Kafrelshiekh University
Mechanical Engineering Department
Mechatronics Program

Serial Manipulator Dynamics

Lagrangian Formulation using MATLAB

Eman M.ElRify

Supervisor: Prof. Roshdy Abo-Shanab

Course: Industrial Robots Dynamics , Control and Simulation

Code: 40143

Outlines

- Introduction
- Collecting Data
- Forward Kinematics
- Jacobian Submatrices
- Manipulator Inertia Matrix
- Velocity Coupling Vector
- Gravitational Vector
- Results

Introduction

- A dynamical equation of motion can be formulated by several methods. One approach is Lagrange's equation of motion. Lagrangian formulation eliminates the constraint forces.
- **Lagrangian function** is defines as the difference between kinetic and potential energy in a mechanical system.

$$L = K - U$$

- Lagrange's equation of motion:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} + \frac{\partial U}{\partial q_i} = \tau_i$$

Introduction

- The state-space equation:

$$\boldsymbol{M\ddot{q} + V + G + F = \tau}$$

$$\begin{bmatrix} M_{11} & .. & .. & M_{1n} \\ : & . & . & : \\ : & . & . & : \\ M_{n1} & .. & .. & M_{nn} \end{bmatrix} \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \\ : \\ \ddot{q}_n \end{bmatrix} + \begin{bmatrix} C_{11} & .. & .. & C_{1n} \\ : & . & . & : \\ : & . & . & : \\ C_{n1} & .. & .. & C_{nn} \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ : \\ \dot{q}_n \end{bmatrix} + \begin{bmatrix} G_1 \\ G_2 \\ : \\ G_n \end{bmatrix} + \begin{bmatrix} F_1 \\ F_2 \\ : \\ F_n \end{bmatrix} = \begin{bmatrix} \tau_1 \\ \tau_2 \\ : \\ \tau_n \end{bmatrix}$$

$\boldsymbol{M}$: *Manipulator Inertia Matrix*

$\boldsymbol{V}$: *Velocity Coupling Vector*

$\boldsymbol{G}$: *Gravitational Vector*

$\boldsymbol{F}$: *Friction Forces Vector*

$\boldsymbol{\tau}$: *Generalized forces Vector*

Introduction

Assumptions

- Resistance forces are neglected: $F = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$ $\boldsymbol{M}\ddot{\boldsymbol{q}} + \boldsymbol{V} + \boldsymbol{G} = \boldsymbol{\tau}$

- Each link has a rectangular surface and cross-section area

$$I_i = \frac{1}{12} m_i a_i^2 \begin{bmatrix} (w_i^2 + h_i^2) & 0 & 0 \\ 0 & (a_i^2 + h_i^2) & 0 \\ 0 & 0 & (w_i^2 + a_i^2) \end{bmatrix}$$

Collecting Data

.xlsx file for parameters

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	**Ji**	**alpha_i**	**a_i**	**d_i**	**m_i**	**flag_i**	**w_i**	**h_i**	**theta_i**				Type	Revolute	Prismatic
2	1	0	a_1	d_1	m_1	1	0	0	theta_1				Flag	1	0
3	2	3.141593	a_2	0	m_2	1	0	0	theta_2		DOF =	3			
4	3	0	0	d_3	m_3	0	0	l	0				Quantity	notation	Unit
5													Link Twist	alpha_i	deg
6													Link Length	a_i	cm
7													Link Width	w_i	cm
8													Link Thickness	h_i	cm
9													Link offset	d_i	cm
10													Joint angle	theta_i	deg
11													Mass of link i	m_i	kg
12															

SCARA Robot Parameters

Note : the shots in this PowerPoint file is taken after the execution of the program with the above parameters

Collecting Data

Paremeters are introduced to MATLAB by reading .xlsx file

Collecting Data

Reading Excel File

```
% Collecting Data
clc
clear all
file_Of_parameters ='SCARA.xlsx';
opts = detectImportOptions(file_Of_parameters);
preview(file_Of_parameters,opts) %display the file
cells = readmatrix(file_Of_parameters,'Range','K3:L3'); %read DOF data
DOF = cells(2) %get the DOF number e.g.(1,2,3,...)
dh_Range =char(compose('A1:I%d',DOF+1)); %Only Cells of the links parameters (link number,link twist,link length,
                                         % link offset,link mass,flag, link width, lenth thickness,joint angle)
Table = readtable(file_Of_parameters,'Range',dh_Range) %display the used parameters
% string(Table{:,3})
```

Collecting Data

Introducing Symbols for the non – numeric values

```
%introducing Symbols for unknown existing parameters
syms 'j_%d' 'alpha_%d' 'a_%d' 'd_%d' 'theta_%d' 'm_%d' 'flag_%d' 'h_%d' 'w_%d' 'l_%d' [DOF,1]
sym('l');
alpha_ %preview a symbolic variable
```

alpha_ =

$$\begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{pmatrix}$$

Collecting Data

Assigning parameters to column vector and conerting data type

Assigning parameters to column vectors

```
%Assigning the values in the file to column vectors
[j,alpha,a,d,m,flag,w,h,theta]=readvars(file_Of_parameters,'Range',dh_Range)
str2double(theta)
str2double(alpha)
str2double(a)
str2double(d)
str2double(h)
str2double(w)
str2double(m)
for i = 1:DOF

    theta_(i)=theta(i);
    alpha_(i)=alpha(i);
    a_(i)=a(i);
    d_(i)=d(i);
    w_(i)=w(i);
    h_(i)=h(i);
    flag_(i)= flag(i);

end
```

Collecting Data

Assigning parameters to column vector and conerting data type

Assigning parameters to column vectors

```
%Assigning the values in the file to column vectors
[j,alpha,a,d,m,flag,w,h,theta]=readvars(file_Of_parameters,'Range',dh_Range)
str2double(theta)
str2double(alpha)
str2double(a)
str2double(d)
str2double(h)
str2double(w)
str2double(m)
for i = 1:DOF

    theta_(i)=theta(i);
    alpha_(i)=alpha(i);
    a_(i)=a(i);
    d_(i)=d(i);
    w_(i)=w(i);
    h_(i)=h(i);
    flag_(i)= flag(i);

end
```

Read data as a column vectors

Converting string data type into double

Assigning values to the symbolic functions

Collecting Data

Cleaning Data (Replacing NaN values with 0)

Cleaning Data

Replacing NaN values with zero

```
alpha_(isnan(alpha_))=0
theta_(isnan(theta_))=0
a_(isnan(a_))=0
h_(isnan(h_))=0
w_(isnan(w_))=0
m_(isnan(m_))=0
d_(isnan(d_))=0
flag_(isnan(flag_))=0
```

alpha_ =

$$\begin{pmatrix} 0 \\ \pi \\ 0 \end{pmatrix}$$

w_ =

$$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

theta_ =

$$\begin{pmatrix} \theta_1 \\ \theta_2 \\ 0 \end{pmatrix}$$

m_ =

$$\begin{pmatrix} m_1 \\ m_2 \\ m_3 \end{pmatrix}$$

a_ =

$$\begin{pmatrix} a_1 \\ a_2 \\ 0 \end{pmatrix}$$

d_ =

$$\begin{pmatrix} d_1 \\ 0 \\ d_3 \end{pmatrix}$$

h_ =

$$\begin{pmatrix} 0 \\ 0 \\ l \end{pmatrix}$$

flag_ =

$$\begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$$

Forward Kinematics

MATLAB®

D-H Transformation Matrix: $A_i^{i-1}\begin{bmatrix} c\theta_i & -c\alpha_i s\theta_i & s\alpha_i s\theta_i & a_i c\theta_i \\ s\theta_i & c\alpha_i c\theta_i & -s\alpha_i c\theta_i & a_i s\theta_i \\ 0 & s\alpha_i & c\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}, (A_1^0, A_2^1, \ldots A_n^{n-1})$

Forward Kinematics

$$A_i^{i-1} = \begin{bmatrix} c\theta_i & -c\alpha_i\, s\theta_i & s\alpha_i\, s\theta_i & a_i\, c\theta_i \\ s\theta_i & c\alpha_i\, c\theta_i & -s\alpha_i\, c\theta_i & a_i\, s\theta_i \\ 0 & s\alpha_i & c\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

```
%Forward Kinematics
for i = 1:DOF
    A{i}=[cos(theta_(i)) -cos(alpha_(i))*sin(theta_(i)) sin(alpha_(i))*sin(theta_(i)) a_(i)*cos(theta_(i));
        sin(theta_(i)) cos(alpha_(i))*cos(theta_(i)) -sin(alpha_(i))*cos(theta_(i)) a_(i)*sin(theta_(i));
        0 sin(alpha_(i)) cos(alpha_(i)) d_(i);0 0 0 1];
end
```

Forward Kinematics

Calculating: $A_1^0, A_2^0 \ldots A_n^0$ $A_n^0 = A_1^0 A_2^1 \ldots A_n^{n-1}$

$$A_0^n = A_0^1 A_2^1 \ldots A_{n-1}^n$$

$$A_0^{n-1} = A_0^1 A_2^1 \ldots A_{n-2}^{n-1}$$

:

```
T{1}=A{1};
for i = 1:DOF-1
    T{i+1}=T{i}*A{i+1}
    T{i+1}=simplify(T{i+1})
end
for i = 1:DOF
    display(compose('A%d_0',i))
    T{i}
end
```

Forward Kinematics

Calculating: $A_1^0, A_2^0 \ldots A_n^0$ $\qquad$ $A_n^0 = A_1^0 A_2^1 \ldots A_n^{n-1}$

Result:

```
{'A1_0'}

ans =
```

$$\begin{pmatrix} \cos(\theta_1) & -\sin(\theta_1) & 0 & a_1\cos(\theta_1) \\ \sin(\theta_1) & \cos(\theta_1) & 0 & a_1\sin(\theta_1) \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
1×1 cell array

{'A2_0'}

ans =
```

$$\begin{pmatrix} \cos(\theta_1+\theta_2) & \sin(\theta_1+\theta_2) & 0 & a_2\cos(\theta_1+\theta_2)+a_1\cos(\theta_1) \\ \sin(\theta_1+\theta_2) & -\cos(\theta_1+\theta_2) & 0 & a_2\sin(\theta_1+\theta_2)+a_1\sin(\theta_1) \\ 0 & 0 & -1 & d_1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
{'A3_0'}

ans =
```

$$\begin{pmatrix} \cos(\theta_1+\theta_2) & \sin(\theta_1+\theta_2) & 0 & a_2\cos(\theta_1+\theta_2)+a_1\cos(\theta_1) \\ \sin(\theta_1+\theta_2) & -\cos(\theta_1+\theta_2) & 0 & a_2\sin(\theta_1+\theta_2)+a_1\sin(\theta_1) \\ 0 & 0 & -1 & d_1-d_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Jacobian Submatrices

Jacobian Submatrices

$$J_{vi} = \left[J^1_{vi}, J^2_{vi}, \ldots, J^i_{vi}, 0, 0, \ldots, 0\right]$$

$$J_{\omega i} = \left[J^1_{\omega i}, J^2_{\omega i}, \ldots, J^i_{\omega i}, 0, 0, \ldots, 0\right]$$

$$J^j_{vi} = z_{j-1} \times p^{j-1}_{ci} * \quad \text{for } a \text{ revolute joint}$$

$$J^j_{vi} = z_{j-1} \quad \text{for } a \text{ prismatic joint}$$

$$J^j_{\omega i} = z_{j-1} \quad \text{for } a \text{ revolute joint}$$

$$J^j_{\omega i} = 0 \quad \text{for } a \text{ prismatic joint}$$

$$J^j_{vi} = 0 \quad \text{for } \; j > i$$

$$J^j_{\omega i} = 0 \quad \text{for } \; j > i$$

$$p^{j-1}_{ci} * = p_{ci} - p_{j-1}$$

$$p_{ci} = A^0_i \, p^i_{ci}$$

$$p^0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \qquad z_0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$z_{j-1} = A^0_{j-1} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

$$p^0_{j-1} = A^0_{j-1} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Jacobian Submatrices

```
%Jacobian
syms 'Jv_%d' 'Jw_%d' [DOF]
c=1; %a avriable uses as an index for storing the jacobian submatrices and extracting it later
for j= 1:DOF

    if j>1
        Pi0 = T{j-1}*transpose([0 0 0 1]) ;
        Pi0 = Pi0(1:3);
        z = T{j-1}*transpose([0 0 1 0]) ;
        z = z(1:3);
    else
        Pi0 = transpose([0 0 0]); %P00 =[0 0 0]'
        z = transpose([0 0 1]);
    end

    for i = DOF:-1:1


        if flag_(j)==1

            Pcii = transpose([-1/2*a_(i) 0 0 1]) ;
        else
            Pcii = transpose([0 0 -1/2*h_(i) 1]) ;
        end
```

Jacobian Submatrices

MATLAB®

```
%Jacobian
syms 'Jv_%d' 'Jw_%d' [DOF]
c=1; %a avriable uses as an index for storing the jacobian submatrices and extracting it later
for j= 1:DOF

    if j>1
        Pi0 = T{j-1}*transpose([0 0 0 1]) ;
        Pi0 = Pi0(1:3);
        z = T{j-1}*transpose([0 0 1 0]) ;
        z = z(1:3);
    else
        Pi0 = transpose([0 0 0]); %P00 =[0 0 0]'
        z = transpose([0 0 1]);
    end

    for i = DOF:-1:1


        if flag_(j)==1

            Pcii = transpose([-1/2*a_(i) 0 0 1]) ;
        else
            Pcii = transpose([0 0 -1/2*h_(i) 1]) ;
        end
```

set initial value for P_{j-1}^0

$$P_0^0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

and for z_{j-1}

$$z_0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Jacobian Submatrices

```
        if flag_(j)==1 %if the joint is revolute
            Pci0 = T{i}*Pcii;
            Pci0 = Pci0(1:3,1);
            pci0_star = Pci0-Pi0;
            jv=cross(z,pci0_star); %cross product
            jw =z;
        elseif flag_(j)==0
            jv =z;
            jw =transpose([0 0 0]);
        end
        if i<j % if j > i Jvij = 0   Jwij =0
            Pi0 = transpose([0 0 0]);
            z = [0 0 0]';
            jv =Pi0;
            jw =transpose([0 0 0]);
        end
        Jv{c}=jv;
        Jw{c}=jw;
        c=c+1;
        jv
        jw
    end
end
```

Jacobian Submatrices

```
        if flag_(j)==1 %if the joint is revolute
            Pci0 = T{i}*Pcii;
            Pci0 = Pci0(1:3,1);
            pci0_star = Pci0-Pi0;
            jv=cross(z,pci0_star); %cross product
            jw =z;
        elseif flag_(j)==0
            jv =z;
            jw =transpose([0 0 0]);
        end
        if i<j % if j > i Jvij = 0   Jwij =0
            Pi0 = transpose([0 0 0]);
            z = [0 0 0]';
            jv =Pi0;
            jw =transpose([0 0 0]);
        end
        Jv{c}=jv;
        Jw{c}=jw;
        c=c+1;
        jv
        jw
    end
end
```

$$if\ j > i\ then\ J_{vi}^{j} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}\ and\ J_{\omega i}^{j} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Jacobian Submatrices

MATLAB

```
        if flag_(j)==1 %if the joint is revolute
            Pci0 = T{i}*Pcii;
            Pci0 = Pci0(1:3,1);
            pci0_star = Pci0-Pi0;
            jv=cross(z,pci0_star); %cross product
            jw =z;
        elseif flag_(j)==0
            jv =z;
            jw =transpose([0 0 0]);
        end
        if i<j % if j > i Jvij = 0   Jwij =0
            Pi0 = transpose([0 0 0]);
            z = [0 0 0]';
            jv =Pi0;
            jw =transpose([0 0 0]);
        end
        Jv{c}=jv;
        Jw{c}=jw;
        c=c+1;
        jv
        jw
    end
end
```

Creating an array with the Jacobian vectors

Jacobian Submatrices

MATLAB®

Check the resulted values

```
% Jv{1}
% Jv{2}
% Jv{3}
% Jv{4}
% Jv{5}
% Jv{6}
% Jv{7}
% Jv{8}
% Jv{9}
```

```
%example DOF =3
%Pc30* Pc20* Pc10* Pc31* Pc21* Pc11* Pc32* Pc22* Pc12*
%jv31 jv21 jv11 jv32 jv22 jv12 jv33 jv23 jv13
```

Set of matrices in particular order resulted from the loop

Jacobian Submatrices

MATLAB®

Concatenating the vectors in one single array for each link i

$$J_{vi} = \left[J^{1}_{vi}, J^{2}_{vi}, \ldots, J^{i}_{vi}, 0, 0, \ldots, 0\right]$$

$$J_{\omega i} = \left[J^{1}_{\omega i}, J^{2}_{\omega i}, \ldots, J^{i}_{\omega i}, 0, 0, \ldots, 0\right]$$

```
%Continue Jacobian
T{1}(1:4,4);

for i = 1:DOF
    Jvi{i}=[];
end
for i = 1:DOF
    for j =1:DOF
        Jvi{i}=horzcat(Jvi{i},Jv{j*(DOF)-i+1});

    end
end

for i = 1:DOF
    Jwi{i}=[];
end
for i = 1:DOF
    for j =1:DOF
        Jwi{i}=horzcat(Jwi{i},Jw{j*(DOF)-i+1});
    end
end
```

Jacobian Submatrices

MATLAB®

Checking the Jacobian Matrix for each link

Checking jacobian submatrices

```
Jvi{1}
simplify(Jvi{2})
simplify(Jvi{3})

Jwi{1}
Jwi{2}
Jwi{3}
```

Jacobian Submatrices

Checking the Jacobian Matrix for each link

ans =

$$\begin{pmatrix} -\frac{a_1 \sin(\theta_1)}{2} & 0 & 0 \\ \frac{a_1 \cos(\theta_1)}{2} & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

ans =

$$\begin{pmatrix} -\frac{a_2 \sin(\theta_1+\theta_2)}{2} - a_1 \sin(\theta_1) & -\frac{a_2 \sin(\theta_1+\theta_2)}{2} & 0 \\ \frac{a_2 \cos(\theta_1+\theta_2)}{2} + a_1 \cos(\theta_1) & \frac{a_2 \cos(\theta_1+\theta_2)}{2} & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

ans =

$$\begin{pmatrix} -a_2 \sin(\theta_1+\theta_2) - a_1 \sin(\theta_1) & -a_2 \sin(\theta_1+\theta_2) & 0 \\ a_2 \cos(\theta_1+\theta_2) + a_1 \cos(\theta_1) & a_2 \cos(\theta_1+\theta_2) & 0 \\ 0 & 0 & -1 \end{pmatrix}$$

```
ans = 3×3

     0     0     0
     0     0     0
     1     0     0
```

ans =

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix}$$

ans =

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix}$$

Manipulator Inertia Matrix

MATLAB

$$M = \sum_{i=1}^{n} \left({J_{v\,i}}^T \; m_i \; J_{v\,i} + {J_{\omega\,i}}^T I_i J_{\omega\,i} \right)$$

Link Inertia Matrix I_i

$$I_i = \frac{1}{12} m_i a_i^2 \begin{bmatrix} \left(w_i^2 + h_i^2\right) & 0 & 0 \\ 0 & \left(a_i^2 + h_i^2\right) & 0 \\ 0 & 0 & \left(w_i^2 + a_i^2\right) \end{bmatrix}$$

Manipulator Inertia Matrix

MATLAB®

Link Inertia Matrix I_i

$$I_i = \frac{1}{12} m_i a_i^2 \begin{bmatrix} (w_i^2 + h_i^2) & 0 & 0 \\ 0 & (a_i^2 + h_i^2) & 0 \\ 0 & 0 & (w_i^2 + a_i^2) \end{bmatrix}$$

```
%inertial Matrix_Moment of inertia
%the link has a rectangular shape length a width w height h
% a-> x  w->y  h->z
syms 'Ixx_%d' 'Iyy_%d' 'Izz_%d' 'Ixy_%d' 'Ixz_%d' 'Iyz_%d' [DOF 1]
for i = 1:DOF
    Ixx_(i) = 1/12*m_(i)*(w_(i)^2 + h_(i)^2);
    Iyy_(i) = 1/12*m_(i)*(a_(i)^2 + h_(i)^2);
    Izz_(i) = 1/12*m_(i)*(a_(i)^2 + w_(i)^2);
    Ixy_(i) = 0;
    Ixz_(i) = 0;
    Iyz_(i) = 0;
    I{i}=[Ixx_(i) -Ixy_(i) -Ixz_(i);-Ixy_(i) Iyy_(i)...
         -Iyz_(i);-Ixz_(i) --Iyz_(i) Izz_(i)];
    I{i}=simplify(I{i});


end
```

Manipulator Inertia Matrix

$$\boldsymbol{M} = \sum_{i=1}^{n} (J_{vi}^{T}\, m_i\, J_{vi} + J_{\omega i}^{T}\, I_i\, J_{\omega i})$$

```
%Manipulator Inertia Matrix
M=zeros(DOF);
for i = 1:DOF
    M=M+m_(i)*transpose(Jvi{i})*Jvi{i} + transpose(Jwi{i})*I{i}*Jwi{i};
    M=simplify(M);
end
```

Manipulator Inertia Matrix

$$\boldsymbol{M} = \sum_{i=1}^{n} (\boldsymbol{J}_{vi}^{T}\, m_i\, \boldsymbol{J}_{vi} + \boldsymbol{J}_{\omega i}^{T}\, \boldsymbol{I}_i\, \boldsymbol{J}_{\omega i})$$

```
M =
```

$$\begin{pmatrix} \frac{a_1{}^2 m_1}{3} + a_1{}^2 m_2 + a_1{}^2 m_3 + \frac{a_2{}^2 m_2}{3} + a_2{}^2 m_3 + a_1\, a_2\, m_2 \cos(\theta_2) + 2\, a_1\, a_2\, m_3 \cos(\theta_2) & \sigma_1 & 0 \\ \sigma_1 & \frac{a_2{}^2\, (m_2 + 3\, m_3)}{3} & 0 \\ 0 & 0 & m_3 \end{pmatrix}$$

where

$$\sigma_1 = \frac{a_2\, (2\, a_2\, m_2 + 6\, a_2\, m_3 + 3\, a_1\, m_2 \cos(\theta_2) + 6\, a_1\, m_3 \cos(\theta_2))}{6}$$

Velocity Coupling Vector

Coriolis Matrix

$$\mathrm{B} = \frac{\partial M}{\partial q_i}\dot{\boldsymbol{q}}$$

$$V = C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}$$

```
%V=C*q_dot
%C=B-0.5Bt Coriollis Matrix B=[B1 B2...Bn]
B=[];
syms 'theta_dot_%d' 'theta_ddot_%d' [DOF 1]
for i =1:DOF
    Bi{i}=diff(M,theta_(i))*theta_dot_;
    B = horzcat(B,Bi{i}); %concatenate the column matrices

end
```

Velocity Coupling Vector

Coriolis Matrix

$$\mathrm{B} = \frac{\partial M}{\partial q_i}\dot{\boldsymbol{q}}$$

$$V = C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}$$

B =

$$\begin{pmatrix} 0 & -\dot{\theta}_1\,(a_1\,a_2\,m_2\sin(\theta_2) + 2\,a_1\,a_2\,m_3\sin(\theta_2)) - \frac{a_2\,\dot{\theta}_2\,(3\,a_1\,m_2\sin(\theta_2) + 6\,a_1\,m_3\sin(\theta_2))}{6} & 0 \\ 0 & -\frac{a_2\,\dot{\theta}_1\,(3\,a_1\,m_2\sin(\theta_2) + 6\,a_1\,m_3\sin(\theta_2))}{6} & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Velocity Coupling Vector

Centrifugal Coefficients Matrix

$$C(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \mathrm{B} - \frac{1}{2}\boldsymbol{B}^T$$

$$V = C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}$$

```
C = B - 0.5*transpose(B);
C=simplify(C)
```

Velocity Coupling Vector

Centrifugal Coefficients Matrix

$$V = C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}$$

$$C(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \mathrm{B} - \frac{1}{2}\boldsymbol{B}^T$$

C =

$$\begin{pmatrix} 0 & -\frac{a_1\, a_2 \sin(\theta_2)\; (m_2 + 2\, m_3)\; \left(2\, \dot{\theta}_1 + \dot{\theta}_2\right)}{2} & 0 \\ \frac{a_1\, a_2 \sin(\theta_2)\; (m_2 + 2\, m_3)\; \left(2\, \dot{\theta}_1 + \dot{\theta}_2\right)}{4} & -\frac{a_1\, a_2\, \dot{\theta}_1 \sin(\theta_2)\; (m_2 + 2\, m_3)}{4} & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Gravitational Vector

Gravitational Acceleration Vector *g*

The **g** vector is determined according to the first link twist

$$if\ \alpha_1 = 0 \,\&\, d_1 = 0{:}\ g = \begin{bmatrix} 0 \\ -g_c \\ 0 \end{bmatrix}$$

$$if\ \alpha_1 = \frac{\pi}{2}\, or\ \frac{-\pi}{2} :\ g = \begin{bmatrix} 0 \\ 0 \\ -g_c \end{bmatrix}$$

```
%Gravitational Matrix G=[G1;G2;...;Gn]
gc = sym('g_c')
%gc =9.81; %m/s^s
if (alpha_(1)==0) & (d_(1)==0)
    g = [0;-gc;0];
    r=2;
else
    g = [0;0;-gc];
    r=3;
end
```

Gravitational Vector

Gravitational Vector $$G_i = -\sum_{j=1}^{n} m_j g^T J_{v\ j}^{i}$$

```
G=[];
Gj=0;
n=0;
for i =1:DOF
    Gj=0;
    s = DOF; % to get the right order in jacobian submatrices array
    for j=1:DOF
        v=s+n;
        Gj=Gj-(m_(j)*transpose(g)*Jv{v})
        Gi{i} = Gj;
        s= s-1;
    end
    n=n+DOF;
    G = vertcat(G,Gi{i});
end
G=simplify(G)
```

Gravitational Vector

Gravitational Vector $$G_i = -\sum_{j=1}^{n} m_j g^T J_{v\ j}^{i}$$

```
G =
```

$$\begin{pmatrix} 0 \\ 0 \\ -g_c m_3 \end{pmatrix}$$

Generalized Forces Vector (Result)

MATLAB®

Selecting the variable parameters

Generalized Forces Vector

$M(q)\ddot{q} + V(q,\dot{q}) + G(q) = \tau$

```
syms 'tau_%d' [DOF 1]
q_ddot=[];
q_dot=[];
for i=1:DOF
    if flag_(i)==1
        q_ddot=vertcat(q_ddot,theta_ddot_(i));
        q_dot=vertcat(q_dot,theta_dot_(i));
    else
        q_ddot=vertcat(q_ddot,d_ddot_(i));
        q_dot=vertcat(q_dot,d_dot_(i));
    end
end
```

Selecting the variable parameters

Generalized Forces Vector (Result)

Selecting the variable parameters

```
q_ddot =
```

$$\begin{pmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \\ \ddot{d}_3 \end{pmatrix}$$

```
q_dot =
```

$$\begin{pmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{d}_3 \end{pmatrix}$$

Generalized Forces Vector (Result)

MATLAB®

Generalized Forces Vector

$$M\ddot{q} + V + G + F = \tau$$

```
tau_ = M*q_ddot + C*q_dot +G;
tau_=simplify(tau_)

display('tau_1')
simplify(tau_(1))
display('tau_2')
simplify(tau_(2))
display('tau_3')
simplify(tau_(3))
```

Generalized Forces Vector (Result)

MATLAB

Generalized Forces Vector

$$\boldsymbol{M\ddot{q} + V + G + F = \tau}$$

```
tau_ =
```

$$\begin{pmatrix} \ddot{\theta}_1 \left(\frac{{a_1}^2 m_1}{3} + {a_1}^2 m_2 + {a_1}^2 m_3 + \frac{{a_2}^2 m_2}{3} + {a_2}^2 m_3 + a_1 a_2 m_2 \cos(\theta_2) + 2 a_1 a_2 m_3 \cos(\theta_2)\right) + \frac{a_2 \ddot{\theta}_2 \sigma_1}{6} - \frac{a_1 a_2 \dot{\theta}_2 \sin(\theta_2)\ (m_2 + 2 m_3)\ \left(2 \dot{\theta}_1 + \dot{\theta}_2\right)}{2} \\ \frac{{a_2}^2 \ddot{\theta}_2\ (m_2 + 3 m_3)}{3} + \frac{a_2 \ddot{\theta}_1 \sigma_1}{6} - \frac{a_1 a_2 \dot{\theta}_1 \dot{\theta}_2 \sin(\theta_2)\ (m_2 + 2 m_3)}{4} + \frac{a_1 a_2 \dot{\theta}_1 \sin(\theta_2)\ (m_2 + 2 m_3)\ \left(2 \dot{\theta}_1 + \dot{\theta}_2\right)}{4} \\ m_3\ \left(\ddot{d}_3 - g_c\right) \end{pmatrix}$$

where

$$\sigma_1 = 2 a_2 m_2 + 6 a_2 m_3 + 3 a_1 m_2 \cos(\theta_2) + 6 a_1 m_3 \cos(\theta_2)$$