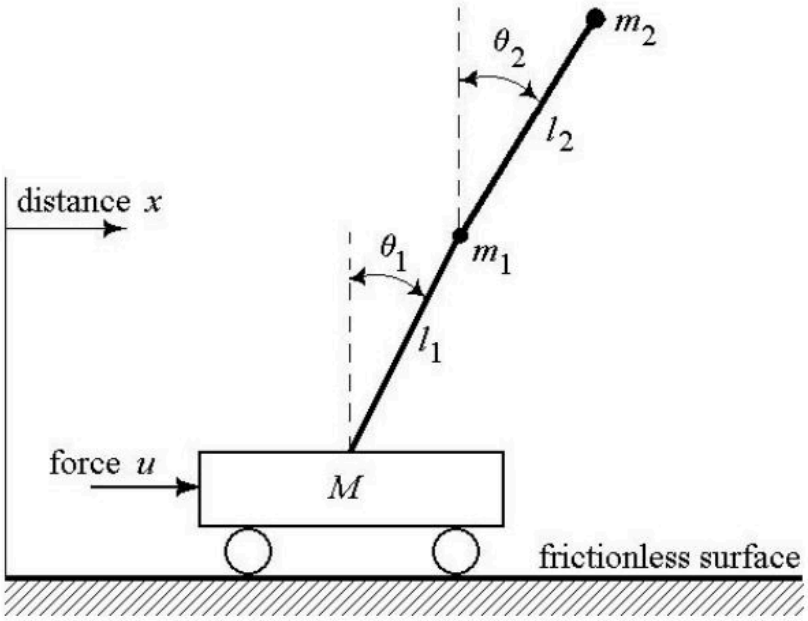
FUNWORK 1

Problem 1. (8 pts) Construct the Lagrangian for the double inverted pendulum on a cart (DIPC) shown in the figure below.



Basic Equations and Lagrange formulation

State Vector $X = [\,x_1\ x_2\ x_3\ x_4\ x_5\ x_6]$,

Where $x = x_1; \theta_1 = x_2; \theta_2 = x_3; \frac{\mathrm{d}}{\mathrm{d}t}x = x_4; \frac{\mathrm{d}}{\mathrm{d}t}\theta_1 = x_5;\ \frac{\mathrm{d}}{\mathrm{d}t}\theta_2 = x_6;$

Input force = u;

We are given the values of Masses for pendulums and the cart, as well as the length of strings.

Our next step is to write the kinetic as well as potential energies of the system.

```
syms x1 x2 x3 x4 x5 x6 u
syms M m1 m2 l1 l2 g
M = 1.5
```

```
M =
1.5000
```

```
m1 = 0.5
```

```
m1 =
0.5000
```

```
l1 = 0.5
```

```
l1 =
0.5000
```

```
m2 = 0.75
```

```
m2 =
0.7500
```

```
l2 = 0.75
```

```
l2 =
0.7500
```

```
g = 9.81
```

```
g =
9.8100
```

Defining the kinetic and potential energy of the system

Kinetic Energy of the Cart ($\frac{1}{2} M x_4^2$)

```
T_cart = 0.5*M*x4^2
```

```
T_cart =
```

$$\frac{3\,{x_4}^2}{4}$$

Kinetic Energy of Pendulum 1 ($\frac{1}{2} m_1 \left((x_4 + l_1 x_5 \cos(x_2))^2 + (l_1 x_5 \sin(x_2))^2 \right)$

```
T_pendulum1 = 0.5*m1*((x4+l1*x5*cos(x2))^2 + (l1*x5*sin(x2))^2)
```

```
T_pendulum1 =
```

$$\frac{{x_5}^2 \sin(x_2)^2}{16} + \frac{\left(x_4 + \frac{x_5 \cos(x_2)}{2}\right)^2}{4}$$

Kinetic Energy of Pendulum 2

$$\left(\frac{1}{2} m_2 \left((x_4 + l_1 x_5 \cos(x_2) + l_2 x_6 \cos(x_3))^2 + (l_1 x_5 \sin(x_2) + l_2 x_6 \sin(x_3))^2 \right)\right)$$

```
T_pendulum2 = 0.5*m2*((x4 + l1*x5*cos(x2) + l2*x6*cos(x3))^2 + (l1*x5*sin(x2) +
l2*x6*sin(x3))^2)
```

```
T_pendulum2 =
```

$$\frac{3\left(\frac{x_5 \sin(x_2)}{2} + \frac{3\,x_6 \sin(x_3)}{4}\right)^2}{8} + \frac{3\left(x_4 + \frac{x_5 \cos(x_2)}{2} + \frac{3\,x_6 \cos(x_3)}{4}\right)^2}{8}$$

Total Kinetic Energy of the System

```
T_total = simplify(T_cart + T_pendulum1 + T_pendulum2)
```

```
T_total =
```

$$\frac{11\,x_4^2}{8}+\frac{5\cos(x_2)\,x_4\,x_5}{8}+\frac{9\cos(x_3)\,x_4\,x_6}{16}+\frac{5\,x_5^2}{32}+\frac{9\cos(x_2-x_3)\,x_5\,x_6}{32}+\frac{27\,x_6^2}{128}$$

Potential Energy of the cart

0 as it is our reference point

```
V_Cart = 0
```

```
V_Cart =
0
```

Potential Energy of Pendulum 1

```
V_pendulum1 = m1*g*l1*cos(x2)
```

```
V_pendulum1 =
```

$$\frac{981\cos(x_2)}{400}$$

Potential Energy of Pendulum 2

```
V_pendulum2 = m2*g*(l1*cos(x2)+l2*cos(x3))
```

```
V_pendulum2 =
```

$$\frac{2943\cos(x_2)}{800}+\frac{8829\cos(x_3)}{1600}$$

Total Potential Energy

```
V_total = simplify(V_Cart + V_pendulum1 + V_pendulum2) % Total potential energy of
the system
```

```
V_total =
```

$$\frac{981\cos(x_2)}{160}+\frac{8829\cos(x_3)}{1600}$$

Lagrangian and Lagrange Equations for a system

We define the lagrangian for a system as $L = T - V$ where V is the total potential energy for the system and T is the total kinetic energy of the system.

We have our state vectors, $[x_1\ x_2\ x_3\ x_4\ x_5\ x_6]$ defined.

For calculating the lagrange equation we have our formula:

Intiutively it means that Rate of Change of Momentum = driving/restoring Forces

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

Where $q_i = [\, x_1 \; x_2 \; x_3 \,]$; $\dot{q}_i = [x_4 \; x_5 \; x_6]$; and $\dot{Q}_i$ = Sum of all input forces that are applied to the system(u for our system).

```
L = simplify(T_total - V_total)
```

L =

$$\frac{11\,x_4^2}{8} + \frac{5\cos(x_2)\,x_4\,x_5}{8} + \frac{9\cos(x_3)\,x_4\,x_6}{16} + \frac{5\,x_5^2}{32} + \frac{9\cos(x_2 - x_3)\,x_5\,x_6}{32} + \frac{27\,x_6^2}{128} - \frac{981\cos(x_2)}{160} - \frac{8829\cos(x_3)}{1600}$$

```
% Taking derivative of the values
dL_dx1 = diff(L, x1)
```

dL_dx1 = 0

```
dL_dx2 = diff(L, x2)
```

dL_dx2 =

$$\frac{981\sin(x_2)}{160} - \frac{5\,x_4\,x_5\sin(x_2)}{8} - \frac{9\,x_5\,x_6\sin(x_2 - x_3)}{32}$$

```
dL_dx3 = diff(L, x3)
```

dL_dx3 =

$$\frac{8829\sin(x_3)}{1600} - \frac{9\,x_4\,x_6\sin(x_3)}{16} + \frac{9\,x_5\,x_6\sin(x_2 - x_3)}{32}$$

```
dL_dx4 = diff(L, x4)
```

dL_dx4 =

$$\frac{11\,x_4}{4} + \frac{5\,x_5\cos(x_2)}{8} + \frac{9\,x_6\cos(x_3)}{16}$$

```
dL_dx5 = diff(L, x5)
```

dL_dx5 =

$$\frac{5\,x_5}{16} + \frac{5\,x_4\cos(x_2)}{8} + \frac{9\,x_6\cos(x_2 - x_3)}{32}$$

```
dL_dx6 = diff(L, x6)
```

dL_dx6 =

$$\frac{27\,x_6}{64} + \frac{9\,x_4\cos(x_3)}{16} + \frac{9\,x_5\cos(x_2 - x_3)}{32}$$

Calculating Jacobian for simplicity

We can't directly calculate the time derivative of $\frac{\partial L}{\partial \dot{q}_i}$ in MATLAB so instead we use the chain rule for it:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}}\right) = \frac{\partial}{\partial q}\left(\frac{\partial L}{\partial \dot{q}}\right)\dot{q} + \frac{\partial}{\partial \dot{q}}\left(\frac{\partial L}{\partial \dot{q}}\right)\ddot{q}$$

Here $\ddot{q}$ is the derivative of $\dot{q}_i \ = [x_4 \ x_5 \ x_6]$

We don't know the value for it either, so we take new variables and define it and we will later get this value in terms of u.

Let's have a look on what is going on and how are we going to calculate the funtion further.

```
q = [x1 ;x2; x3];
qd = [x4; x5; x6];
syms qdd1 qdd2 qdd3 real
qdd = [qdd1; qdd2; qdd3];
```

Jaco

So Jaco is the matrix of partial derivative of $\frac{\partial L}{\partial \dot{q}_i}$ with respect to q_i and $\dot{q}_i$, basically it is a matrix containing values of $\frac{\partial}{\partial q}\left(\frac{\partial L}{\partial \dot{q}}\right) and \frac{\partial}{\partial \dot{q}}\left(\frac{\partial L}{\partial \dot{q}}\right)$. This helps us in applying chain rule easily.

```
Jaco = jacobian([dL_dx4;dL_dx5;dL_dx6],[q;qd])
```

```
Jaco =
```

$$\begin{pmatrix} 0 & -\frac{5\,x_5\sin(x_2)}{8} & -\frac{9\,x_6\sin(x_3)}{16} & \frac{11}{4} & \sigma_5 & \sigma_4 \\ 0 & -\frac{5\,x_4\sin(x_2)}{8}-\sigma_1 & \sigma_1 & \sigma_5 & \frac{5}{16} & \sigma_3 \\ 0 & -\sigma_2 & \sigma_2-\frac{9\,x_4\sin(x_3)}{16} & \sigma_4 & \sigma_3 & \frac{27}{64} \end{pmatrix}$$

where

$$\sigma_1 = \frac{9\,x_6\sin(x_2-x_3)}{32}$$

$$\sigma_2 = \frac{9\,x_5\sin(x_2-x_3)}{32}$$

$$\sigma_3 = \frac{9\cos(x_2-x_3)}{32}$$

$$\sigma_4 = \frac{9\cos(x_3)}{16}$$

$$\sigma_5 = \frac{5\cos(x_2)}{8}$$

Completion of Chain Rule

Multiplying the above matrix jaco with $\dot{q}_i$ and $\ddot{q}_i$ gives us the time derivative of $\frac{\partial L}{\partial \dot{q}_i}$.

After that we go ahead and calculate our euler lagrange equation, which is $\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i}$; where i =1,2,3(for all states)

```
d_dt_dLdqdot = Jaco*[qd; qdd]
```

```
d_dt_dLdqdot =
```

$$\begin{pmatrix} -\frac{5\sin(x_2)\,x_5^2}{8}-\frac{9\sin(x_3)\,x_6^2}{16}+\frac{11\,\mathrm{qdd}_1}{4}+\frac{5\,\mathrm{qdd}_2\cos(x_2)}{8}+\frac{9\,\mathrm{qdd}_3\cos(x_3)}{16} \\ \frac{5\,\mathrm{qdd}_2}{16}+\frac{9\,x_6^2\sin(x_2-x_3)}{32}+\frac{5\,\mathrm{qdd}_1\cos(x_2)}{8}+\frac{9\,\mathrm{qdd}_3\cos(x_2-x_3)}{32}-x_5\left(\frac{5\,x_4\sin(x_2)}{8}+\frac{9\,x_6\sin(x_2-x_3)}{32}\right) \\ \frac{27\,\mathrm{qdd}_3}{64}-\frac{9\,x_5^2\sin(x_2-x_3)}{32}+\frac{9\,\mathrm{qdd}_1\cos(x_3)}{16}+\frac{9\,\mathrm{qdd}_2\cos(x_2-x_3)}{32}-x_6\left(\frac{9\,x_4\sin(x_3)}{16}-\frac{9\,x_5\sin(x_2-x_3)}{32}\right) \end{pmatrix}$$

```
dLdq = [dL_dx1;dL_dx2;dL_dx3];
```

Solving Lagrange equation gives us the initial imput force, we can take that vector to the left side to simplify the equation and make it equal to 0 to solve it.

Input force = u, in x direction, no other external force is applied.

```
Force = [u; 0; 0];
Lagrange_eq = simplify(d_dt_dLdqdot - dLdq -Force)
```

Lagrange_eq =

$$\begin{pmatrix} -\frac{5\sin(x_2)\,x_5^2}{8} - \frac{9\sin(x_3)\,x_6^2}{16} + \frac{11\,\mathrm{qdd}_1}{4} - u + \frac{5\,\mathrm{qdd}_2\cos(x_2)}{8} + \frac{9\,\mathrm{qdd}_3\cos(x_3)}{16} \\ \frac{9\sin(x_2-x_3)\,x_6^2}{32} + \frac{5\,\mathrm{qdd}_2}{16} - \frac{981\sin(x_2)}{160} + \frac{5\,\mathrm{qdd}_1\cos(x_2)}{8} + \frac{9\,\mathrm{qdd}_3\cos(x_2-x_3)}{32} \\ -\frac{9\sin(x_2-x_3)\,x_5^2}{32} + \frac{27\,\mathrm{qdd}_3}{64} - \frac{8829\sin(x_3)}{1600} + \frac{9\,\mathrm{qdd}_1\cos(x_3)}{16} + \frac{9\,\mathrm{qdd}_2\cos(x_2-x_3)}{32} \end{pmatrix}$$

Simplifying the Equation

Now from lagrangian we do have the three equations but they contain terms of $\ddot{q}$ in them. So, now we take use of the equation:

$D(q)\,\ddot{q} + C(q,\dot{q})\,\dot{q} + G(q) = H\,u$ (Taken inspiration from Alexander Bogdanov)

Lagrange equations for the DIPC system can be written in a more compact matrix form:

$$\mathbf{D}(\theta)\ddot{\theta} + \mathbf{C}(\theta,\dot{\theta})\dot{\theta} + \mathbf{G}(\theta) = \mathbf{H}u \tag{2}$$

where

$$\mathbf{D}(\theta) = \begin{pmatrix} d_1 & d_2\cos\theta_1 & d_3\cos\theta_2 \\ d_2\cos\theta_1 & d_4 & d_5\cos(\theta_1-\theta_2) \\ d_3\cos\theta_2 & d_5\cos(\theta_1-\theta_2) & d_6 \end{pmatrix} \tag{3}$$

$$\mathbf{C}(\theta,\dot{\theta}) = \begin{pmatrix} 0 & -d_2\sin(\theta_1)\dot{\theta}_1 & -d_3\sin(\theta_2)\dot{\theta}_2 \\ 0 & 0 & d_5\sin(\theta_1-\theta_2)\dot{\theta}_2 \\ 0 & -d_5\sin(\theta_1-\theta_2)\dot{\theta}_1 & 0 \end{pmatrix} \tag{4}$$

$$\mathbf{G}(\theta) = \begin{pmatrix} 0 \\ -f_1\sin\theta_1 \\ -f_2\sin\theta_2 \end{pmatrix} \tag{5}$$

$$\mathbf{H} = (1\ 0\ 0)^T$$

This equation is *result* of applying the Euler-Lagrange method.

Explaination:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = Q_i, \quad i = 1, \ldots, n$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}}\right) = \frac{\partial}{\partial q}\left(\frac{\partial L}{\partial \dot{q}}\right)\dot{q} + \frac{\partial}{\partial \dot{q}}\left(\frac{\partial L}{\partial \dot{q}}\right)\ddot{q}$$

We define quadratic Kinetic Energy as:

$T = \frac{1}{2}\dot{q}^T D(q)\,\dot{q}$, Therefore, this gives

$D_{ij}(q) = \frac{\partial^2 T}{\partial \dot{q}_i\,\partial \dot{q}_j}$; This matrix represents the inertial properties of the system. It's a symmetric, positive-definite matrix that relates the generalized accelerations to the forces required to produce them

The entries of the Coriolis matrix C are derived from the Christoffel symbols of the first kind. These symbols are a fundamental concept in differential geometry and are used here to mathematically represent the Coriolis and centrifugal effects. As the formula shows, each entry of C is a linear combination of velocities, and its coefficients are calculated from the partial derivatives of the mass-inertia matrix D. Each entry of C is a **linear combination of velocities.**

This derivation ensures that all kinetic energy effects are accurately captured.

C is our Coriolis and Centrifugal terms containing matrix and is given from the Christoffel Symbols:

$$C_{ij}(q,\dot{q}) = \sum_{k=1}^{n} c_{ijk}(q)\,\dot{q}_k$$

Coefficients come from the geometry-dependent inertia matrix $D(q)$

$$c_{ijk}(q) = \frac{1}{2}\left(\frac{\partial D_{ij}}{\partial q_k} + \frac{\partial D_{ik}}{\partial q_j} - \frac{\partial D_{jk}}{\partial q_i}\right)$$

G: This term represents the forces due to gravity acting on the system's components. For the DIPC, these forces are dependent on the pendulum angles (θ1 and θ2), but not the cart's position.

$$G(q) = \frac{\partial V}{\partial q}$$

H is just the input force mapping to the generalised coordinates, this is the external force or torque applied to the system. In this case, u is the control input (e.g., a force applied to the cart), and the matrix H maps this scalar input to the appropriate component of the generalized force vector

$$Q = H\,u$$

For our case H is just [1 0 0] because we have only one input force in x direction and coordinates are pre-defined and standard

```
D = sym(zeros(3,3));
for i=1:3
  for j=1:3
    D(i,j) = simplify(diff(diff(T_total, qd(i)), qd(j)));% Computing D(q)
  end

end
D
```

D =

$$\begin{pmatrix} \frac{11}{4} & \frac{5\cos(x_2)}{8} & \frac{9\cos(x_3)}{16} \\ \frac{5\cos(x_2)}{8} & \frac{5}{16} & \frac{9\cos(x_2 - x_3)}{32} \\ \frac{9\cos(x_3)}{16} & \frac{9\cos(x_2 - x_3)}{32} & \frac{27}{64} \end{pmatrix}$$

```
G = simplify([ diff(V_total, x1);
      diff(V_total, x2);
      diff(V_total, x3) ])
```

G =

$$\begin{pmatrix} 0 \\ -\frac{981\sin(x_2)}{160} \\ -\frac{8829\sin(x_3)}{1600} \end{pmatrix}$$

```
n = 3;
c = sym(zeros(n,n,n));
for i=1:n
  for j=1:n
    for k=1:n
      c(i,j,k) = 1/2*( diff(D(i,j), q(k)) + diff(D(i,k), q(j)) - diff(D(j,k),
q(i)) );
    end
  end
end

C = sym(zeros(n,n));
for i=1:n
  for j=1:n
    % reshape c(i,j,:) into a column vector (3x1) and dot with qd (3x1)
    C(i,j) = simplify( reshape(c(i,j,:),[n,1]).' * qd );
  end
end

C
```

C =

$$\begin{pmatrix} 0 & -\frac{5\,x_5\sin(x_2)}{8} & -\frac{9\,x_6\sin(x_3)}{16} \\ 0 & 0 & \frac{9\,x_6\sin(x_2-x_3)}{32} \\ 0 & -\frac{9\,x_5\sin(x_2-x_3)}{32} & 0 \end{pmatrix}$$

Getting the values of $\ddot{q}$

We take the organised equation and get the values of double derivatives in terms of input force and derivatives.

```
H = [1;0;0];
qdd = simplify(D\(H*u - C*qd-G));
X = [q;qd];
f_sym = [qd; qdd]
```

f_sym =

$$\begin{pmatrix} x_4 \\ x_5 \\ x_6 \\ -\frac{280\,u - 981\sin(2\,x_2) + 100\,{x_5}^2\sin(x_2) + 45\,{x_6}^2\sin(x_3) - 120\,u\cos(\ \\ 20\,\sigma_2 \\ \frac{1575\,{x_6}^2\sin(x_2-x_3) - 30411\sin(x_2) - 8829\sin(x_2-2\,x_3) + 250\,{x_5}^2\sin(2\,x_2) + 1400\,u\cos(x_2) + 450\,{x_5}^2\ s\ }{50\,\sigma_2} \\ -\frac{300\sin(x_2-x_3)\,{x_5}^2 + 135\sin(\sigma_3)\,{x_6}^2 - 2943\,\sigma_1 + 2943\sin(x_3) + 200\,u\cos(2\,x\ }{75\cos(2\,x_2) + 135\cos(\sigma_3) - 390} \end{pmatrix}$$

where

$$\sigma_1 = \sin(2\,x_2 - x_3)$$

$$\sigma_2 = 5\cos(2\,x_2) + 9\cos(\sigma_3) - 26$$

$$\sigma_3 = 2\,x_2 - 2\,x_3$$

Taylor Linearization

$\dot{X} = f(X, u)$

$\dot{X} = \mathrm{Ax} + \mathrm{Bu}$; For a linear model

We have a non-linear model and we need to linearize it by taking equilibrium around 0.

$$f(X,u) \approx f(X_0,u_0) + \frac{\partial f}{\partial X}\Big|_{(X_0,u_0)} (X - X_0) + \frac{\partial f}{\partial u}\Big|_{(X_0,u_0)} (u - u_0)$$

$$A = \frac{\mathrm{d}}{\mathrm{d}X} f; B = \frac{\mathrm{d}}{\mathrm{d}u} f; \text{here } f = f_{\text{sym;}}$$

After getting the Jacobians for the following we can substitute the values to get the linearized model's matrices.

```
% Define the state-space representation
A_sym = simplify(jacobian(f_sym, X));
B_sym = simplify(jacobian(f_sym, u));
vars = [x1; x2; x3; x4; x5; x6; u];
vals = [0; 0; 0; 0; 0; 0; 0];

A_lin_syms = double(simplify(subs(A_sym, vars, vals)))
```

```
A_lin_syms = 6×6
         0         0         0    1.0000         0         0
         0         0         0         0    1.0000         0
         0         0         0         0         0    1.0000
         0   -8.1750         0         0         0         0
         0   65.4000  -29.4300         0         0         0
         0  -32.7000   32.7000         0         0         0
```

```
B_lin_syms = double(simplify(subs(B_sym, vars, vals)))
```

```
B_lin_syms = 6×1
         0
         0
         0
    0.6667
   -1.3333
         0
```

Checking Controllability

$$C(\text{Contro}) = [B\ \text{AB}\ A^2B\ A^3B. \ldots\ldots A^{(N-1)}B]$$

We check the rank of our controllability matrix and match it with the rank of A.

If they are equal, the system is fully controllable

```
Contro = ctrb(A_lin_syms, B_lin_syms)
```

```
Contro = 6×6
10^3 ×
         0    0.0007         0    0.0109         0    0.7129
         0   -0.0013         0   -0.0872         0   -6.9860
         0         0         0    0.0436         0    4.2772
    0.0007         0    0.0109         0    0.7129         0
   -0.0013         0   -0.0872         0   -6.9860         0
         0         0    0.0436         0    4.2772         0
```

```
rankContro = rank(Contro);

if rankContro == size(A_lin_syms,1)
```

```
    disp('Fully Controllable system');
else
    disp('Not fully controllable');
end
```

```
Fully Controllable system
```

Checking Observability

We have:

$\dot{X} = \mathrm{Ax} + \mathrm{Bu}$

$y = \mathrm{Cx} + \mathrm{Du}$

C is our output matrix, which must give us our first three state vectors because we need them as our output. Which are $q_i = [\, x_1 \; x_2 \; x_3 \,]$.

Hence we define C as follows; for D, it should be 0 because our output doesn't directly depend on the input function.

$O = [C \; \mathrm{CA} \; \mathrm{CA}^2 \; \mathrm{CA}^3 \ldots\ldots \mathrm{CA}^{(N-1)}]$

If the rank is same as A then it's fully observable

```
C_output = [1 0 0 0 0 0;
            0 1 0 0 0 0;
            0 0 1 0 0 0;];
D = zeros(3,1)
```

```
D = 3×1
     0
     0
     0
```

```
Obs = obsv(A_lin_syms, C_output);
rankObs = rank(Obs);
disp(['Observability rank = ', num2str(rankObs)]);
```

```
Observability rank = 6
```

```
if rankObs == size(A_lin_syms,1)
    disp('Fully Observable');
else
    disp('NOT fully observable');
end
```

```
Fully Observable
```

Transforming the model into Controller and Observer form

We can use the ss and compreal functions to calculate the controller and observable form.

```
C_row = C_output(1,:);

sys = ss(A_lin_syms, B_lin_syms, C_row,0);

sys_canon = compreal(sys,'c');

[A_controller, B_controller, C_controller, D_controller] = ssdata(sys_canon);

disp('Controller canonical A ='); disp(A_controller);
```

```
Controller canonical A =
   1.0e+03 *

         0         0         0         0         0   -0.0000
    0.0010         0         0         0         0         0
         0    0.0010         0         0         0   -1.1762
         0         0    0.0010         0         0         0
         0         0         0    0.0010         0    0.0981
         0         0         0         0    0.0010         0
```

```
disp('Controller canonical B ='); disp(B_controller);
```

```
Controller canonical B =
     1
     0
     0
     0
     0
     0
```

```
disp('Controller canonical C ='); disp(C_controller);
```

```
Controller canonical C =
         0    0.6667         0   10.9000         0  712.8600
```

```
Co = ctrb(A_lin_syms, B_lin_syms);
Co_canon = ctrb(A_controller, B_controller);

T_controller = Co * inv(Co_canon)
```

```
T_controller = 6×6
10^3 ×
         0    0.0007         0    0.0109         0    0.7129
         0   -0.0013         0   -0.0872         0   -6.9860
         0         0         0    0.0436         0    4.2772
    0.0007         0    0.0109         0    0.7129         0
   -0.0013         0   -0.0872         0   -6.9860         0
         0         0    0.0436         0    4.2772         0
```

```
sys_observable = compreal(sys,'o');

[A_obs, B_obs, C_obs, D_obs] = ssdata(sys_observable);
disp(A_obs);
```

```
   1.0e+03 *

         0    0.0010         0         0         0         0
```

```
         0         0    0.0010         0         0         0
         0         0         0    0.0010         0         0
         0         0         0         0    0.0010         0
         0         0         0         0         0    0.0010
         0         0   -1.1762         0    0.0981         0
```

```
disp(B_obs);
```

```
         0
    0.6667
         0
   10.9000
         0
  712.8600
```

```
disp(C_obs);
```

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

Transfer Function

The general form of the state-space model is:

$\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + B\mathbf{u}(t)$ (State Equation) $\mathbf{y}(t) = C\mathbf{x}(t) + D\mathbf{u}(t)$ (Output Equation)

Where:

x(t) is the state vector.

u(t) is the input vector.

y(t) is the output vector.

A,B,C,D are the system matrices.

$\mathbf{X}(s) = A\mathbf{X}(s) + B\mathbf{U}(s)$; After Laplace Transform

$\mathbf{Y}(s) = C\mathbf{X}(s) + D\mathbf{U}(s)$

$(sI - A)\mathbf{X}(s) = B\mathbf{U}(s)$

$Y(s) = C[(sI - A)^{-1}B\mathbf{U}(s)] + D\mathbf{U}(s)$

$Y(s) = [C(sI - A)^{-1}B + D]\mathbf{U}(s)$

Transfer function G(s) is given by:

$$G(S) = \frac{Y(S)}{U(S)} = [C(\mathrm{sI} - A)^{-1}B + D]$$

We use MATLAB's ss funtion to calculate our G matrix and each row gives us the value of transfer function with respect to x, θ_1, θ_2

```
sys = ss(A_lin_syms, B_lin_syms, C_output, D);
sys.InputName = 'u';
sys.OutputName = {'x', 'theta1', 'theta2'};
```

```
G = tf(sys);
```

The resulting G is a 3×1 matrix where each element is a separate transfer function:

```
G(1,1)
```

```
ans =

  From input "u" to output "x":
  0.6667 s^4 + 8.29e-15 s^3 - 54.5 s^2 - 1.516e-13 s + 427.7
  ------------------------------------------------------------
  s^6 - 2.665e-15 s^5 - 98.1 s^4 - 1.705e-13 s^3 + 1176 s^2

Continuous-time transfer function.
Model Properties
```

```
G(2,1)
```

```
ans =

  From input "u" to output "theta1":
            -1.333 s^2 - 1.184e-15 s + 43.6
  -------------------------------------------------
  s^4 + 1.021e-14 s^3 - 98.1 s^2 - 1.705e-13 s + 1176

Continuous-time transfer function.
Model Properties
```

```
G(3,1)
```

```
ans =

  From input "u" to output "theta2":
                         43.6
  -------------------------------------------------
  s^4 + 1.021e-14 s^3 - 98.1 s^2 - 1.705e-13 s + 1176

Continuous-time transfer function.
Model Properties
```

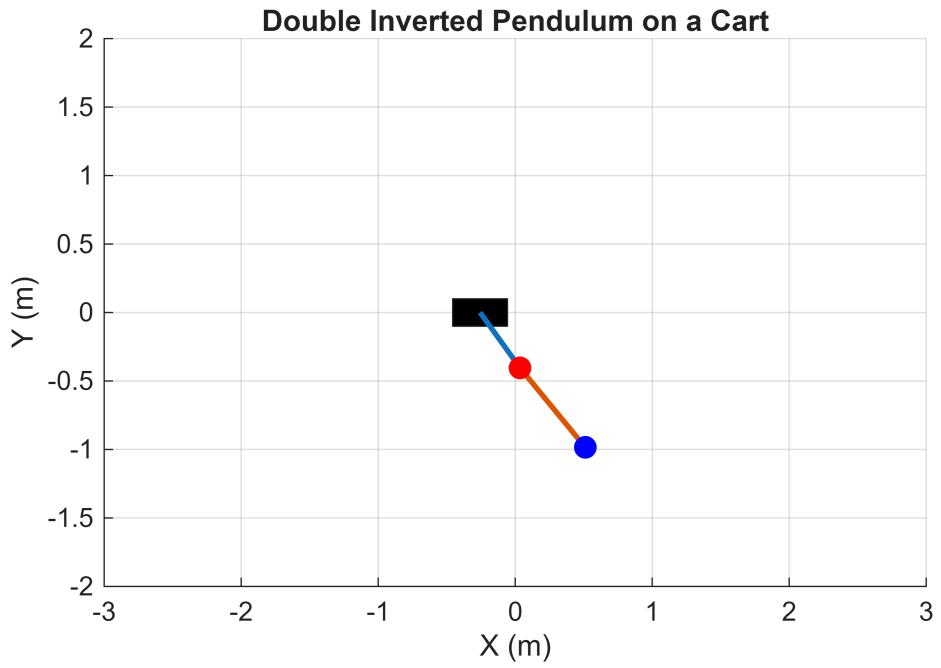
```

```

```
% Funwork1
% State Space
% q = [x1 x2 x3 x4 x5 x6]
% x1 = x
% x2 = theta_1
% x3 = theta_2
% x4 = x1_dot
% x5 = theta_1_dot
% x6 = theta_2_dot

% Define the generalized coordinates and velocities
syms x1 x2 x3 x4 x5 x6 u
syms M m1 m2 l1 l2 g real
M_val = 1.5;
m1_val = 0.5;
l1_val = 0.5;
m2_val = 0.75;
l2_val = 0.75;
g_val = 9.81;

% Define the kinetic and potential energy of the system
T_cart = 0.5*M*x4^2;
T_pendulum1 = 0.5*m1*((x4+l1*x5*cos(x2))^2 + (l1*x5*sin(x2))^2);
T_pendulum2 = 0.5*m2*((x4 + l1*x5*cos(x2) + l2*x6*cos(x3))^2 + (l1*x5*sin(x2) +
l2*x6*sin(x3))^2);
T_total = T_cart + T_pendulum1 + T_pendulum2;

% Define the potential energy of the system
V_Cart = 0;
V_pendulum1 = m1*g*l1*cos(x2);
V_pendulum2 = m2*g*(l1*cos(x2)+l2*cos(x3));
V_total = V_Cart + V_pendulum1 + V_pendulum2;
L = T_total - V_total;

q = [x1 ;x2; x3];
qd = [x4; x5; x6];

% Computing D(q)
D = sym(zeros(3,3));
for i=1:3
  for j=1:3
    D(i,j) = diff(diff(T_total, qd(i)), qd(j));
  end
end
D = simplify(D);

% Computing G(q)
G = [ diff(V_total, x1);
      diff(V_total, x2);
      diff(V_total, x3) ];
```

```
G = simplify(G);

% Computing C(q,qd)
n = 3;
c = sym(zeros(n,n,n));
for i=1:n
  for j=1:n
    for k=1:n
      c(i,j,k) = 1/2*( diff(D(i,j), q(k)) + diff(D(i,k), q(j)) - diff(D(j,k),
q(i)) );
    end
  end
end
C = sym(zeros(n,n));
for i=1:n
  for j=1:n
    C(i,j) = reshape(c(i,j,:),[n,1]).' * qd;
  end
end
C = simplify(C);

% The applied force
H = [1;0;0];

% Convert symbolic expressions to MATLAB functions for faster evaluation
matlabFunction(D, 'file', 'compute_D', 'vars', {x1, x2, x3, M, m1, m2, l1, l2});
matlabFunction(C, 'file', 'compute_C', 'vars', {x1, x2, x3, x4, x5, x6, m1, m2, l1,
l2});
matlabFunction(G, 'file', 'compute_G', 'vars', {x1, x2, x3, m1, m2, l1, l2, g});
function dxdt = double_inv_pendulum(t, X, params, u)
    q = X(1:3);
    qd = X(4:6);

    % Unpack parameters
    M_val = params.M;
    m1_val = params.m1;
    m2_val = params.m2;
    l1_val = params.l1;
    l2_val = params.l2;
    g_val = params.g;

    % Numerically compute D, C, and G using the generated functions
    D_mat = compute_D(q(1), q(2), q(3), M_val, m1_val, m2_val, l1_val, l2_val);
    C_mat = compute_C(q(1), q(2), q(3), qd(1), qd(2), qd(3), m1_val, m2_val,
l1_val, l2_val);
    G_vec = compute_G(q(1), q(2), q(3), m1_val, m2_val, l1_val, l2_val, g_val);

    H_vec = [1;0;0];

    % Compute the accelerations (qdd)
```

```
    qdd = D_mat \ (H_vec * u - C_mat * qd - G_vec);

    % Combine velocities and accelerations to get the state derivative
    dxdt = [qd; qdd];
end
% Set up simulation parameters and initial conditions
params.M = M_val;
params.m1 = m1_val;
params.m2 = m2_val;
params.l1 = l1_val;
params.l2 = l2_val;
params.g = g_val;

% Initial state: [x, theta1, theta2, x_dot, theta1_dot, theta2_dot]
x0 = [0; 0.01; 0.02; 0; 0; 0];
u = 0; % No external force

% Simulate the system
[t, X_sol] = ode45(@(t,x) double_inv_pendulum(t,x,params,u), [0 10], x0);

% Extract solution components for animation
x = X_sol(:, 1);
theta1 = X_sol(:, 2);
theta2 = X_sol(:, 3);

% Cart Position
xc = x;
yc = zeros(size(x));
% First pendulum
x1_p = xc + params.l1*sin(theta1);
y1_p = yc + params.l1*cos(theta1);
% Second pendulum
x2_p = x1_p + params.l2*sin(theta2);
y2_p = y1_p + params.l2*cos(theta2);

% Animation loop
figure;
axis equal;
axis([-3 3 -2 2]);
xlabel('X (m)');
ylabel('Y (m)');
title('Double Inverted Pendulum on a Cart');
grid on;
hold on;

hCart = rectangle('Position',[-0.2, -0.1, 0.4, 0.2],'FaceColor','k');
hLink1 = line([xc(1), x1_p(1)],[yc(1), y1_p(1)],'LineWidth',2);
hLink2 = line([x1_p(1), x2_p(1)],[y1_p(1), y2_p(1)],'LineWidth',2);
hMass1 = plot(x1_p(1),y1_p(1),'ro','MarkerSize',8,'MarkerFaceColor','r');
hMass2 = plot(x2_p(1),y2_p(1),'bo','MarkerSize',8,'MarkerFaceColor','b');
```

```
% Loop through the simulation results to animate
for k = 1:length(t)
    set(hCart,'Position',[xc(k)-0.2, -0.1, 0.4, 0.2]);
    set(hLink1,'XData',[xc(k), x1_p(k)],'YData',[yc(k), y1_p(k)]);
    set(hLink2,'XData',[x1_p(k), x2_p(k)],'YData',[y1_p(k), y2_p(k)]);
    set(hMass1,'XData',x1_p(k),'YData',y1_p(k));
    set(hMass2,'XData',x2_p(k),'YData',y2_p(k));
    drawnow;
end
```




```
writeAnimation('DIPC')
```

```
H = [1;0;0];
qdd = simplify(D\(H*u - C*qd-G));
X = [q;qd];
f_sym = [qd; qdd]
```

f_sym =

$$\begin{pmatrix} x_4 \\ x_5 \\ x_6 \\ -\dfrac{280\,u - 981\sin(2\,x_2) + 100\,x_5^2\sin(x_2) + 45\,x_6^2\sin(x_3) - 120\,u\cos(\sigma_3) + 45\,x_6^2\,\sigma_1}{20\,\sigma_2} \\ \dfrac{1575\,x_6^2\sin(x_2 - x_3) - 30411\sin(x_2) - 8829\sin(x_2 - 2\,x_3) + 250\,x_5^2\sin(2\,x_2) + 1400\,u\cos(x_2) + 450\,x_5^2\sin(\sigma_3) - 600\,u\cos(x_2 - 2\,x_3) + 225\,x_6^2\sin(x_2 + x_3)}{50\,\sigma_2} \\ -\dfrac{300\sin(x_2 - x_3)\,x_5^2 + 135\sin(\sigma_3)\,x_6^2 - 2943\,\sigma_1 + 2943\sin(x_3) + 200\,u\cos(2\,x_2 - x_3) - 200\,u\cos(x_3)}{75\cos(2\,x_2) + 135\cos(\sigma_3) - 390} \end{pmatrix}$$

where

$$\sigma_1 = \sin(2\,x_2 - x_3)$$

$$\sigma_2 = 5\cos(2\,x_2) + 9\cos(\sigma_3) - 26$$

$$\sigma_3 = 2\,x_2 - 2\,x_3$$