Robotics & Automation

HW-6

1.1 For $||F_s|| = ||F_b||$, and $f_s \neq 0$ and $m_s \neq 0$,

The force applied should be at a point where the moment is equal from each of the frames.

→ f_s and f_b are required to be in the same direction (or opposite but equal magnitude of forces).

→ But if the moment is the same for both the frames' perspectives, then the absolute magnitude remains same for the vectors

$||F_s|| = ||F_b||$

$$F_s = \begin{bmatrix} r_s \times f_s \\ f_s \end{bmatrix} \quad \& \quad F_b = \begin{bmatrix} r_b \times f_b \\ f_b \end{bmatrix}$$

We know,

→ $m_b = R^T((-p + r_a) \times f_a)$

Let's say :-

$\Rightarrow (-p + r_s) = r_b$

$\Rightarrow (r_s - r_b) = p$

Considering $(p = 0)$.

$\Rightarrow (r_s = r_b)$ [Considering the equal and opposite directions too].

Similarly, for f_s and f_b;

$f_s = f_b$ (regarding the direction of force applied in each of the frames).

$\cancel{\|F_s\| = \sqrt{(f_s \times f_s)^2}}$

$\therefore (r_s \times f_s) = (r_f \times f_b)$

$\Rightarrow f_s = f_b$ (both directions possibility)

Are the conditions required to be fulfilled for

$\|F_s\| = \|F_b\|$

$\boxed{1.2}$ Body frame on prismatic joint :

We find T_{sb} ;

$$T_{sb} = \begin{bmatrix} 0 & 1 & 0 & L \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2L \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

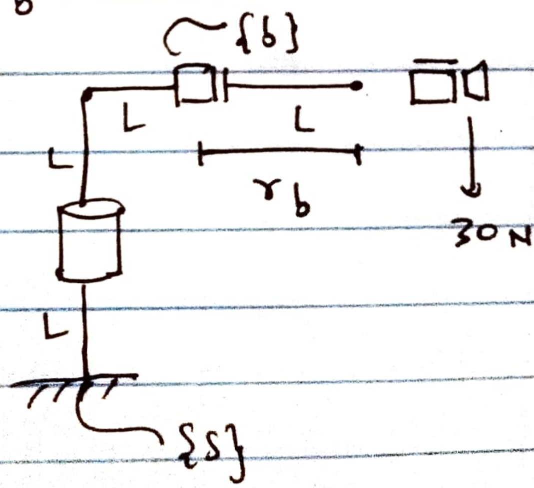
We know ;

$$F_b = \begin{bmatrix} m_b \\ f_b \end{bmatrix}$$; we can find f_b as

$$f_b = \begin{bmatrix} 0 \\ 0 \\ -30 \end{bmatrix}$$ Force acting along the $-z_b$- axis

To find <u>moment</u> m_b :

$\Rightarrow$ $m_b = (r_b \times f_b)$



$\therefore \quad r_b = \begin{bmatrix} 0 \\ L \\ 0 \end{bmatrix}$ in frame b's perspective

$$\therefore \quad f_b = \begin{bmatrix} m_b \\ f_b \end{bmatrix} = \begin{bmatrix} r_b \times f_b \\ f_b \end{bmatrix}$$

$$\Rightarrow \boxed{f_b = \begin{bmatrix} -30L \\ 0 \\ 0 \\ 0 \\ 0 \\ -30 \end{bmatrix}}$$

1.3 We found T_{sb} :

$$T_{sb} = \begin{bmatrix} 0 & 1 & 0 & L \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2L \\ & 0 & & 1 \end{bmatrix}$$

To find F_s, we use r/ship :-

$$F_s = (Ad_{T_{bs}})^T F_b$$

$$\Rightarrow F_s = (Ad_{T_{sb}^{-1}})^T F_b$$

Using MATLAB we calculate the $(Ad_{T_{sb}^{-1}})^T$ part.

$$\Rightarrow F_s = \begin{bmatrix} 0 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 2L & 0 & -L & 0 & -1 & 0 \\ 0 & 2L & 0 & 1 & 0 & 0 \\ 0 & -L & 0 & 0 & 0 & 1 \end{bmatrix}^T \cdot \begin{bmatrix} -30L \\ 0 \\ 0 \\ 0 \\ 0 \\ -30 \end{bmatrix}$$

$$F_s = \begin{bmatrix} 0 \\ 60L \\ 0 \\ 0 \\ 0 \\ -30 \end{bmatrix}$$

```
syms L real

Tsb = [[0 1 0 L]
    [-1 0 0 0]
    [0 0 1 2*L]
    [0 0 0 1]];

adjointTsbinv = adjointM(inv(Tsb))

Adj13 = (adjointM(inv(Tsb)))'*[-30*L;0;0;0;0;-30]
```

adjointTsbinv =

$$\begin{pmatrix} 0 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 2L & 0 & -L & 0 & -1 & 0 \\ 0 & 2L & 0 & 1 & 0 & 0 \\ 0 & -L & 0 & 0 & 0 & 1 \end{pmatrix}$$

Adj13 =

$$\begin{pmatrix} 0 \\ 60L \\ 0 \\ 0 \\ 0 \\ -30 \end{pmatrix}$$

```
clc
clear

syms theta1 theta2 theta3 real

% Problem 2.1:

% Opposite force of +15 N over positive y-axis
% Thus the wrench required is provided below with the formulae: Fb = [mb;fb] and
moment is zero as the force is directly applied over the body frame
Fb = [0;0;0;0;15;0];

theta = [theta1;theta2;theta3];


omega = [0;0;1];

q1 = [0;0;0];
q2 = [1;0;0];
q3 = [2;0;0];

S1 = [omega; -cross(omega, q1)];
S2 = [omega; -cross(omega, q2)];
S3 = [omega; -cross(omega, q3)];

S_eq = [S1, S2, S3];
M = [eye(3), [3;0;0]; 0 0 0 1];

% T with initial joint positions
T_0 = simplify(expand(fk(M, S_eq, theta)))
```

T_0 =

$$\begin{pmatrix} \cos(\theta_1+\theta_2+\theta_3) & -\sin(\theta_1+\theta_2+\theta_3) & 0 & \cos(\theta_1+\theta_2+\theta_3)+\cos(\theta_1+\theta_2)+\cos(\theta_1) \\ \sin(\theta_1+\theta_2+\theta_3) & \cos(\theta_1+\theta_2+\theta_3) & 0 & \sin(\theta_1+\theta_2+\theta_3)+\sin(\theta_1+\theta_2)+\sin(\theta_1) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
R_0 = T_0(1:3, 1:3);
JS = simplify(expand(JacS(S_eq, theta))) %Space Jacobian
```

JS =

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

```
Jb = simplify(expand(adjointM(inv(T_0))*JS)) %Body Jacobian
```

Jb =

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \\ \sin(\theta_2+\theta_3)+\sin(\theta_3) & \sin(\theta_3) & 0 \\ \cos(\theta_2+\theta_3)+\cos(\theta_3)+1 & \cos(\theta_3)+1 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

```
J_geometric = simplify(expand([R_0, zeros(3); zeros(3), R_0] * Jb)) %Geometric
Jacobian
```

J_geometric =

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \\ -\sigma_1-\sin(\theta_1+\theta_2)-\sin(\theta_1) & -\sigma_1-\sin(\theta_1+\theta_2) & -\sigma_1 \\ \sigma_2+\cos(\theta_1+\theta_2)+\cos(\theta_1) & \sigma_2+\cos(\theta_1+\theta_2) & \sigma_2 \\ 0 & 0 & 0 \end{pmatrix}$$

where

$$\sigma_1 = \sin(\theta_1+\theta_2+\theta_3)$$

$$\sigma_2 = \cos(\theta_1+\theta_2+\theta_3)$$

```
% Fs calculation:
Fs = simplify(expand(adjointM(inv(T_0)))'*Fb)
```

Fs =

$$\begin{pmatrix} 0 \\ 0 \\ 15\cos(\theta_2+\theta_3)+15\cos(\theta_3)+15 \\ -15\sin(\theta_1+\theta_2+\theta_3) \\ 15\cos(\theta_1+\theta_2+\theta_3) \\ 0 \end{pmatrix}$$

```
% symbolic tau calculation
tau = simplify(expand(JS'*Fs))
```

tau =

$$\begin{pmatrix} 15\cos(\theta_2+\theta_3)+15\cos(\theta_3)+15 \\ 15\cos(\theta_3)+15 \\ 15 \end{pmatrix}$$

```
% Problem 2.2: CASE 1
Case1_tau = double(subs(tau,[theta1,theta2,theta3],[0,pi/4,pi/4]))
```

```
Case1_tau = 3×1
   25.6066
   25.6066
   15.0000
```

```
% Problem 2.3: CASE 2
Case2_tau = double(subs(tau,[theta1,theta2,theta3],[0,pi/8,0]))
```

```
Case2_tau = 3×1
   43.8582
   30.0000
   15.0000
```

```
% Problem 2.4

% ||tau||
magnitude_tau = simplify(expand(norm(tau)))
```

magnitude_tau =

$$\frac{15\,\sqrt{2}\,\sqrt{\cos(2\,\theta_3)+4\cos(\theta_3)+2\,\left(\cos(\theta_2+\theta_3)+\cos(\theta_3)+1\right)^2+5}}{2}$$

```
% Maximum ||tau|| with theta1 = theta2 = theta3 = 0
% or theta1 = 100 degrees, theta2 = 0, theta3 = 0
Max__mag_tau = double(subs(magnitude_tau,[theta1,theta2,theta3],[0,0,0]))
```

```
Max__mag_tau = 56.1249
```

```
% f = (15*sqrt(sym(2))*sqrt(cos(2*theta3) + 4*cos(theta3) + 2*(cos(theta2 + theta3)
+ cos(theta3) + 1)^2 + 5))/2

% Minimum ||tau|| with theta1 = 90 degrees, theta2 = 90 degrees, theta3 = 180
degrees
Min_mag_tau = double(subs(magnitude_tau,[theta1,theta2,theta3],[-pi/2,pi/2,pi]))
```

```
Min_mag_tau = 15
```

```
% Verifying the optimal results of all theta values for maximum and minimum
% evaluation of the magnitude of joint torque values:
```

```
fprintf(['Verifying results for maximum and minimum magnitude of tau with' ...
    'optimal theta values:\n']);
```

```
Verifying results for maximum and minimum magnitude of tau withoptimal theta values:
```

```
% Define the objective function to maximize Magnitude_tau
objectiveFunction_Max = @(theta) -double(norm(subs(magnitude_tau, [theta1, theta2,
theta3], double(theta))));
% Define the objective function to mainimize Magnitude_tau
objectiveFunction_Min = @(theta) double(norm(subs(magnitude_tau, [theta1, theta2,
theta3], double(theta))));

% Define initial guess for thetas
x0 = [pi/4, pi/4, pi/4];

% Define bounds on thetas
lb = [0, 0, 0];
ub = [pi, pi, pi];

% Set up the optimization options
options = optimoptions('fmincon', 'Display', 'iter'); % Display optimization process

% Solve the optimization problem to find maximum magnitude_tau
[xMax, fMax] = fmincon(objectiveFunction_Max, x0, [], [], [], [], lb, ub, [],
options);
```

```
                                            First-order      Norm of
 Iter F-count            f(x)  Feasibility   optimality         step
    0       4   -3.919689e+01    0.000e+00    9.966e+00
    1       8   -5.609177e+01    0.000e+00    1.272e+01    1.060e+00
    2      12   -5.608346e+01    0.000e+00    9.967e-02    6.658e-02
    3      16   -5.611481e+01    0.000e+00    2.462e-01    1.043e-01
    4      20   -5.612178e+01    0.000e+00    3.808e-02    4.584e-02
    5      24   -5.612351e+01    0.000e+00    3.991e-02    9.528e-03
    6      28   -5.612379e+01    0.000e+00    8.705e-03    8.547e-03
    7      32   -5.612387e+01    0.000e+00    2.934e-03    8.270e-03
    8      36   -5.612392e+01    0.000e+00    7.059e-03    2.686e-02
    9      40   -5.612394e+01    0.000e+00    8.432e-03    1.066e-01
   10      44   -5.612391e+01    0.000e+00    5.702e-03    2.502e-01
   11      48   -5.612387e+01    0.000e+00    2.014e-03    3.047e-01
   12      52   -5.612386e+01    0.000e+00    1.000e-03    1.057e-01
   13      56   -5.612436e+01    0.000e+00    3.966e-02    3.296e-02
   14      60   -5.612466e+01    0.000e+00    4.157e-03    2.772e-02
   15      64   -5.612466e+01    0.000e+00    2.304e-03    9.228e-03
   16      68   -5.612466e+01    0.000e+00    2.000e-04    1.521e-03
   17      72   -5.612479e+01    0.000e+00    7.200e-05    1.730e-03
   18      76   -5.612482e+01    0.000e+00    4.355e-05    8.456e-04
   19      80   -5.612485e+01    0.000e+00    1.109e-05    8.913e-04
   20      84   -5.612486e+01    0.000e+00    2.976e-06    4.371e-04
   21      88   -5.612486e+01    0.000e+00    9.574e-07    2.019e-04

Local minimum found that satisfies the constraints.

Optimization completed because the objective function is non-decreasing in
feasible directions, to within the value of the optimality tolerance,
```

```
and constraints are satisfied to within the value of the constraint tolerance.

<stopping criteria details>
```

```
% Display the results (Maximum)
fprintf('Maximum Magnitude_tau: %f\n', abs(-fMax)); % Negate the value back to the original form
```

```
Maximum Magnitude_tau: 56.124860
```

```
fprintf('Optimal values for theta1, theta2, and theta3: %f, %f, %f\n', rad2deg(xMax(1)), rad2deg(xMax(2)), rad2deg(xMax(3)));
```

```
Optimal values for theta1, theta2, and theta3: 89.992386, 0.012731, 0.007796
```

```
% Solve the optimization problem to find minimum magnitude_tau
[xMin, fMin] = fmincon(objectiveFunction_Min, x0, [], [], [], [], lb, ub, [], options);
```

```
                                            First-order      Norm of
 Iter F-count            f(x)  Feasibility   optimality         step
    0       4    3.919689e+01    0.000e+00    2.214e+01
    1       8    1.626950e+01    0.000e+00    6.725e+00    2.648e+00
    2      13    1.531488e+01    0.000e+00    2.331e+00    6.207e-01
    3      17    1.500300e+01    0.000e+00    4.903e-01    2.868e-01
    4      21    1.501394e+01    0.000e+00    4.960e-01    2.316e-01
    5      25    1.501695e+01    0.000e+00    9.723e-02    9.207e-02
    6      29    1.500593e+01    0.000e+00    1.378e-01    1.109e-01
    7      33    1.500462e+01    0.000e+00    2.021e-02    8.398e-03
    8      37    1.500207e+01    0.000e+00    2.610e-02    5.367e-02
    9      41    1.500075e+01    0.000e+00    2.970e-02    5.741e-02
   10      45    1.500030e+01    0.000e+00    1.618e-02    3.999e-02
   11      49    1.500013e+01    0.000e+00    7.487e-03    2.953e-02
   12      53    1.500007e+01    0.000e+00    2.871e-03    1.762e-02
   13      57    1.500005e+01    0.000e+00    5.473e-04    7.422e-03
   14      61    1.500005e+01    0.000e+00    2.000e-04    1.676e-03
   15      65    1.500002e+01    0.000e+00    2.684e-03    1.976e-02
   16      69    1.500001e+01    0.000e+00    4.277e-04    8.775e-03
   17      73    1.500001e+01    0.000e+00    2.442e-04    3.851e-03
   18      77    1.500001e+01    0.000e+00    4.907e-05    6.269e-04
   19      81    1.500001e+01    0.000e+00    4.000e-05    8.088e-05
   20      85    1.500000e+01    0.000e+00    1.377e-04    1.197e-02
   21      89    1.500000e+01    0.000e+00    5.174e-04    7.289e-03
   22      93    1.500000e+01    0.000e+00    9.668e-05    2.785e-03
   23      97    1.500000e+01    0.000e+00    8.002e-06    5.607e-04
   24     101    1.500000e+01    0.000e+00    8.000e-06    9.221e-05
   25     105    1.500000e+01    0.000e+00    5.457e-04    1.075e-02
   26     109    1.500000e+01    0.000e+00    2.207e-04    5.955e-03
   27     113    1.500000e+01    0.000e+00    3.215e-05    2.612e-03
   28     117    1.500000e+01    0.000e+00    5.963e-06    5.719e-04
   29     121    1.500000e+01    0.000e+00    1.135e-05    1.139e-04
   30     125    1.500000e+01    0.000e+00    2.641e-05    5.544e-04

                                            First-order      Norm of
 Iter F-count            f(x)  Feasibility   optimality         step
   31     129    1.500000e+01    0.000e+00    8.033e-05    3.022e-03
   32     133    1.500000e+01    0.000e+00    2.478e-04    1.476e-02
   33     137    1.500000e+01    0.000e+00    4.596e-04    3.045e-02
   34     141    1.500000e+01    0.000e+00    8.102e-04    8.421e-02
```

```
    35      145     1.500000e+01     0.000e+00     1.127e-03     1.524e-01
    36      149     1.500000e+01     0.000e+00     9.945e-04     1.782e-01
    37      153     1.500000e+01     0.000e+00     2.145e-04     1.382e-01
    38      157     1.500000e+01     0.000e+00     4.519e-06     1.195e-02
    39      161     1.500000e+01     0.000e+00     1.825e-06     1.939e-03
    40      165     1.500000e+01     0.000e+00     1.600e-06     5.409e-04
    41      169     1.500000e+01     0.000e+00     2.055e-04     4.460e-02
    42      173     1.500000e+01     0.000e+00     1.798e-05     3.395e-02
    43      177     1.500000e+01     0.000e+00     1.452e-05     9.646e-03
    44      181     1.500000e+01     0.000e+00     5.795e-06     1.337e-04
    45      185     1.500000e+01     0.000e+00     3.196e-07     2.981e-05

Local minimum found that satisfies the constraints.

Optimization completed because the objective function is non-decreasing in
feasible directions, to within the value of the optimality tolerance,
and constraints are satisfied to within the value of the constraint tolerance.

<stopping criteria details>
```

```
% Display the results (Minimum)
fprintf('Minimum Magnitude_tau: %f\n', abs(-fMin)); % Negate the value back to the original form
```

```
Minimum Magnitude_tau: 15.000000
```

```
fprintf('Optimal values for theta1, theta2, and theta3: %f, %f, %f\n', rad2deg(xMin(1)), rad2deg(xMin(2)), rad2deg(xMin(3)));
```

```
Optimal values for theta1, theta2, and theta3: 90.001144, 90.817724, 179.176356
```

```
% PROBLEM 3: 3.1

clc;
clear;

syms L g m1 m2 theta1 theta2 theta1_dot theta2_dot theta1_dot_dot theta2_dot_dot
Ix1 Ix2 Ix3 Iy1 Iy2 Iy3 Iz1 Iz2 real

theta = [theta1; theta2];
thetadot = [theta1_dot; theta2_dot];
thetadotdot = [theta1_dot_dot; theta2_dot_dot];

% home matrix for center of mass m1
M1 = [roty(0),[0;L;-L/2]; 0 0 0 1];

% home matrix for center of mass m2
M2 = [eye(3), [0;L;-L]; 0 0 0 1];

S1 = [0;1;0;0;0;0];
S2 = [0;0;0;0;0;-1];

S_eq1 = [S1, [0;0;0;0;0;0]];
S_eq2 = [S1, S2];

% For center of mass m1
T_1 = fk(M1, S_eq1, theta)
```

T_1 =

$$\begin{pmatrix} \cos(\theta_1) & 0 & \sin(\theta_1) & -\frac{L\sin(\theta_1)}{2} \\ 0 & 1 & 0 & L \\ -\sin(\theta_1) & 0 & \cos(\theta_1) & -\frac{L\cos(\theta_1)}{2} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
R_1 = T_1(1:3, 1:3);
JS_1 = simplify(expand(JacS(S_eq1, theta))); % Space Jacobian
Jb_1 = adjointM(inv(T_1))*JS_1; % Body Jacobian
J_geometric_1 = simplify(expand([R_1, zeros(3); zeros(3), R_1] * Jb_1)); %
Geometric Jacobian
Jw1 = J_geometric_1(1:3,1:2)
```

Jw1 =

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

```
Jv1 = J_geometric_1(4:6, 1:2)
```

Jv1 =

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

```
Inertia_1 = [[Ix1 0 0]
    [0 Iy1 0]
    [0 0 Iz1]];



T_2 = fk(M2, S_eq2, theta)
```

T_2 =

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

```
R_2 = T_2(1:3, 1:3);
JS_2 = simplify(expand(JacS(S_eq2, theta))); % Space Jacobian
Jb_2 = adjointM(inv(T_2))*JS_2; % Body Jacobian
J_geometric_2 = simplify(expand([R_2, zeros(3); zeros(3), R_2] * Jb_2)); %
Geometric Jacobian
Jw2 = J_geometric_2(1:3,1:2)
```

Jw2 =

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

```
Jv2 = J_geometric_2(4:6, 1:2)
```

Jv2 =

$$\begin{pmatrix} -\cos(\theta_1)\,(L+\theta_2) & -\sin(\theta_1) \\ 0 & 0 \\ \sin(\theta_1)\,(L+\theta_2) & -\cos(\theta_1) \end{pmatrix}$$

```

Inertia_2 = [[Ix2 0 0]
    [0 Iy2 0]
    [0 0 Iz2]];

% Mass matrix evaluation
Mass_Matrix = simplify(expand(m1*(Jv1'*Jv1) + Jw1'*R_1*Inertia_1*R_1'*Jw1 +
m2*(Jv2'*Jv2) + Jw2'*R_2*Inertia_2*R_2'*Jw2))
```

```
Mass_Matrix =
```

$$\begin{pmatrix} \mathrm{Iy}_1 + \mathrm{Iy}_2 + \frac{L^2\, m_1}{4} + L^2\, m_2 + m_2\, \theta_2{}^2 + 2\, L\, m_2\, \theta_2 & 0 \\ 0 & m_2 \end{pmatrix}$$

```
% Coriolis matrix evaluation
Coriolis_Matrix = coriolis(Mass_Matrix, theta, thetadot)
```

```
Coriolis_Matrix =
```

$$\begin{pmatrix} \theta_{\dot{2}}\, (L\, m_2 + m_2\, \theta_2) & \theta_{\dot{1}}\, (L\, m_2 + m_2\, \theta_2) \\ -\theta_{\dot{1}}\, (L\, m_2 + m_2\, \theta_2) & 0 \end{pmatrix}$$

```
% Height evaluations as it is acting along the z-axis
h1 = T_1(3, 4);
h2 = T_2(3, 4);

% Potential energy evaluation
P = g*m1*h1 + g*m2*h2;

% Gravity vector evaluation
gravity_vector = simplify(expand([diff(P, theta1); diff(P, theta2)]))
```

```
gravity_vector =
```

$$\begin{pmatrix} \frac{g\, \sin(\theta_1)\, (L\, m_1 + 2\, L\, m_2 + 2\, m_2\, \theta_2)}{2} \\ -g\, m_2 \cos(\theta_1) \end{pmatrix}$$

```
% Final tau (joint torques) evaluation
tau = simplify(expand(Mass_Matrix*thetadotdot + Coriolis_Matrix*thetadot +
gravity_vector))
```

```
tau =
```

$$\begin{pmatrix} \mathrm{Iy}_1\, \theta_{\ddot{1}} + \mathrm{Iy}_2\, \theta_{\ddot{1}} + m_2\, \theta_2{}^2\, \theta_{\ddot{1}} + \frac{L^2\, m_1\, \theta_{\ddot{1}}}{4} + L^2\, m_2\, \theta_{\ddot{1}} + 2\, L\, m_2\, \theta_2\, \theta_{\ddot{1}} + 2\, L\, m_2\, \theta_{\dot{1}}\, \theta_{\dot{2}} + 2\, m_2\, \theta_2\, \theta_{\dot{1}}\, \theta_{\dot{2}} + \frac{L\, g\, m_1 \sin(}{2} \\ -m_2\, \left(L\, \theta_{\dot{1}}{}^2 - \theta_{\ddot{2}} + g\, \cos(\theta_1) + \theta_2\, \theta_{\dot{1}}{}^2\right) \end{pmatrix}$$

```
% PROBLEM 4: 4.1

clc;
clear;

syms L g m1 m2 m3 theta1 theta2 theta3 theta1_dot theta2_dot theta3_dot
theta1_dot_dot theta2_dot_dot theta3_dot_dot Ix1 Ix2 Ix3 Iy1 Iy2 Iy3 Iz1 Iz2 Iz3
real

theta = [theta1; theta2; theta3];
thetadot = [theta1_dot; theta2_dot; theta3_dot];
thetadotdot = [theta1_dot_dot; theta2_dot_dot; theta3_dot_dot];

% Home matrix till m1 center of mass
M1 = [eye(3),[L;0;0]; 0 0 0 1];
% Home matrix till m2 center of mass
M2 = [eye(3), [L;0;0]; 0 0 0 1];
% Home matrix till m3 center of mass
M3 = [eye(3), [2*L;0;0]; 0 0 0 1];

% Screw for joint 1
S1 = [0;0;0;1;0;0];
% Screw for joint 2
S2 = [0;0;0;0;1;0];
% Screw for joint 3
S3 = [0;0;1;0;0;0];

% Considering m1 center of mass as end-effector
S_eq1 = [S1, zeros(6, 1), zeros(6, 1)];
% Considering m2 center of mass as end-effector
S_eq2 = [S1, S2, zeros(6, 1)];
% Considering m3 center of mass as end-effector
S_eq3 = [S1, S2, S3];

% For center of mass m1
% Forward kinematics
T_1 = fk(M1, S_eq1, theta)
```

T_1 =

$$\begin{pmatrix} 1 & 0 & 0 & L+\theta_1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
R_1 = T_1(1:3, 1:3);
% Space Jacobian
Js_1 = simplify(expand(JacS(S_eq1, theta)));
% Body Jacobian
Jb_1 = adjointM(inv(T_1))*Js_1;
```

```
% Geometric Jacobian
J_geometric_1 = simplify(expand([R_1, zeros(3); zeros(3), R_1] * Jb_1));
% NOTE: For Jw(x1:y1, x2:y2) and Jv(x1:y1, x2:y2), number of columns y1 and
% y2 vary according to the number of joints, thus we change accordingly
Jw1 = J_geometric_1(1:3,1:3)
```

Jw1 =

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

```
Jv1 = J_geometric_1(4:6, 1:3)
```

Jv1 =

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

```
Inertia_1 = [[Ix1 0 0]
    [0 Iy1 0]
    [0 0 Iz1]];

% For m2 center of mass
T_2 = fk(M2, S_eq2, theta)
```

T_2 =

$$\begin{pmatrix} 1 & 0 & 0 & L+\theta_1 \\ 0 & 1 & 0 & \theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
R_2 = T_2(1:3, 1:3);
Js_2 = simplify(expand(JacS(S_eq2, theta))); % Space Jacobian
Jb_2 = adjointM(inv(T_2))*Js_2; % Body Jacobian
J_geometric_2 = simplify(expand([R_2, zeros(3); zeros(3), R_2] * Jb_2)); %
Geometric Jacobian
Jw2 = J_geometric_2(1:3,1:3)
```

Jw2 =

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

```
Jv2 = J_geometric_2(4:6, 1:3)
```

Jv2 =

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

```
Inertia_2 = [[Ix2 0 0]
    [0 Iy2 0]
    [0 0 Iz2]];

% For m3 center of mass
T_3 = fk(M3, S_eq3, theta)
```

T_3 =

$$\begin{pmatrix} \cos(\theta_3) & -\sin(\theta_3) & 0 & \theta_1 + 2\,L\cos(\theta_3) \\ \sin(\theta_3) & \cos(\theta_3) & 0 & \theta_2 + 2\,L\sin(\theta_3) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
R_3 = T_3(1:3, 1:3);
Js_3 = simplify(expand(JacS(S_eq3, theta))); % Space Jacobian
Jb_3 = adjointM(inv(T_3))*Js_3; % Body Jacobian
J_geometric_3 = simplify(expand([R_3, zeros(3); zeros(3), R_3] * Jb_3)); %
Geometric Jacobian
Jw3 = J_geometric_3(1:3,1:3)
```

Jw3 =

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

```
Jv3 = J_geometric_3(4:6, 1:3)
```

Jv3 =

$$\begin{pmatrix} 1 & 0 & -2\,L\sin(\theta_3) \\ 0 & 1 & 2\,L\cos(\theta_3) \\ 0 & 0 & 0 \end{pmatrix}$$

```
Inertia_3 = [[Ix3 0 0]
    [0 Iy3 0]
    [0 0 Iz3]];


% Mass Matrix
Mass_Matrix = simplify(expand(m1*(Jv1'*Jv1) + Jw1'*R_1*Inertia_1*R_1'*Jw1
+ m2*(Jv2'*Jv2) + Jw2'*R_2*Inertia_2*R_2'*Jw2 + m3*(Jv3'*Jv3) +
Jw3'*R_3*Inertia_3*R_3'*Jw3))
```

Mass_Matrix =

$$\begin{pmatrix} m_1 + m_2 + m_3 & 0 & -2\,L\,m_3\sin(\theta_3) \\ 0 & m_2 + m_3 & 2\,L\,m_3\cos(\theta_3) \\ -2\,L\,m_3\sin(\theta_3) & 2\,L\,m_3\cos(\theta_3) & 4\,m_3\,L^2 + \mathrm{Iz}_3 \end{pmatrix}$$

```
% Coriolis Matrix
Coriolis_Matrix = coriolis(Mass_Matrix, theta, thetadot)
```

Coriolis_Matrix =

$$\begin{pmatrix} 0 & 0 & -2\,L\,m_3\,\dot{\theta}_3\cos(\theta_3) \\ 0 & 0 & -2\,L\,m_3\,\dot{\theta}_3\sin(\theta_3) \\ 0 & 0 & 0 \end{pmatrix}$$

```
% Height of each center of mass
h1 = T_1(2, 4)
```

h1 = 0

```
h2 = T_2(2, 4)
```

h2 = θ_2

```
h3 = T_3(2, 4)
```

h3 = $\theta_2 + 2\,L\,\sin(\theta_3)$

```
% Potential Energy
P = g*m1*h1 + g*m2*h2 + g*m3*h3
```

P = $g\,m_3\,(\theta_2 + 2\,L\,\sin(\theta_3)) + g\,m_2\,\theta_2$

```
% Gravity vector
gravity_vector = simplify(expand([diff(P, theta(1)); diff(P, theta(2)); diff(P,
theta(3))]))
```

gravity_vector =

$$\begin{pmatrix} 0 \\ g\,(m_2 + m_3) \\ 2\,L\,g\,m_3\cos(\theta_3) \end{pmatrix}$$

```
% Tau calculation
tau = simplify(expand(Mass_Matrix*thetadotdot + Coriolis_Matrix*thetadot +
gravity_vector))
```

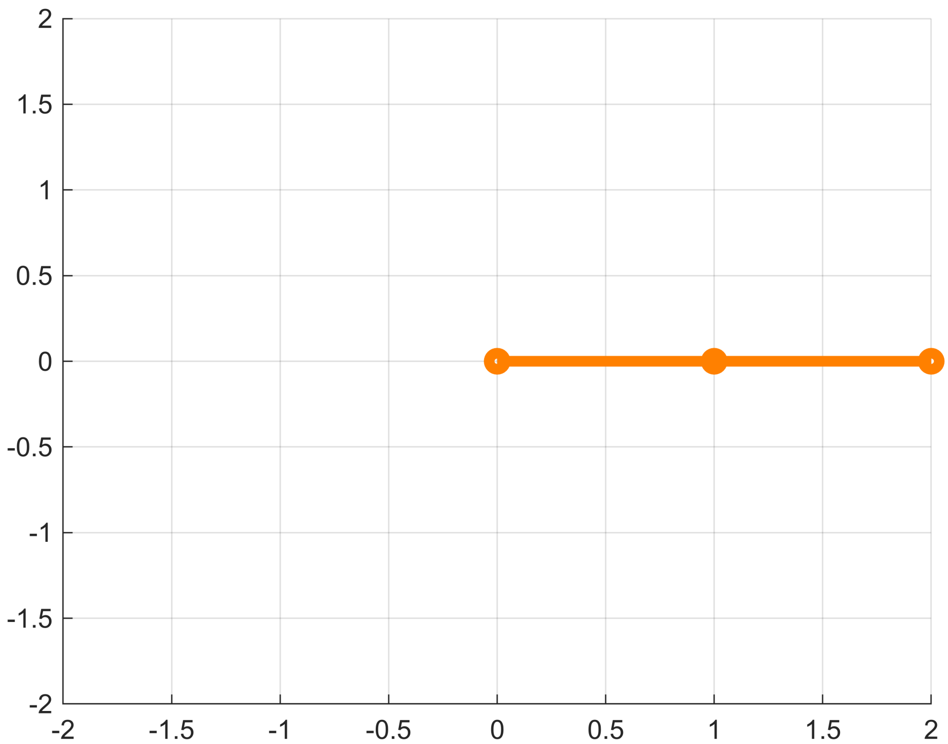
tau =

$$
\begin{pmatrix}
-2\,L\,m_3\cos(\theta_3)\,\dot{\theta}_3^{\,2}+m_1\,\ddot{\theta}_1+m_2\,\ddot{\theta}_1+m_3\,\ddot{\theta}_1-2\,L\,m_3\,\ddot{\theta}_3\sin(\theta_3)\\
-2\,L\,m_3\sin(\theta_3)\,\dot{\theta}_3^{\,2}+g\,m_2+g\,m_3+m_2\,\ddot{\theta}_2+m_3\,\ddot{\theta}_2+2\,L\,m_3\,\ddot{\theta}_3\cos(\theta_3)\\
\mathrm{Iz}_3\,\ddot{\theta}_3+4\,L^2\,m_3\,\ddot{\theta}_3+2\,L\,g\,m_3\cos(\theta_3)+2\,L\,m_3\,\ddot{\theta}_2\cos(\theta_3)-2\,L\,m_3\,\ddot{\theta}_1\sin(\theta_3)
\end{pmatrix}
$$

```
% Problem 5: 5.1
% CASE 1: ( tau = [0;0] and B=zeros(2))
% CASE 2: ( tau = [0;0] and B=I)
% CASE 3: ( tau = [20;5] and B=I)

close all
clear
clc


% create figure
figure
axis([-2, 2, -2, 2])
grid on
hold on

% save as a video file
v = VideoWriter('Problem5_1.mp4', 'MPEG-4');
v.FrameRate = 100;
open(v);

% pick your system parameters
L1 = 1;
L2 = 1;
m1 = 1;
m2 = 1;
I1 = 0.1;
I2 = 0.1;
g = 9.81;
tau = [0;0]; % Case 1 & 2
% tau = [20;5]; % Case 3

% Initial conditions
theta = [0;0]; % joint position
thetadot = [0;0]; % joint velocity
thetadotdot = [0;0]; % joint acceleration

masses = [m1,m2];
omega = [0;0;1];

Inertia_1 = [0 0 0;0 0 0;0 0 I1];
Inertia_2 = [0 0 0;0 0 0;0 0 I2];
q1 = [0;0;0]; % Position of Joint 1
q2 = [L1;0;0]; % Position of Joint 2
q3 = [L1+L2;0;0]; % end effector position

S1 = [omega; -cross(omega,q1)];
S2 = [omega;-cross(omega,q2)];
S_eq1 = [S1,[0;0;0;0;0;0]];
S_eq2 = [S1, S2];
```

```
M1 = [eye(3),q2; 0 0 0 1];
M2 = [eye(3), [L1+L2;0;0]; 0 0 0 1];

T_1 = fk(M1, S_eq1, theta);
R_1 = T_1(1:3, 1:3);
Js_1 = JacS(S_eq1, theta); % Space Jacobian
Jb_1 = (adjointM(inv(T_1))*Js_1); % Body Jacobian
J_geometric_1 = [R_1, zeros(3); zeros(3), R_1] * Jb_1; % Geometric Jacobian
Jw1 = J_geometric_1(1:3,:);
Jv1 = J_geometric_1(4:6, :);

T_2 = fk(M2, S_eq2, theta);
R_2 = T_2(1:3, 1:3);
Js_2 = JacS(S_eq2, theta); % Space Jacobian
Jb_2 = (adjointM(inv(T_2))*Js_2); % Body Jacobian
J_geometric_2 = [R_2, zeros(3); zeros(3), R_2] * Jb_2; % Geometric Jacobian
Jw2 = J_geometric_2(1:3,1:2);
Jv2 = J_geometric_2(4:6, 1:2);

gravity_vector = (zeros(length(theta),1));
Coriolis_Matrix = (zeros(2,2));
Mass_Matrix = [I1 + I2 + L1^2*m1 + L1^2*m2 + L2^2*m2 + 2*L1*L2*m2*cos(theta(2)),
m2*L2^2 + L1*m2*cos(theta(2))*L2 + I2; m2*L2^2 + L1*m2*cos(theta(2))*L2 + I2,
m2*L2^2 + I2];


for idx = 1:1000

    % plot the robot
    % 1. get the position of each link
    p0 = [0; 0];
    T1 = fk(M1,S_eq2(:,1:1),theta(1:1,:));
    p1 = T1(1:2,4); % position of link 1 (location of joint 2)
    T2 = fk(M2,S_eq2,theta);
    p2 = T2(1:2,4); % position of link 2 (the end-effector)
    P = [p0, p1, p2];
    % 2. draw the robot and save the frame
    cla;
    plot(P(1,:), P(2,:), 'o-', 'color',[1, 0.5, 0],'linewidth',4);
    drawnow
    frame = getframe(gcf);
    writeVideo(v,frame);

    % integrate to update velocity and position
    % your code here
    deltaT = 0.01;
    thetadot = thetadot + deltaT * thetadotdot;
    theta = theta + deltaT * thetadot;
```

```
    T_1 = fk(M1, S_eq1, theta);
    R_1 = T_1(1:3, 1:3);
    Js_1 = JacS(S_eq1, theta); % Space Jacobian
    Jb_1 = (adjointM(inv(T_1))*Js_1); % Body Jacobian
    J_geometric_1 = [R_1, zeros(3); zeros(3), R_1] * Jb_1; % Geometric Jacobian
    Jw1 = J_geometric_1(1:3,:);
    Jv1 = J_geometric_1(4:6, :);

    T_2 = fk(M2, S_eq2, theta);
    R_2 = T_2(1:3, 1:3);
    Js_2 = JacS(S_eq2, theta); % Space Jacobian
    Jb_2 = (adjointM(inv(T_2))*Js_2); % Body Jacobian
    J_geometric_2 = [R_2, zeros(3); zeros(3), R_2] * Jb_2; % Geometric Jacobian
    Jw2 = J_geometric_2(1:3,1:2);
    Jv2 = J_geometric_2(4:6, 1:2);

    Mass_Matrix =[ I1 + I2 + L1^2*m1 + L1^2*m2 + L2^2*m2 +
2*L1*L2*m2*cos(theta(2)), m2*L2^2 + L1*m2*cos(theta(2))*L2 + I2; m2*L2^2 +
L1*m2*cos(theta(2))*L2 + I2, m2*L2^2 + I2];

    Coriolis_Matrix = [-
L1*L2*m2*thetadot(2)*sin(theta(2)),  -L1*L2*m2*sin(theta(2))*(thetadot(1) +
thetadot(2));L1*L2*m2*thetadot(1)*sin(theta(2)), 0] ;

    gravity_vector = [(g*(m1+m2)*L1*cos(theta(1))) + g*m2*L2*cos(theta(1) +
theta(2)); g*m2*L2*cos(theta(1) + theta(2))];

    B = [[0 0]
        [0 0]]; % Case 1
    % B = eye(2); % Case 2 and 3

    thetadotdot = (inv(Mass_Matrix)) * (tau - Coriolis_Matrix * thetadot
-B*thetadot - gravity_vector);

    tau = Mass_Matrix * thetadotdot + Coriolis_Matrix * thetadot + B * thetadot +
gravity_vector;

end
```




```
Warning: The video's width and height has been padded to be a multiple of two as required by the H.264 codec.
```

```
close(v);
close all
```

```
function cmatrix = coriolis(m, theta, thetadot)

    n = length(theta); % Depends upon the no. of joints
    cmatrix = sym(zeros(size(m))); % Pre-allocating and initializing the matrix


    for k = (1:(size(cmatrix,1)))
        sum = 0; % Initializing the coriolis
        for j = (1:(size(cmatrix,2)))
            for i = (1:n)
                sum = sum + 1/2*(gradient(m(k,j),theta(i)) +
gradient(m(k,i),theta(j)) - gradient(m(i,j),theta(k)))*thetadot(i);
            end
            cmatrix(k, j) = sum;
            sum = 0;
        end
    end
end
```