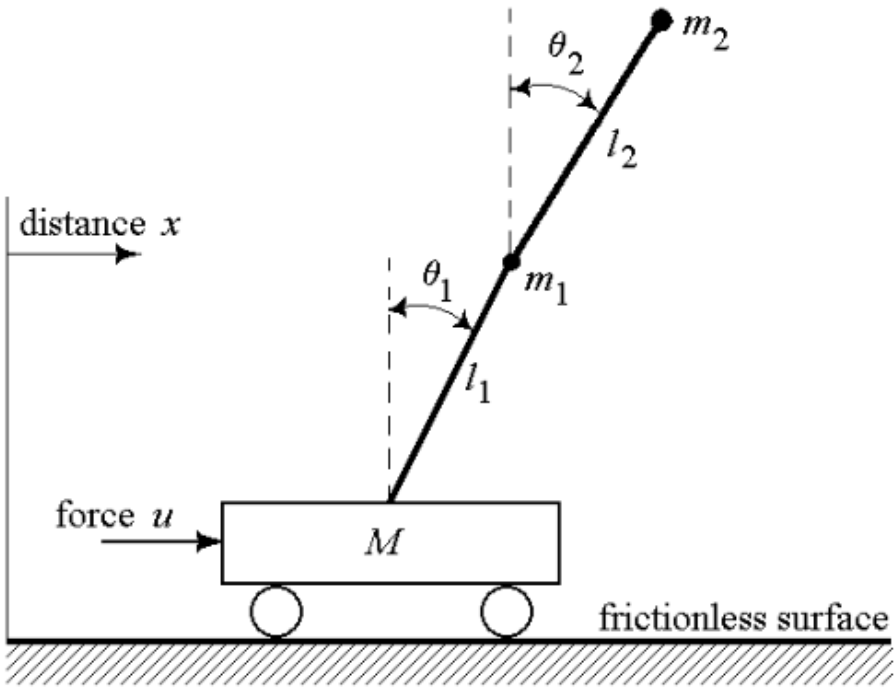
FunWork #1

Victoria Nagorski

Derive Equations of Motion

The problem begins with constructing the equations of motion through the Lagrangian for the double inverted pendulum on a cart (DIPC) as given in the figure below:



With the given properties of the system:

- $m_1 = 0.5kg$
- $l_1 = 0.5m$
- $m_2 = 0.75kg$
- $l_2 = 0.75m$
- $M = 1.5kg$
- $g = 9.81m/sec^2$

These values will not be exchanged until the very end so the derivation remains true for all values. We are also given that the state variables follow the following definition: $x_1 = x$, $x_2 = \theta_1$, $x_3 = \theta_2$, $x_4 = \dot{x}$, $x_5 = \dot{\theta_1}$, $x_6 = \dot{\theta_2}$.

With the problem statement, the following assumptions will be made:

- The body is rigid
- The rods are mass-less
- We are working with a simplified point-mass system

We will be using a set of tools when solving for the equations of motion. One of the first tools is the Transport Theorem- which allows us to go one frame to another in kinematics. It takes the following form:

$$\frac{^N d\vec{\rho}_1}{dt} = \frac{^A d\vec{\rho}_1}{dt} + \vec{\omega}_{A/N} \times \vec{\rho}_1$$

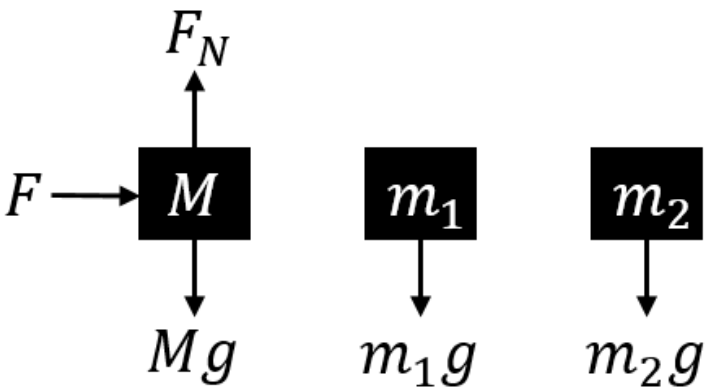
N in this example is the inertial reference frame. A is the first frame we jump to going through the kinematics for the vector $\vec{\rho}_1$. We will also use the following principles for enegry:

$$T = \frac{1}{2} mv \cdot v$$

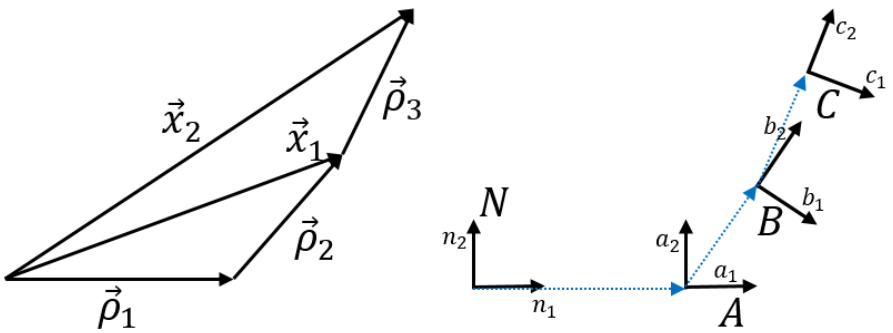
$$T_{total} = T_1 + T_2 + ... + T_n$$

$$V_{total} = V_1 + V_2 + ... + V_n$$

The first step is draw the Free Body Diagrams (FBDs) of each mass as shown in the following figure:



The frames of the problem and vectors are then described in the following figure:



The following definitions were then made:

- $\vec{\rho_1} = x\hat{a}_1$
- $\vec{\omega}_{B/A} = -\dot{\theta}_1\hat{b}_3$
- $\vec{\omega}_{C/B} = -\dot{\theta}_3\hat{c}_3$
- $\vec{\omega}_{A/N} = 0$
- $\hat{b}_3 = \hat{c}_3$
- $\theta_2 = \theta_1 + \theta_3$
- $\vec{\omega}_{C/B} = -(\dot{\theta}_2 - \dot{\theta}_1)\hat{c}_3$

For each mass we are going to derive the kinematics, change coordinate frames (as applicable), solve for the kinetic energy, and then solve for the potential energy. From there, the Lagrangian can be solved for to find the equations of motion.

Mass 1:

First derive the kinematics:

$$\frac{^N d\vec{\rho_1}}{dt} = \frac{^A d\vec{\rho_1}}{dt} + \vec{\omega}_{A/N} \times \vec{\rho_1}$$

Which simplifies to:

$$\frac{^N d\vec{\rho_1}}{dt} = \frac{^A d\vec{\rho_1}}{dt}$$
$$\frac{^N d\vec{\rho_1}}{dt} = \dot{x}\hat{a}_1$$

The kinetic energy for this mass:

$$\boxed{T_1 = \frac{1}{2}M\dot{x}^2}$$

The potential energy for this mass:

$$\boxed{V_1 = 0}$$

Mass 2:

Derive the kinematics:

$$\frac{^N d\vec{x}_1}{dt} = \frac{^N d\vec{\rho_1}}{dt} + \frac{^N d\vec{\rho}_2}{dt}$$
$$\frac{^N d\vec{\rho}_2}{dt} = \frac{^A d\vec{\rho}_2}{dt} + \vec{\omega}_{A/N} \times \vec{\rho_2} \Rightarrow \frac{^N d\vec{\rho}_2}{dt} = \frac{^A d\vec{\rho}_2}{dt}$$
$$\frac{^A d\vec{\rho}_2}{dt} = \frac{^B d\vec{\rho}_2}{dt} + \vec{\omega}_{B/A} \times \vec{\rho_2} \Rightarrow \frac{^A d\vec{\rho}_2}{dt} = \vec{\omega}_{B/A} \times \vec{\rho_2}$$

Which simplifies to:

$$\frac{^N d\vec{x}_1}{dt} = \frac{^A d\vec{\rho_1}}{dt} + \vec{\omega}_{B/A} \times \vec{\rho_2}$$
$$\frac{^N d\vec{x}_1}{dt} = \dot{x}\hat{a}_1 + \dot{\theta}_1 l_1 \hat{b}_1$$

Which can be written in the following coordinate frame:

$$\frac{^N d\vec{x}_1}{dt} = \dot{x}\hat{a}_1 + \dot{\theta}_1 l_1(\cos\theta_1\hat{a}_1 + \sin\theta_1\hat{a}_2)$$

The kinetic energy for this mass is:

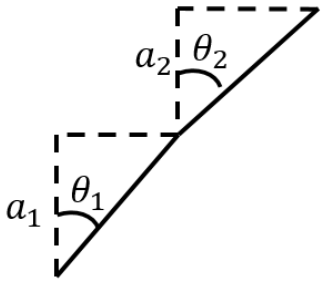
$$T_2 = \frac{1}{2}m_1(\dot{x}\hat{a}_1 + \dot{\theta}_1 l_1(\cos\theta_1\hat{a}_1 + \sin\theta_1\hat{a}_2))(\dot{x}\hat{a}_1 + \dot{\theta}_1 l_1(\cos\theta_1\hat{a}_1 + \sin\theta_1\hat{a}_2))$$

$$T_2 = \frac{1}{2}m_1(\dot{x} + \dot{\theta}_1 l_1\cos\theta_1)^2 + (\dot{\theta}_1 l_1\sin\theta_1)^2$$

$$T_2 = \frac{1}{2}m_1(\dot{x} + \dot{\theta}_1^2 l_1^2\cos^2\theta_1 + 2\dot{x}\dot{\theta}_1 l_1\cos\theta_1 + \dot{\theta}_1^2 l_1^2\sin^2\theta_1)$$

$$\boxed{T_2 = \frac{1}{2}m_1(\dot{x}^2 + \dot{\theta}_1^2 l_1^2 + 2\dot{x}\dot{\theta} l_1\cos\theta_1)}$$

The potential energy for this mass can be solved using the following diagram:



$$\boxed{V_2 = m_1 l_1 g\cos\theta_1}$$

Mass 3:

Derive the kinematics:

$$\frac{^N d\vec{x}_2}{dt} = \frac{^N d\vec{\rho}_1}{dt} + \frac{^N d\vec{\rho}_2}{dt} + \frac{^N d\vec{\rho}_3}{dt}$$

$$\frac{^N d\vec{\rho}_3}{dt} = \frac{^A d\vec{\rho}_3}{dt} + \vec{\omega}_{A/N}\times\vec{\rho}_3 \Rightarrow \frac{^N d\vec{\rho}_3}{dt} = \frac{^A d\vec{\rho}_3}{dt}$$

$$\frac{^A d\vec{\rho}_3}{dt} = \frac{^B d\vec{\rho}_3}{dt} + \vec{\omega}_{B/A}\times\vec{\rho}_3$$

$$\frac{^B d\vec{\rho}_3}{dt} = \frac{^C d\vec{\rho}_3}{dt} + \vec{\omega}_{C/B}\times\vec{\rho}_3 \Rightarrow \frac{^B d\vec{\rho}_3}{dt} = \vec{\omega}_{C/B}\times\vec{\rho}_3$$

$$\frac{^A d\vec{\rho}_3}{dt} = (\vec{\omega}_{B/A} + \vec{\omega}_{C/B})\times\vec{\rho}_3$$

Which simplifies to:

$$\frac{^N d\vec{x}_2}{dt} = \frac{^A d\vec{\rho}_1}{dt} + \vec{\omega}_{B/A}\times\vec{\rho}_2 + (\vec{\omega}_{B/A} + \vec{\omega}_{C/B})\times\vec{\rho}_3$$

$$\frac{^N d\vec{x}_2}{dt} = \dot{x}\hat{a}_1 + \dot{\theta}_1 l_1\hat{b}_1 + \dot{\theta}_2 l_2\hat{c}_1$$

Which can be written in the following coordinate frame:

$$\frac{^N d\vec{x}_2}{dt} = \dot{x}\hat{a}_1 + \dot{\theta}_1 l_1\hat{b}_1 + \dot{\theta}_2 l_2(\cos\theta_2\hat{a}_1 + \sin\theta_2\hat{a}_2)$$

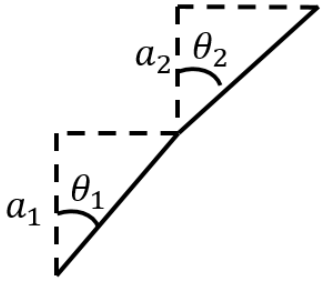
The kinetic energy for this mass is:

$$T_3 = \frac{1}{2}m_2(\dot{x}\hat{a}_1 + \dot{\theta}_1 l_1(\cos\theta_1\hat{a}_1 + \sin\theta_1\hat{a}_2) + \dot{\theta}_2 l_2(\cos\theta_2\hat{a}_1 + \sin\theta_2\hat{a}_2))(\dot{x}\hat{a}_1 + \dot{\theta}_1 l_1(\cos\theta_1\hat{a}_1 + \sin\theta_1\hat{a}_2) + \dot{\theta}_2 l_2(\cos\theta_2\hat{a}_1 + \sin\theta_2\hat{a}_2))$$

$$T_3 = \frac{1}{2}m_2(\dot{x}^2 + \dot{\theta}_1^2 l_1^2 \cos^2\theta_1 + \dot{\theta}_2^2 l_2^2 \cos^2\theta_2 + 2\dot{x}\dot{\theta}_1 l_1 \cos\theta_1 + 2\dot{x}\dot{\theta}_2 l_2 \cos\theta_2 + 2\dot{\theta}_1\dot{\theta}_2 l_1 l_2 \cos\theta_1 \cos\theta_2 + \dot{\theta}_1^2 l_2^2 \sin^2\theta_1 + \dot{\theta}_2^2 l_2^2 \sin^2\theta_2 + 2\dot{\theta}_1\dot{\theta}_2 l_1 l_2 \sin\theta_1 \sin\theta_2)$$

$$\boxed{T_3 = \frac{1}{2}m_2(\dot{x}^2 + \dot{\theta}_1^2 l_1^2 + \dot{\theta}_2^2 l_2^2 + 2\dot{x}(\dot{\theta}_1 l_1 \cos\theta_1 + \dot{\theta}_2 l_2 \cos\theta_2) + 2\dot{\theta}_1\dot{\theta}_2 l_1 l_2 \cos(\theta_1 - \theta_2)}$$

The potential energy for this mass can be solved for using the diagram again:



$$\boxed{V_3 = m_2 g(l_1 \cos\theta_1 + l_2 \cos\theta_2)}$$

The next part is to combine all the solved for parts into the Lagrangian $L = T - V$.

$$L = \frac{1}{2}(M + m_1 + m_2)\dot{x}^2 + \frac{1}{2}(m_1 + m_2)\dot{\theta}_1^2 l_1^2 + (m_1 + m_2)\dot{x}\dot{\theta}_1 l_1 \cos\theta_1 + \frac{1}{2}m_2(\dot{\theta}_2^2 l_2^2) + m_2\dot{x}\dot{\theta}_2 l_2 \cos\theta_2 + m_2\dot{\theta}_1\dot{\theta}_2 l_1 l_2 \cos(\theta_1 - \theta_2) - m_1 l_1 g \cos\theta_1 - m_2 g(l_1 \cos\theta_1 + l_2 \cos\theta_2)$$

$$\frac{d}{dt}\left\{\frac{\partial L}{\partial \dot{q}_j}\right\} - \frac{\partial L}{\partial q_j} = Q_{nc_j}$$

There are three variables of which the equations are dependent: $q_1 = x$, $q_2 = \theta_1$, and $q_3 = \theta_2$.
Starting with $q_1 = x$:

$$\frac{\partial L}{\partial \dot{x}} = (M + m_1 + m_2)\dot{x} + (m_1 + m_2)\dot{\theta}_1 l_1 \cos\theta_1 + m_2\dot{\theta}_2 l_2 \cos\theta_2$$

$$\frac{d}{dt}\left\{\frac{\partial L}{\partial \dot{x}}\right\} = (M + m_1 + m_2)\ddot{x} + (m_1 + m_2)l_1(\ddot{\theta}_1 \cos\theta_1 - \dot{\theta}_1^2 \sin\theta_1) + m_2 l_2(\ddot{\theta}_2 \cos\theta_2 - \dot{\theta}_2^2 \sin\theta_2)$$

$$\frac{\partial L}{\partial x} = 0$$

$q_2 = \theta_1$:

$$\frac{\partial L}{\partial \dot{\theta}_1} = (m_1 + m_2)\dot{\theta}_1 l_1^2 + (m_1 + m_2)\dot{x} l_1 \cos\theta_1 + m_2\dot{\theta}_2 l_1 l_2 \cos(\theta_1 - \theta_2)$$

$$\frac{d}{dt}\left\{\frac{\partial L}{\partial \dot{\theta}_1}\right\} = (m_1+m_2)\ddot{\theta}_1 l_1^2 + (m_1+m_2)l_1(\ddot{x}\cos\theta_1 - \dot{x}\dot{\theta}_1 \sin\theta_1) + m_2 l_1 l_2(\ddot{\theta}_2 \cos(\theta_1-\theta_2) - \dot{\theta}_2 \sin(\theta_1-\theta_2)(\dot{\theta}_1-\dot{\theta}_2))$$

$$\frac{\partial L}{\partial \theta_1} = -(m_1 + m_2)\dot{x}\dot{\theta}_1 l_1 \sin\theta_1 - m_2\dot{\theta}_1\dot{\theta}_2 l_1 l_2 \sin(\theta_1 - \theta_2) + m_1 l_1 g \sin\theta_1 + m_2 g l_1 \sin\theta_1$$

$q_3 = \theta_2$:

$$\frac{\partial L}{\partial \dot{\theta}_2} = m_2\dot{\theta}_2 l_2^2 + m_2\dot{x} l_2 \cos\theta_2 + m_2\dot{\theta}_1 l_1 l_2 \cos(\theta_1 - \theta_2)$$

$$\frac{d}{dt}\left\{\frac{\partial L}{\partial \dot{\theta}_2}\right\} = m_2\ddot{\theta}_2 l_2^2 + m_2 l_2(\ddot{x}\cos\theta_2 - \dot{x}\dot{\theta}_2 \sin\theta_2) + m_2 l_1 l_2(\ddot{\theta}_1 \cos(\theta_1 - \theta_2) - \dot{\theta}_1 \sin(\theta_1 - \theta_2)(\dot{\theta}_1 - \dot{\theta}_2))$$

$$\frac{\partial L}{\partial \theta_2} = -m_2 \dot{x}\dot{\theta}_2 l_2 \sin\theta_2 + m_2 \dot{\theta}_1 \dot{\theta}_2 l_1 l_2 \sin(\theta_1 - \theta_2) + m_2 l_2 g \sin\theta_2$$

Put the equations together, and simplify by expanding and using trigonometry:
Equation 1:

$$(M + m_1 + m_2)\ddot{x} + (m_1 + m_2) l_1 \ddot{\theta}_1 \cos\theta_1 + m_2 l_2 \ddot{\theta}_2 \cos\theta_2 - (m_1 + m_2) l_1 \dot{\theta}_1^2 \sin\theta_1 - m_2 l_2 \dot{\theta}_2^2 \sin\theta_2 = F$$

Equation 2:

$$(m_1 + m_2) l_1 \ddot{x} \cos\theta_1 + (m_1 + m_2)\ddot{\theta}_1 l_1^2 + m_2 l_1 l_2 \ddot{\theta}_2 \cos(\theta_1 - \theta_2) + m_2 l_1 l_2 \dot{\theta}_2^2 \sin(\theta_1 - \theta_2) - m_1 l_1 g \sin\theta_1 - m_2 l_1 g \sin\theta_1 = 0$$

Equation 3:

$$m_2 l_2 \ddot{x} \cos\theta_2 + m_2 l_1 l_2 \ddot{\theta}_1 \cos(\theta_1 - \theta_2) + m_2 \ddot{\theta}_2 l_2^2 - m_2 l_1 l_2 \dot{\theta}_1^2 \sin(\theta_1 - \theta_2) - m_2 g l_2 \sin\theta_2 = 0$$

These equations of motion follow the form $M\ddot{x} + C\dot{x} + G = Hu$

$$M = \begin{bmatrix} (M + m_1 + m_2) & (m_1 + m_2) l_1 \cos\theta_1 & m_2 l_2 \cos\theta_2 \\ (m_1 + m_2) l_1 \cos\theta_1 & (m_1 + m_2) l_1^2 & (m_2 l_1 l_2)\cos(\theta_1 - \theta_2) \\ (m_2 l_2)\cos\theta_2 & (m_2 l_1 l_2)\cos(\theta_1 - \theta_2) & m_2 l_2^2 \end{bmatrix}$$

$$C = \begin{bmatrix} 0 & -(m_1 + m_2) l_1 \dot{\theta}_1 \sin\theta_1 & -m_2 l_2 \dot{\theta}_2 \sin\theta_2 \\ 0 & 0 & (m_2 l_1 l_2)\dot{\theta}_2 \sin(\theta_1 - \theta_2) \\ 0 & -(m_2 l_1 l_2)\dot{\theta}_1 \sin(\theta_1 - \theta_2) & 0 \end{bmatrix}$$

$$G = \begin{bmatrix} 0 \\ -(m_1 l_1 + m_2 l_1) g \sin\theta_1 \\ -(m_2 l_2) g \sin\theta_2 \end{bmatrix}$$

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

Rewrite the equation to be $\ddot{x} = -M^{-1}C\dot{x} - M^{-1}G + M^{-1}Hu$. This form can be reduced if we use a change in variable k.

$$k_1 = x$$

$$k_2 = \dot{k}_1 = \dot{x}$$

$$k_3 = \dot{k}_2 = \ddot{x}$$

This allows us to rewrite:

$$\begin{bmatrix} \dot{k}_1 \\ \dot{k}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & -M^{-1}C \end{bmatrix} \begin{bmatrix} k_1 \\ k_2 \end{bmatrix} + \begin{bmatrix} 0 \\ -M^{-1}G \end{bmatrix} + \begin{bmatrix} 0 \\ M^{-1}H \end{bmatrix} u$$

The last step is to rewrite the solved for equations in terms of the state variables given earlier in the problem. To make life easier, I am going define constants and exchange those into the matrix. (This step will also make it easier to track things when I go to code.)

- $r_1 = M + m_1 + m_2$
- $r_2 = (m_1 + m_2)l_1$
- $r_3 = m_2 l_2$
- $r_4 = (m_1 + m_2)l_1^2$
- $r_5 = m_2 l_1 l_2$
- $r_6 = m_2 l_2^2$
- $f_1 = (m_1 l_1 + m_2 l_2)g$
- $f_2 = m_2 l_2 g$

Rewrite with the correct state variables:

$$\boxed{M = \begin{bmatrix} r_1 & r_2\cos x_2 & r_3\cos x_3 \\ r_2\cos x_2 & r_4 & r_5\cos(x_2 - x_3) \\ r_3\cos x_3 & r_5\cos(x_2 - x_3) & r_6 \end{bmatrix}}$$

$$\boxed{C = \begin{bmatrix} 0 & -r_2 x_5 \sin x_2 & -r_3 x_6 \sin x_3 \\ 0 & 0 & r_5 x_6 \sin(x_2 - x_3) \\ 0 & -r_5 x_5 \sin(x_2 - x_3) & 0 \end{bmatrix}}$$

$$\boxed{G = \begin{bmatrix} 0 \\ -f_1 \sin x_2 \\ -f_2 \sin x_3 \end{bmatrix}}$$

$$\boxed{H = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}}$$

$$\boxed{\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \end{bmatrix} = \begin{bmatrix} 0 & I_{3x3} \\ 0 & -M^{-1}C \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} + \begin{bmatrix} 0 \\ -M^{-1}G \end{bmatrix} + \begin{bmatrix} 0 \\ M^{-1}H \end{bmatrix} u}$$

```
%% Nonlinear Equations Load
% Problem Contants
m1 = 0.5;                               % kg
l1 = 0.5;                               % m
m2 = 0.75;                              % kg
l2 = 0.75;                              % m
M = 1.5;                                % kg
g = 9.81;                               % m/sec^2

% Solve for Matrix Contants
r1 = M + m1 + m2;
r2 = (m1 + m2) * l1;
r3 = m2*l2;
r4 = (m1 + m2) * l1^2;
r5 = m2 * l1 * l2;
r6 = m2 * l2^2;
f1 = (m1 * l1 + m2 * l1) * g;
f2 = m2 * l2 * g;

% Define Non-Linear Matrices
syms x1 x2 x3 x4 x5 x6 u
M = [  r1         r2*cos(x2)      r3*cos(x3);
     r2*cos(x2)      r4          r5*cos(x2-x3);
     r3*cos(x3) r5*cos(x2-x3)        r6];
C = [0 -r2*x5*sin(x2)     -r3*x6*sin(x3);
      0        0             r5*x6*sin(x2-x3);
      0 -r5*x5*sin(x2-x3)        0];
G = [       0;
     -f1 * sin(x2);
     -f2 * sin(x3)];
H = [1;0;0];
```

Linearization

Linearization through Taylor's expansion utilizes perturbations around an equilibrium point to approximate a linear model. The variables can be described as $x = x_e + \Delta x$ or $u = u_e + \Delta u$. These equations say that state and inputs are an accumulation of both equilibrium point and the small perturbations around that equilibrium. Taylor's expansion is the following:

$$\frac{d}{dt}x = f(x_e + \Delta x, u_e + \Delta u) = f(x_e, u_e) + \frac{\partial f}{\partial x}(x_e, u_e)\Delta x + \frac{\partial f}{\partial u}(x_e, u_e)\Delta u + H.O.T.$$

The differentials can then be lumped into the following matrix form (Jacobian matrices):

$$A = \frac{\partial f}{\partial x}(x_e, u_e) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}$$

$$B = \frac{\partial f}{\partial u}(x_e, u_e) = \begin{bmatrix} \frac{\partial f_1}{\partial u_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial u_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}$$

The end result is $\frac{d}{dt}\Delta x = A\Delta x + B\Delta u$.
The equilibrium points will be about the origin. In the code, the equilibrium points for states and inputs are put in the same vector. This was a preference choice.

$$EquilibriumVector = \begin{bmatrix} x_e \\ u_e \end{bmatrix} = \begin{bmatrix} x_{1e} \\ x_{2e} \\ x_{3e} \\ x_{4e} \\ x_{5e} \\ x_{6e} \\ u_e \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

The previous derived $\dot{x}$ is set equal to $f(x, u)$. It is this $f(x, u)$ that is then passed into the 'jacobian' function in Matlab. Once the Jacobian matrix is solved for, the equilibrium points need to be passed into the matrix. What results is the A and B matrices. The following is the code used to linearize the non-linear equations of motion:

```
%% Linearize Non-Linear Equations
% Solve for the Non-Linear Functions
matrix1 = [zeros(3,3)    eye(3);      % Break up equation
           zeros(3,3) -M^-1 * C];
matrix2 = [zeros(3,1); -M^-1 * G];   % Break up equation
matrix3 = [zeros(3,1); M^-1 * H];    % Break up equation
x = [x1; x2; x3; x4; x5; x6];        % Define x vector
f = matrix1 * x + matrix2 + matrix3 * u;

% Linearize with the Jacobian
```

```
equil = [0;0;0;0;0;0;0];                % Equilibrium point (about orgin)
a = jacobian(f,x);                      % Create Jacobian matrix for A
b = jacobian(f,u);                      % Create Jacobian matrix for B
A = double(subs(a,[x;u],equil));        % Plug in equilibrium points
B = double(subs(b,[x;u],equil));        % Plug in equilibrium points
C = [1 0 0 0 0 0;                       % Define the C matrix for Sys
     0 1 0 0 0 0;
     0 0 1 0 0 0];
D = zeros(3,1);                         % Define the D matrix for Sys

% Save Matrices for Easy Access
save('Values.mat','A','B','C','D','l1','l2','f','x')
```

The resulting matrices that are used for the rest of this report:

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & -8.1759 & 0 & 0 & 0 & 0 \\ 0 & 65.4 & -29.43 & 0 & 0 & 0 \\ 0 & -32.7 & 32.7 & 0 & 0 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0.6667 \\ -1.3333 \\ 0 \end{bmatrix}$$

We are also going to define matrices C and D at this point. We only have 3 outputs, so we should only have 3 rows within C. D will be a zero matrix. In most instances (at least in aerospace engineering), D will be a zero matrix.

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

$$D = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Controllability and Observability

First we start with controllability of the system- which looks at the A and B matrix. We will start by building the controllabilty matrix and then testings its rank. The controllability matrix is built in the following way for this example:

$$C = \begin{bmatrix} B & AB & A^2B & ... & A^5B \end{bmatrix}$$

The result is a $6x6$ matrix. The 'cntb' command in Matlab will also build this exact matrix. Checking the rank in Matlab, we find that the rank is equal to 6. This allows us to move forward with solving for the controller form of the system. It is to be noted that the B matrix is really a vector since the system only has one input. The first step is to take the inverse the controllability matrix. We then take last row of the inverse and detonate the this as q_1. With q_1 we form the T matrix:

$$T = \begin{bmatrix} q_1 \\ q_1A \\ \vdots \\ q_1A^5 \end{bmatrix} = \begin{bmatrix} 0.0023 & 0.0012 & 0.0018 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.0023 & 0.0012 & 0.0018 \\ 0 & 0 & 0.0229 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.0229 \\ 0 & -0.75 & 0.75 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -0.75 & 0.75 \end{bmatrix}$$

Once we have the T matrix, then we are able to solve for $\tilde{A}$ and $\tilde{B}$.

$$\boxed{\tilde{A} = TAT^{-1} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & -1176.2 & 0 & 98.1 & 0 \end{bmatrix}}$$

$$\boxed{\tilde{B} = TB = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}}$$

Going onto the observability of the system, we begin to look at the A and C matrix. The Matlab command 'obsv' will build the following matrix:

$$O = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^5 \end{bmatrix}$$

Which results in $18x6$ matrix. The rank of this matrix is 6. Moving forward, we begin to construct the observer form. The first thing to note is that C has multiple rows- so it will require a few

more steps compared to the controller form. Also to note, when looking at the associated code, to make things easier I save a variable of the transposes instead of continuously transposing the matrices. The first step is to build the controllabilty matrix with (A^T, C^T). This results in the following matrix:

$$\begin{bmatrix} c_1^T & c_2^T & c_3^T & A^T c_1^T & A^T c_2^T & A^T c_3^T & (A^T)^2 c_1^T & (A^T)^2 c_2^T & (A^T)^2 c_3^T & ... \end{bmatrix} =$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & ... \\ 0 & 1 & 0 & 0 & 0 & 0 & -8.18 & 65.4 & -32.7 & ... \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & -29.43 & 32.7 & ... \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & ... \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & ... \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & ... \end{bmatrix}$$

From the above matrix, we can see that the first 6 vectors are linearly independent. This means that d_1 = 6. The next step is to create the matrix L:

$$L = \begin{bmatrix} c_1^T & A^T c_1^T & c_2^T & A^T c_2^T & c_3^T & A^T c_3^T \end{bmatrix}$$

We then take the inverse of this matrix, and since $d_1 = 6$ we will take the last row from L^{-1} to be equal to q_1. L^{-1}:

$$L^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

We are then able to build T:

$$T = \begin{bmatrix} q_1 \\ q_1 A^T \\ \vdots \\ q_1 (A^T)^5 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -29.4 & 32.7 \\ 0 & -29.4 & 32.7 & 0 & 0 & 0 \\ 0 & 0 & 0 & 240.6 & -2887.1 & 2031.7 \\ 240.6 & -2887.1 & 2031.7 & 0 & 0 & 0 \end{bmatrix}$$

The last step is to solve for $\hat{A}$ and $\hat{C}$:

$$\boxed{\hat{A} = (T^T)^{-1} A T^T = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1176.2 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 98.1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}}$$

$$\hat{C} = CT^T = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 240.6 \\ 0 & 0 & 0 & -29.4 & 0 & -2887.1 \\ 0 & 1 & 0 & 32.7 & 0 & 2031.7 \end{bmatrix}$$

The code used to build the previous steps:

```
%% Controllability and Observability
% Controller Form
Co = ctrb(A,B);                         % Create controllability matrix
n = rank(Co);                           % Rank of controllability matrix
Co_inv = Co^-1;                         % Inverse
q = Co_inv(end,:);                      % Last row of inverse
T = [q;                                 % Transformation matrix
     q * A;
     q * A^2;
     q * A^3;
     q * A^4;
     q * A^5];
A_c = T*A*T^-1                          % Controller form A
B_c = T*B                               % Controller form B

% Observer Form
O = obsv(A,C)                           % Observability matrix
n_O = rank(O)                           % Rank of observability matrix
A_t = A';                               % Transpose of A
B_t = C';                               % Transpose of C saved under B^T
% Calculate part of the controllability matrix of A^T and C^T
temp = [B_t(:,1) B_t(:,2) B_t(:,3)...
         A_t * B_t(:,1) A_t * B_t(:,2) A_t * B_t(:,3)...
         A_t^2 * B_t(:,1) A_t^2 * B_t(:,2) A_t^2 * B_t(:,3)];
rank(temp)                              % Calculate the rank of partial ↩
    matrix
L = [B_t(:,1) A_t * B_t(:,1) ...        % Create L with independent ↩
    columns
      B_t(:,2) A_t * B_t(:,2) ...
      B_t(:,3) A_t * B_t(:,3)];
L_inv = L^-1;                           % Calculate inverse of L
q1 = L_inv(end,:);                      % Pull out last row of L
T_o = [q1;                              % Calculate the transformation ↩
    matrix
        q1 * A_t;
        q1 * A_t^2;
        q1 * A_t^3;
        q1 * A_t^4;
        q1 * A_t^5];
A_o = T_o'^-1*A*T_o'                    % Observer form of A
C_o = C*T_o'                            % Observer form of C
```

Transfer Function

Starting off with the state equations $\dot{x} = Ax + Bu$ and $y = Cx$, we are going to derive the transfer functions for the state space. Take the Laplace of the state space:

$$\mathcal{L}\{\dot{x} = Ax + Bu\} \Rightarrow sX(s) = AX(s) + BU(s)$$

$$\mathcal{L}\{y = Cx\} \Rightarrow Y(s) = CX(s)$$

Rearrange:

$$(sI - A)X(s) = BU(s) \Rightarrow X(s) = (sI - A)^{-1}BU(s)$$

$$Y(s) = C(sI - A)^{-1}BU(s) \Rightarrow H(s) = \frac{Y(s)}{U(s)} = C(sI - A)^{-1}B$$

Since there are 3 inputs and 1 output, there will be 3 transfer functions in the form:

$$x : H_1(s) = \frac{Y_1(s)}{U(s)}$$

$$\theta_1 : H_2(s) = \frac{Y_2(s)}{U(s)}$$

$$\theta_2 : H_3(s) = \frac{Y_3(s)}{U(s)}$$

The denominator of each of the transfer functions will be the same across the three transfer functions, and will be the characteristic equation of the system. There is 1 of 2 ways you can choose to solve the transfer function using Matlab- both equivalent. However, Method 2 will give a prettier output with more complex systems.

```
%% Transfer Function
% Method 1
syms s
Y = C * (s*eye(6)-A)^-1 * B
% Method 2
[b,a] = ss2tf(A,B,C,D)
```

The final transfer functions are:

$$x : H_1(s) = \frac{0.6667s^4 - 54.5s^2 + 427.716}{s^6 - 98.1s^4 + 1176.2s^2}$$

$$\theta_1 : H_2(s) = \frac{-1.3333s^4 + 43.6s^2}{s^6 - 98.1s^4 + 1176.2s^2}$$

$$\theta_2 : H_3(s) = \frac{43.6s^2}{s^6 - 98.1s^4 + 1176.2s^2}$$

Animate Model

The next few paragraphs will describe the code that follows at the end of the section. The first section is the initial values that can be changed by the user at the beginning of the simulation. The system is to be animated from a time interval of $[t_0, t_f]$. In the case of this specific simulation, $t_0 = 0$ and $t_f = 60$ was used. 2000 frames were use in the simulation. The θ_1 and θ_2 angles are described in this section at a negative angle since the problem had the bar to the right of the vertical. Next, the values are converted to radians from degrees. Then time step 'dt' is also calculated in this section.

Initialization of variables is next. A vector of time is created by telling Matlab to create a vector from 0 to 60 with a spacing of 'dt'. The states matrix is then initialized. Columns represent 'snapshots' of time of the state vector. The next two sections describe the object itself and give distances relative to one another. The section after that gives the first frame of the movie.

The section 'Animate Object' is where at each frame the states for the next time step are solved for. This is done using forward Euler Method:

$$\frac{dx(t_k)}{dt} \approx \frac{x(t_{k+1}) - x(t_k)}{dt}$$

$$x(t_{k+1}) = x(t_k) + dt * f(t_k, x(t_k), u(t_k))$$

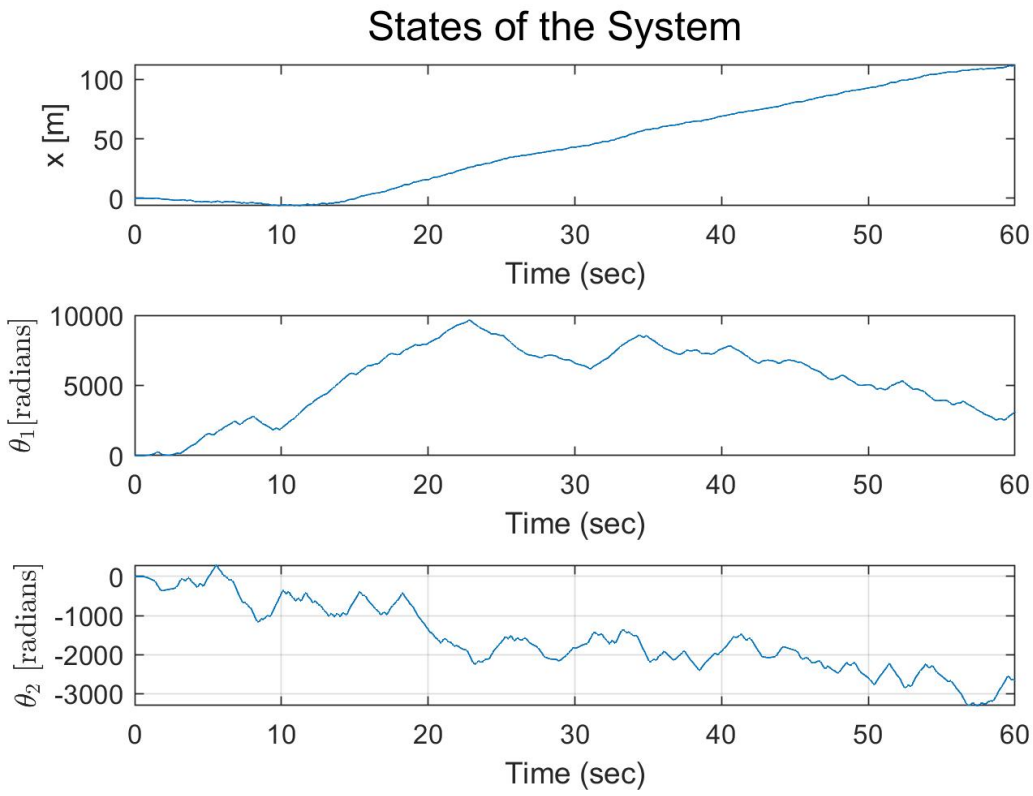
Which then can be re-arranged for our purposes to:

$$\boxed{x_{k+1} = (Ax + Bu)dt + x_k}$$

A video can be found here: `https://youtu.be/gPLsh0b_HH0`

Watching the video, one will notice that the pendulum does not act as one would expect in real life due to the fact friction terms were not added to the model.

The graphs of the States vs. Time are on the next page.




```
%% Animate an Object Example
clear; close all; clc;

load('Values.mat');
x_vec = x;                                  % Differentiate between to 'x's
clear x                                     % Clear old variable from mem
syms u                                      % Initialize u as a syms var

% Initial Values - Change these values
font_size = 10;
theta_d1 = -0.01*180/pi;                    % Theta 1 [degrees]
theta_d2 = -0.02*180/pi;                    % Theta 2 [degrees]
tfinal = 60;                                % Final time [s]
t = 0;                                      % Time [s]
nframes = 2000;                             % Number of points
%u = 0;

% Convert Values - Do not hard-code these values
theta1 = theta_d1*pi/180;                   % Theta 1 convered to radians
theta2 = theta_d2*pi/180;                   % Theta 2 convered to radians
dt = (tfinal-t)/nframes;                    % Step size of animation

% Initialize variables
```

```
time = t:dt:tfinal;                          % Create vector of time
states = zeros(6,nframes);                   % Matrix to hold states for time
states(:,1) = [0;theta1;theta2;0;0;0];       % Initial values

% Initialize Object
data_init1 = [0 l1]';                        % Pendulum length for length 1
data_init2 = [0 l2]';                        % Pendulum length for length 2
R1 = [cos(theta1) -sin(theta1);
      sin(theta1)  cos(theta1)];             % Rotation matrix 1
R2 = [cos(theta2) -sin(theta2);
      sin(theta2)  cos(theta2)];             % Rotation matrix 2
data1 = R1 * data_init1;                     % Initial pendulum position
data2 = R2 * data_init2;                     % Initial pendulum position

% Define Distances
d1_x = states(1,1);                          % Where M is located
d2_x = states(1,1) + data1(1);               % Where m1 is located in x
d3_x = d2_x + data2(1);                      % Where m2 is located in x
d3_y = data1(2)+data2(2);                    % Where m2 is located in y

% Graphics for pendulum 1
bar1 = line('xdata',[d1_x d2_x],'ydata',[0 data1(2)]','linewidth',3);
mass1 = line('xdata',d2_x,'ydata',data1(2),'marker','o',...
    'markersize',10,'MarkerFaceColor','b');
hinge1 = line('xdata',d1_x,'ydata',0,'marker','o','markersize',7);
% Graphics for pendulum 2
bar2 = line('xdata',[d2_x d3_x],'ydata',[data1(2) d3_y]','linewidth',3);
mass2 = line('xdata',d3_x,'ydata',d3_y,'marker','o',...
    'markersize',10,'MarkerFaceColor','b');
hinge2 = line('xdata',d2_x,'ydata',data1(2),'marker','o','markersize',7);
% Graphics for mass M
mass = line('xdata',d1_x,'ydata',0,'marker','s',...
    'markersize',17,'MarkerFaceColor','b');
% Graphics for Ground
Ground = line('xdata',[-4 4],'ydata',[-.2 -.20]','linewidth',3);
% Graphics for handle
axis(4*[-1 1 -1/2 1/2]); grid on
set(gca,'Fontsize',font_size)
set(gca,'dataaspectratio',[1 1 1])
box on

% Animate Object
vidObj = VideoWriter('Double_Pendulum.avi');
open(vidObj);
```

```
Frames = moviein(nframes)
for i = 2:(nframes + 1)
    % Equations of motion
    states(:,i) = dt * (double(subs(f,[x_vec;u],[states(:,i-1);0]))) +...
        states(:,i-1);
    % Pull out locations
    x = states(1,i);
    theta1 = states(2,i);
    theta2 = states(3,i);

    % Calculate new locations to update video frames
    R1 = [cos(theta1) -sin(theta1);
     sin(theta1)  cos(theta1)];          % Rotation matrix 1
    R2 = [cos(theta2) -sin(theta2);
     sin(theta2)  cos(theta2)];          % Rotation matrix 2
    data1 = R1 * data_init1;             % Initial pendulum position
    data2 = R2 * data_init2;             % Initial pendulum position
    d1_x = states(1,i);                  % Where M is located
    d2_x = states(1,i) + data1(1);       % Where m1 is located in x
    d3_x = d2_x + data2(1);              % Where m2 is located in x
    d3_y = data1(2) + data2(2);          % Where m2 is located in y

    % Change the property values m1
    set(bar1,'xdata',[d1_x d2_x],'ydata',[0 data1(2)]);
    set(mass1,'xdata',d2_x,'ydata',data1(2));
    set(hinge1,'xdata',d1_x,'ydata',0)
    % Change the property values m2
    set(bar2,'xdata',[d2_x d3_x],'ydata',[data1(2) d3_y]);
    set(mass2,'xdata',d3_x,'ydata',d3_y);
    set(hinge2,'xdata',d2_x,'ydata',data1(2))
    % Change the property values M
    set(mass,'xdata',d1_x,'ydata',0);


    % For the animation
    drawnow;
    Frames(:,i) = getframe;
    writeVideo(vidObj,Frames(:,i));
end

% Plot state vs time
hold on
sgtitle('States of the System')
subplot(3,1,1)
plot(time,states(1,:))
xlabel('Time (sec)')
ylabel('x [m]')
```

```
114 set(gca,'Fontsize',font_size)
115 subplot(3,1,2)
116 plot(time,states(2,:)*180/pi)
117 xlabel('Time (sec)')
118 ylabel('$\theta_1$[radians]','Interpreter','latex')
119 set(gca,'Fontsize',font_size)
120 subplot(3,1,3)
121 plot(time,states(3,:)*180/pi)
122 xlabel('Time (sec)')
123 ylabel('$\theta_2$ [radians]','Interpreter','latex')
124 grid
125 set(gca,'Fontsize',font_size)
126 hold off
127 close(vidObj);
```