot{x}_i\dot{x}_j - C_{l,it}\dot{x}_i - C_{l,tt} \end{bmatrix} \tag{9}$$

4

The $C_{l,it}$ and $C_{l,tt}$ were derived by MATLAB symbolic toolbox and will since they are not done by hand not be depicted here. As a result of the motor force we now get a fifth Laplace multiplier. The MATLAB code for doing this can be found in Appendix B.

3.1 Calculate accelerations

With $x_0 = \left[0.5\,\pi\ 0.5\pi\ -120\left(\frac{2\pi}{60}\right) 120\left(\frac{2\pi}{60}\right)\right]$ we get the following result:

1

$\ddot{x}_1$	0.00 $\frac{m}{s^2}$
$\ddot{y}_1$	-43.4263 $\frac{m}{s^2}$
$\ddot{\phi}_1$	0.00 $\frac{rad}{s^2}$
$\ddot{x}_2$	7.36 $\frac{m}{s^2}$
$\ddot{y}_2$	-130.28 $\frac{m}{s^2}$
$\ddot{\phi}_2$	-26.75 $\frac{rad}{s^2}$
λ_1	1.5917 N
λ_2	22.5469 N
λ_3	0.3183 N
λ_4	16.9102 N
λ_5	-0.5253 N

Since λ_5 = -0.52 Nm in clockwise direction the motor power supplied to the mechanical system is equal to:

$$P = Torque \cdot \omega = -0.52 \cdot -4\pi = 6.61\ \boldsymbol{W} \tag{10}$$

9

4 Adding an impulsive contact

In this part of the question we were asked to calculate the impulsive forces for the case when the bars hit the wall when they are vertically up. This was said to occur with an angular speed of $\omega = 120\,RPM = 4\pi\frac{rad}{s}$ in the counter clockwise direction. The constraint equations for the impact position are as follows:

$$x_B = x_1 + \frac{1}{2}Lcos(\phi_1) \tag{11}$$

$$x_C = x_2 + \frac{1}{2}Lcos(\phi_2) \tag{12}$$

By differentiating these constraints and filling them in in virtual power equation we get the following result:

$$M\ddot{x} - \mathrm{F} - C_{k,i}\lambda_k = 0 \tag{13}$$

Next to get the impulsive forces we need to look at the contacts during a very short impact time. We do this by integrating over the time and then taking the limit from $t^- \to t^+$.

$$\lim_{t^- \to t^+}\left(\int_{t^-}^{t^+} F_i dt - \int_{t^-}^{t^+} M_{ij}\ddot{x}_j\, dt = \ C_{k,i}\int_{t^-}^{t^+} \lambda_k\ dt\right) \tag{14}$$

Since the $\int_{t^-}^{t^+} F_i dt$ term is equal to s_i and the $\int_{t^-}^{t^+} \lambda_k\ dt$ to ρ_k this results in the following:

$$M_{ij}\dot{x}_j^{\ +} + C_{k,i}\rho_k = s_i + M_{ij}\dot{x}_j^{\ -} \tag{15}$$

After some reordering we get the following matrix vector product:

$$\begin{pmatrix} M_{ij} & C_{l,i} \\ C_{l,j} & 0 \end{pmatrix}\begin{bmatrix} x_j^+ \\ \rho_k \end{bmatrix} = \begin{pmatrix} s_i + M_{ij}\dot{x}_j^{\ -} \\ -e \cdot C_{l,j}\dot{x}_j^{\ -} \end{pmatrix} \tag{16}$$

In which the e term represents the coefficient of restitution and is given as:

$$\frac{\dot{x}_{rel}^+}{\dot{x}_{rel}^-} = -e \tag{17}$$

4.1 Calculate and post impact velocities, contact impulses, kinetic energy and work.

These calculations need to be done at the earlier names initial state $x_0 = [0.5\pi\ 0.5\pi\ 4\pi\frac{rad}{s}\ 4\pi\frac{rad}{s}]$. The velocities, contact impulses, kinetic energy and work are calculate in the MATLAB code in Appendix C – Impulsive impact forceThe kinetic energy before impact, after impact and the work done by the contact impulses were calculated as successively:

$$K^- = \frac{1}{2}\dot{x}_i^- M_{ij}\dot{x}_j^- \tag{18}$$

$$K^+ = \frac{1}{2}\dot{x}_i^+ M_{ij}\dot{x}_j^+ \tag{19}$$

$$W_\rho = \frac{1}{2}(1-e)\rho_k\dot{x}_k^- \tag{20}$$

4.1.1 e = 1

$\ddot{x}_1$	3.46 $\frac{m}{s^2}$
$\ddot{y}_1$	0.00 $\frac{m}{s^2}$
$\ddot{\phi}_1$	$-12.57\frac{rad}{s^2}$
$\ddot{x}_2$	10.37 $\frac{m}{s^2}$
$\ddot{y}_2$	0.00 $\frac{m}{s^2}$
$\ddot{\phi}_2$	-12.57 $\frac{rad}{s^2}$
λ_1	-0.30 N
λ_2	0.00N
λ_3	-1.20 N
λ_4	0.00 N
λ_5	-1.80 N
λ_6	-1.50 N
K^-	8.27 J
K^+	8.27 J
W_ρ	0.00 J

units

4.1.2 e = 0.5

$\ddot{x}_1$	1.73 $\frac{m}{s^2}$
$\ddot{y}_1$	0.00 $\frac{m}{s^2}$
$\ddot{\phi}_1$	$-6.28\frac{rad}{s^2}$
$\ddot{x}_2$	5.19 $\frac{m}{s^2}$
$\ddot{y}_2$	0.00 $\frac{m}{s^2}$

$\ddot{\phi}_2$	-6.28 $\frac{rad}{s^2}$
λ_1	-0.22 N
λ_2	0.00N
λ_3	-0.90 N
λ_4	0.00 N
λ_5	-1.35 N
λ_6	-1.12 N
K^-	8.27 J
K^+	2.07 J
W_ρ	-6.15 J

4.1.3 e = 0

1

$\ddot{x}_1$	0.00 $\frac{m}{s^2}$
$\ddot{y}_1$	0.00 $\frac{m}{s^2}$
$\ddot{\phi}_1$	0.00 $\frac{rad}{s^2}$
$\ddot{x}_2$	0.00 $\frac{m}{s^2}$
$\ddot{y}_2$	0.00 $\frac{m}{s^2}$
$\ddot{\phi}_2$	0.00 $\frac{rad}{s^2}$
λ_1	-0.15 N
λ_2	0.00N
λ_3	-0.60 N
λ_4	0.00 N
λ_5	-0.90 N
λ_6	-0.75 N
K^-	8.27 J
K^+	0.00 J
W_ρ	-8.20 J

5 Unknown Velocity jumps instead of unknown velocities after impact

When we use the velocity jumps $\Delta\dot{x} = \dot{x}_i{}^+ - \dot{x}_i{}^-$ as the unknowns the following things change in the impulsive equations:

1

$$M_{ij}x_j^+ + C_{k,i}\rho_i = M_{ij}x_j^- + s_i \tag{21}$$

$$M_{ij}\left(x_j^+ - x_j^-\right) + C_{k,i}\rho_i = s_i \tag{22}$$

$$M_{ij}\Delta x_j + C_{k,i}\rho_i = s_i \tag{23}$$

$$C_{k,j}x_j{}^+ = -eC_{k,j}x_j^- \tag{24}$$

$$C_{k,j}(x_j{}^+ - x_j{}^-) = -eC_{k,j}x_j^- - C_{k,j}x_j{}^- \tag{25}$$

$$C\_(k,j)\Delta x_j = -(1+e)C_{k,j}x_j^- \quad (26)$$

These equations will result in the following model equations:

$$\begin{bmatrix} M & C_{k,x}^T \\ C_{k,x} & 0 \end{bmatrix} \begin{bmatrix} \Delta x_j \\ e_k \end{bmatrix} = \begin{bmatrix} s_i \\ -(1+e)C_{k,ij}x_j^- \end{bmatrix} \quad (27)$$

Advantage)
Mass matrix now only on one side of the Vector Matrix Product.

Disadvantage:
You still need to compute the velocities after impact since you now are only looking at the difference.

6 References

[1] A. L. Schwab, “Virtual power and Lagrance multipliers,” in *Multibody Dynamics*, Delft, The Netherlands: TU Delft, 2018.

Appendix A – Passive element

```
%% MBD_B: Assignment 3
% Question 1 - Passive Element

%  Rick Staa (4511328)
%  Last edit: 05/03/2018
clear all; close all; % clc;

%% Parameters
% Segment 1
parms.L      = 0.55;                                                    % [m]
parms.w      = 0.05;                                                    % [m]
parms.t      = 0.004;                                                   % [m]
parms.p      = 1180;                                                    % [kg/m^3]
parms.m      = parms.p * parms.w * parms.t * parms.L;                   % [kg]
parms.I      = (1/12) * parms.m * parms.L^2;                            % [kg*m^2]

% World parameters
parms.g       = 9.81;                       % [m/s^2]

% Spring parameters
parms.k                      = (15/2)*parms.m*parms.g/parms.L;     % stiffness of spring

%% Compute constraint matrices

% Use symbolic toolbox to calculate derivatives
syms x1 y1 phi1 x2 y2 phi2
syms dx1 dy1 dphi1 dx2 dy2 dphi2
x = [x1 y1 phi1 x2 y2 phi2];
dx = [dx1 dy1 dphi1 dx2 dy2 dphi2];
parms.x = x;
parms.dx = dx;

% The normal constrain equations
C = [x1-(parms.L/2)*cos(phi1); y1-(parms.L/2)*sin(phi1) ;   ...
    (x2-(parms.L/2)*cos(phi2))-(x1+(parms.L/2)*cos(phi1));            ...
    (y2-(parms.L/2)*sin(phi2))-(y1+(parms.L/2)*sin(phi1))];

% Calculate the jacobian to create the constraint equations
Cx = jacobian(C,x);
Cx = simplify(Cx);

% Take the second derivative to create the gluing constraints
Cd = Cx*dx';
Cdp = jacobian(Cd,x)*dx';
Cdp = simplify(Cdp);

% Now also calculate the spring constraint terms
Cs = sqrt((x1 + (parms.L/6)*cos(phi1) + parms.L/2)^2 + (y1 + (parms.L/6)*sin(phi1))^2) -
2*(parms.L/3);
Csx = jacobian(Cs,x); % Jacobian of Cs
Csx = simplify(Csx)';

% Put Csx Cdp Cd cx in parms struct and feed tem into the solver
syst.Cx      = Cx;
syst.Cd      = Cd;
syst.Cdp     = Cdp;
syst.Cs      = Cs;
syst.Csx     = Csx;

%% A:
x0  = [0.5*pi 0.5*pi 0 0];
[X] = state_calc(x0,parms,syst);
X   = double(vpa(X));

%% A: Create state space matrices for the case when we add a spring
function [X] = state_calc(x0,parms,syst)

% Get system matrices out
structname_fields = fields(parms);
```

```
for i = 1:size(fields(parms))
    eval_str = [structname_fields{i,:},'=','parms.',structname_fields{i,:},';'];
    eval(eval_str);
end
structname_fields = fields(syst);
for i = 1:size(fields(syst))
    eval_str = [structname_fields{i,:},'=','syst.',structname_fields{i,:},';'];
    eval(eval_str);
end

% Calculate missing initial states
x1 = 0.5*parms.L*cos(x0(1));
y1 = 0.5*parms.L*sin(x0(2));
x1 = 0;
y1 = parms.L/2;

Cx = subs(Cx, {'phi1','phi2','dphi1','dphi2'},...
    [x0(1),x0(2),x0(3),x0(4)]);

Cdp = subs(Cdp, {'phi1','phi2','dphi1','dphi2'},...
    [x0(1),x0(2),x0(3),x0(4)]);

Cs = subs(Cs, {'phi1','phi2','x1','y1'},...
    [x0(1),x0(2),x1,y1]);

Csx = subs(Csx, {'phi1','phi2','x1','y1'},...
    [x0(1),x0(2),x1,y1]);

sigma = parms.k*Cs; % Spring Force

% Create matrices
M = diag([parms.m,parms.m,parms.I,parms.m,parms.m,parms.I]);
A = [M Cx';Cx zeros(4,4)];
F = [parms.m*parms.g 0 0 parms.m*parms.g 0 0]' - Csx*sigma; % Updated F vector
b = [F;-Cdp];
X = A\b;

end
```

Appendix B – Active element

```
%% MBD_B: Assignment 3
% Question 1 - Active Element

%  Rick Staa (4511328)
%  Last edit: 05/03/2018
clear all; close all; % clc;

%% Parameters
% Segment 1
parms.L      = 0.55;                                                % [m]
parms.w      = 0.05;                                                % [m]
parms.t      = 0.004;                                               % [m]
parms.p      = 1180;                                                % [kg/m^3]
parms.m      = parms.p * parms.w * parms.t * parms.L;               % [kg]
parms.I      = (1/12) * parms.m * parms.L^2;                        % [kg*m^2]

% World parameters
parms.g       = 9.81;                       % [m/s^2]

% Spring parameters
parms.k                     = (15/2)*parms.m*parms.g/parms.L;      % stiffness of spring

% Motor constraint
omega = -120*(2*pi/60); % motor speed

%% Compute constraint matrices

% Use symbolic toolbox to calculate derivatives
syms x1 y1 phi1 x2 y2 phi2 t
```

```matlab
syms dx1 dy1 dphi1 dx2 dy2 dphi2
x = [x1 y1 phi1 x2 y2 phi2];
dx = [dx1 dy1 dphi1 dx2 dy2 dphi2];
parms.x = x;
parms.dx = dx;

% The normal constrain equations
C = [x1-(parms.L/2)*cos(phi1); y1-(parms.L/2)*sin(phi1) ;    ...
    (x2-(parms.L/2)*cos(phi2))-(x1+(parms.L/2)*cos(phi1));  ...
    (y2-(parms.L/2)*sin(phi2))-(y1+(parms.L/2)*sin(phi1));  ...
    (phi1 - omega*parms.t)];                    % Add extra motor constraint

% Calculate the jacobian to create the constraint equations
Cx = jacobian(C,x);
Cx = simplify(Cx);

% Constraint derivative with respect to time
Ct = simplify(jacobian(C,t));
Ctt = simplify(jacobian(Ct,t)); % double derivative

% Take the second derivative to create the gluing constraints
Cd = Cx*dx';
Cdp = jacobian(Cd,x)*dx';
Cdp = simplify(Cdp);
Cdt = simplify(jacobian(Cd,t));

% Put Csx Cdp Cd cx in parms struct and feed tem into the solver
syst.Cx     = Cx;
syst.Cd     = Cd;
syst.Cdp    = Cdp;
syst.Ct     = Ct;
syst.Ctt    = Ctt;
syst.Cdt    = Cdt;

%% A:
x0  = [0.5*pi 0.5*pi omega omega];
[X] = state_calc(x0,parms,syst);
X   = double(vpa(X));

%% A: Create state space matrices for the case when we add a spring
function [X] = state_calc(x0,parms,syst)

% Get system matrices out
structname_fields = fields(parms);
for i = 1:size(fields(parms))
    eval_str = [structname_fields{i,:},'=','parms.',structname_fields{i,:},';'];
    eval(eval_str);
end
structname_fields = fields(syst);
for i = 1:size(fields(syst))
    eval_str = [structname_fields{i,:},'=','syst.',structname_fields{i,:},';'];
    eval(eval_str);
end

Cx = subs(Cx, {'phi1','phi2','dphi1','dphi2'},...
    [x0(1),x0(2),x0(3),x0(4)]);

Cdp = subs(Cdp, {'phi1','phi2','dphi1','dphi2'},...
    [x0(1),x0(2),x0(3),x0(4)]);

Ct = subs(Ct, {'phi1','phi2','dphi1','dphi2'},...
    [x0(1),x0(2),x0(3),x0(4)]);

Cdt = subs(Cdt, {'phi1','phi2','dphi1','dphi2'},...
    [x0(1),x0(2),x0(3),x0(4)]);


% Create matrices
M = diag([parms.m,parms.m,parms.I,parms.m,parms.m,parms.I]);
A = [M Cx';Cx zeros(5,5)];
F = [parms.m*parms.g 0 0 parms.m*parms.g 0 0]';
b = [F;-(Cdp+Cdt+Ctt)];
```

```
X = A\b;

end
```

Appendix C – Impulsive impact force

```
%% MBD_B: Assignment 3
% Question 3 - impact

%  Rick Staa (4511328)
%  Last edit: 05/03/2018
clear all; close all; % clc;

%% Parameters
% Segment 1
parms.L     = 0.55;                                                 % [m]
parms.w     = 0.05;                                                 % [m]
parms.t     = 0.004;                                                % [m]
parms.p     = 1180;                                                 % [kg/m^3]
parms.m     = parms.p * parms.w * parms.t * parms.L;                % [kg]
parms.I     = (1/12) * parms.m * parms.L^2;                         % [kg*m^2]

% World parameters
parms.g      = 9.81;                    % [m/s^2]

% Motor constraint
omega = 120*(2*pi/60);
parms.e = 0.5; % Coefficient of restitution

%% Compute constraint matrices

% Use symbolic toolbox to calculate derivatives
syms x1 y1 phi1 x2 y2 phi2 t
syms dx1 dy1 dphi1 dx2 dy2 dphi2
x = [x1 y1 phi1 x2 y2 phi2];
dx = [dx1 dy1 dphi1 dx2 dy2 dphi2];
parms.x = x;
parms.dx = dx;

% The normal constrain equations
C = [x1-(parms.L/2)*cos(phi1); y1-(parms.L/2)*sin(phi1) ;   ...
     (x2-(parms.L/2)*cos(phi2))-(x1+(parms.L/2)*cos(phi1));  ...
     (y2-(parms.L/2)*sin(phi2))-(y1+(parms.L/2)*sin(phi1));  ...
     (x2 + (parms.L/2)*cos(phi2))]; % Extra constraint

% Calculate the jacobian to create the constraint equations
Cx = jacobian(C,x);
Cx = simplify(Cx);

% Put Csx Cdp Cd cx in parms struct and feed tem into the solver
syst.Cx     = Cx;

%% C:
parms.phi1_0  = pi/2;   % angle of bar 1 (with x-axis)
parms.phi2_0  = pi/2;   % angle of bar 2
dphi1_0 = 120*(2*pi/60);  % angular velocity of bar 1
dphi2_0 = 120*(2*pi/60);  % angular velocity of bar 2
dx1_0 = 0;%-(1/2)*sin(phi1_init)*dphi1_init;
dx2_0 = 0;%-(3*1/2)*sin(phi1_init)*dphi1_init;
dy1_0 = 0;
dy2_0 = 0;

% Calculate impatct forces
x0  = [dx1_0;dy1_0;dphi1_0;dx2_0;dy2_0;dphi2_0];
[X] = state_calc(x0,parms,syst);
X   = double(vpa(X));
M = diag([parms.m,parms.m,parms.I,parms.m,parms.m,parms.I]);
KE_before = 0.5*x0'*M*x0;
KE_after = 0.5*X(1:6)'*M*X(1:6);
```

```matlab
%% A: Create state space matrices for the case when we add a spring
function [X] = state_calc(x0,parms,syst)

% Get system matrices out
structname_fields = fields(parms);
for i = 1:size(fields(parms))
    eval_str = [structname_fields{i,:},'=','parms.',structname_fields{i,:},';'];
    eval(eval_str);
end
structname_fields = fields(syst);
for i = 1:size(fields(syst))
    eval_str = [structname_fields{i,:},'=','syst.',structname_fields{i,:},';'];
    eval(eval_str);
end

Cx = subs(Cx, {'phi1','phi2','dphi1','dphi2'},...
    [parms.phi1_0,parms.phi2_0,x0(3),x0(6)]);

% Create matrices

M = diag([parms.m,parms.m,parms.I,parms.m,parms.m,parms.I]);
A = [M Cx';Cx zeros(5,5)];
F = [M*x0];
b = [F;-parms.e*Cx*x0];
X = A\b;

end
```