This was my final project for my graduate level Linear Controls Class at UCSD. The assignment was to complete steps 2-5 of the example on the following page, taken from Numerical Renaissance (http://numerical-renaissance.com) by Thomas Bewley.

Example 21.1 Swing up and stabilization of the dual inverted pendulum

To illustrate the power of the

The nonlinear equations of motion of a single pendulum are given in §17.5 [see (17.16)]. Denoting $\{m_1, I_1, \ell_1\}$ and $\{m_2, I_2, \ell_2\}$ the mass, moment of inertia, and distance from the center of mass to the point where it is attached to the cart of pendulums 1 and 2 respectively, these equations may easily be extended to the dual pendulum case depicted in Figure 21.1 as follows:

$$\begin{aligned}
&(m_c+m_1+m_2)\frac{d^2x}{dt^2} - m_1\ell_1\cos\theta_1\frac{d^2\theta_1}{dt^2} - m_2\ell_2\cos\theta_2\frac{d^2\theta_2}{dt^2} + m_1\ell_1\sin\theta_1\Big(\frac{d\theta_1}{dt}\Big)^2 + m_2\ell_2\sin\theta_2\Big(\frac{d\theta_2}{dt}\Big)^2 = u,\\
&-m_1\ell_1\cos\theta_1\frac{d^2x}{dt^2} + (I_1+m_1\ell_1^2)\frac{d^2\theta_1}{dt^2} - m_1 g\ell_1\sin\theta_1 = 0,\\
&-m_2\ell_2\cos\theta_2\frac{d^2x}{dt^2} + (I_2+m_2\ell_2^2)\frac{d^2\theta_2}{dt^2} - m_2 g\ell_2\sin\theta_2 = 0.
\end{aligned} \tag{21.32}$$

Linearization of this system is performed by taking $x = \overline{x} + x'$, $\theta_1 = \overline{\theta}_1 + \theta_1'$, $\theta_2 = \overline{\theta}_2 + \theta_2'$, and $u = \overline{u} + u'$ in (21.32), expanding with Taylor series, multiplying out, applying the fact that the nominal condition $\{\overline{x}, \overline{\theta}_1, \overline{\theta}_2, \overline{u}\}$ is itself also a solution of (21.32), and keeping only those terms which are linear in the perturbation (primed) quantities, as terms that are quadratic or higher in the perturbations are negligible if the perturbations are sufficiently small. Taking $\{\overline{x} = 0, \overline{\theta}_1 = 0, \overline{\theta}_2 = 0, \overline{u} = 0\}$, the linearized equations of motion of the dual inverted pendulum are

$$\begin{aligned}
&(m_c+m_1+m_2)\frac{d^2x'}{dt^2} - m_1\ell_1\frac{d^2\theta_1'}{dt^2} - m_2\ell_2\frac{d^2\theta_2'}{dt^2} = u',\\
&-m_1\ell_1\frac{d^2x'}{dt^2} + (I_1+m_1\ell_1^2)\frac{d^2\theta_1'}{dt^2} - m_1 g\ell_1\theta_1' = 0,\\
&-m_2\ell_2\frac{d^2x'}{dt^2} + (I_2+m_2\ell_2^2)\frac{d^2\theta_2'}{dt^2} - m_2 g\ell_2\theta_2' = 0.
\end{aligned} \tag{21.33}$$

On the other hand, considering an unsteady nominal trajectory $\{\overline{x}(t), \overline{\theta}_1(t), \overline{\theta}_2(t), \overline{u}(t)\}$ gives the tangent linear equations for the dual inverted pendulum:

$$\begin{aligned}
&(m_c+m_1+m_2)\frac{d^2x'}{dt^2} - m_1\ell_1\cos(\overline{\theta}_1)\frac{d^2\theta_1'}{dt^2} - m_2\ell_2\cos(\overline{\theta}_2)\frac{d^2\theta_2'}{dt^2}\\
&\quad + m_1\ell_1\Big[\frac{d^2\overline{\theta}_1}{dt^2}\sin(\overline{\theta}_1)\,\theta_1' + \Big(\frac{d\overline{\theta}_1}{dt}\Big)^2(\cos\overline{\theta}_1)\theta_1' + 2\frac{d\overline{\theta}_1}{dt}\sin(\overline{\theta}_1)\frac{d\theta_1'}{dt}\Big]\\
&\quad + m_2\ell_2\Big[\frac{d^2\overline{\theta}_2}{dt^2}\sin(\overline{\theta}_2)\,\theta_2' + \Big(\frac{d\overline{\theta}_2}{dt}\Big)^2(\cos\overline{\theta}_2)\theta_2' + 2\frac{d\overline{\theta}_2}{dt}\sin(\overline{\theta}_2)\frac{d\theta_2'}{dt}\Big] = u',\\
&-m_1\ell_1\cos(\overline{\theta}_1)\frac{d^2x'}{dt^2} + (I_1+m_1\ell_1^2)\frac{d^2\theta_1'}{dt^2} - m_1\ell_1\Big[g(\cos\overline{\theta}_1)\,\theta_1' - \frac{d^2\overline{x}}{dt^2}\sin(\overline{\theta}_1)\,\theta_1'\Big] = 0,\\
&-m_2\ell_2\cos(\overline{\theta}_2)\frac{d^2x'}{dt^2} + (I_2+m_2\ell_2^2)\frac{d^2\theta_2'}{dt^2} - m_2\ell_2\Big[g(\cos\overline{\theta}_2)\,\theta_2' - \frac{d^2\overline{x}}{dt^2}\sin(\overline{\theta}_2)\,\theta_2'\Big] = 0.
\end{aligned} \tag{21.34}$$

Step 1: Optimizing $\overline{\mathbf{u}}(t)$ on $t \in [0,T]$. We first optimize the nominal control input $\bar{\mathbf{u}}(t)$ on $t \in [0,T]$ to swing up both pendulums. The nonlinear equation of motion (21.32) may be written in the first-order form (21.1a)-(21.1b) by defining

$$\mathbf{x} = \begin{pmatrix} x \\ \theta_1 \\ \theta_2 \\ dx/dt \\ d\theta_1/dt \\ d\theta_2/dt \end{pmatrix}, \quad E(\theta_1,\theta_2) = \begin{pmatrix} 1 & & & & & \\ & 1 & & & & \\ & & 1 & & & \\ & & & m_c+m_1+m_2 & -m_1\ell_1\cos\theta_1 & -m_2\ell_2\cos\theta_2 \\ & & & -m_1\ell_1\cos\theta_1 & I_1+m_1\ell_1^2 & \\ & & & -m_2\ell_2\cos\theta_2 & & I_2+m_2\ell_2^2 \end{pmatrix},$$

$$N(\mathbf{x},\mathbf{f},\mathbf{u}) = \begin{pmatrix} dx/dt \\ d\theta_1/dt \\ d\theta_2/dt \\ -m_1\ell_1\sin\theta_1\,(d\theta_1/dt)^2 - m_2\ell_2\sin\theta_2\,(d\theta_2/dt)^2 + u \\ m_1 g\ell_1\sin\theta_1 \\ m_2 g\ell_2\sin\theta_2 \end{pmatrix}$$

The initial condition on the system is $\mathbf{x}^T(0) = \begin{pmatrix} 0 & \pi & \pi & 0 & 0 & 0 \end{pmatrix}$; the desired (unstable) terminal condition is $\mathbf{x}^T(T) = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$. We target the dynamics of interest on the horizon $(0,T)$ by defining the cost function (21.1c), at least initially, such that $Q = I$ and $R = \alpha^2 I$ and $Q_T = \beta^2 I$. We will take $\alpha = O(1)$ and β as relatively large in order to emphasize bringing the system close to the desired state at the end of the swingup (that is, at $t = T$). We can adjust these weighting matrices in the cost function after optimizing the nominal trajectory of the system if there are any features of this initial optimized trajectory that are found to be undesirable. The tangent linear equation (21.34) may be written in the form (21.2) by taking $E = E(\overline{\theta}_1, \overline{\theta}_2)$, using the (symmetric) formula for $E(\theta_1,\theta_2)$ given above, and defining

$$\mathbf{x}' = \begin{pmatrix} x' \\ \theta_1' \\ \theta_2' \\ dx'/dt \\ d\theta_1'/dt \\ d\theta_2'/dt \end{pmatrix}, \quad A = \begin{pmatrix} & & & 1 & & \\ & & & & 1 & \\ & & & & & 1 \\ -a_{42} & -a_{43} & & -a_{45} & -a_{46} & \\ a_{52} & & & & & \\ & a_{63} & & & & \end{pmatrix}, \quad B = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix},$$

where

$$a_{42} = m_1\ell_1\left[\frac{d^2\overline{\theta}_1}{dt^2}\sin\overline{\theta}_1 + \left(\frac{d\overline{\theta}_1}{dt}\right)^2\cos\overline{\theta}_1\right], \quad a_{45} = 2m_1\ell_1\frac{d\overline{\theta}_1}{dt}\sin\overline{\theta}_1, \quad a_{52} = m_1\ell_1\left[g\cos\overline{\theta}_1 - \frac{d^2\overline{x}}{dt^2}\sin\overline{\theta}_1\right],$$

$$a_{43} = m_2\ell_2\left[\frac{d^2\overline{\theta}_2}{dt^2}\sin\overline{\theta}_2 + \left(\frac{d\overline{\theta}_2}{dt}\right)^2\cos\overline{\theta}_2\right], \quad a_{46} = 2m_2\ell_2\frac{d\overline{\theta}_2}{dt}\sin\overline{\theta}_2, \quad a_{63} = m_2\ell_2\left[g\cos\overline{\theta}_2 - \frac{d^2\overline{x}}{dt^2}\sin\overline{\theta}_2\right].$$

With the state vector $\mathbf{x}$, the nonlinear equation of motion in first-order form, the matrices $\{Q, Q_\mathbf{u}, Q_T\}$ defining the cost function, and the matrices $\{E, A, B\}$ defining the tangent linear model so identified, the powerful machinery of (21.1.1) may be used to optimize the control input $\bar{\mathbf{u}}(t)$ on $t \in [0,T]$ to swing up both pendulums.

Step 2: Computing $K(t)$ on $t \in [0,T]$.

Step 3: Computing $L(t)$ on $t \in [0,T]$.

Step 4: Computing K for $t \in [T,\infty)$.

Step 5: Computing L for $t \in [T,\infty)$.

MAE 280B: Dual Inverted Pendulum

Brian Filarsky

Note: The steps in this document do not correspond to the steps in Numerical Renaissance. They have been renumbered, starting with 1. Anytime Step 1 from Numerical Renaissance is referenced, it is indicated as "Example 21.1 Step 1 in Numerical Renaissance." All other references to steps refer to this document.

Step 1: Calculating K for $t \in [T, \infty)$

The first step to calculating K on the infinite horizon is to calculate the **CT Algebraic Riccati Equation (CARE)** for X

$$0 = A^H XE + E^H XA - E^H XBR^{-1}B^H XE + Q$$

Utilizing A, B, and E from Example 21.1 Step 1 in Numerical Renaissance, and taking $Q = I$ and $R = I$, this can be solved using methods developed in §4.5.2 of Numerical Renaissance, or by using the MATLAB care function. With X in hand, we now proceed to calculate K via

$$K = -R^{-1}B^H XE$$

Step 2: Calculating L for $t \in [T, \infty)$

L is calculated on the infinite horizon in the same way as K. The first step is to calculate the **CARE** for P

$$0 = APE + E^H PA^H - E^H PC^H Q_2^{-1} CPE + Q_1$$

Utilizing A and E from Step 1, and taking $C = [I\ 0]$, $Q_1 = I$ and $Q_2 = I$, we obtain P. With P in hand, we now proceed to calculate L via

$$L = -E^H PC^H Q_2^{-1}$$

Step 3: Calculating $K(t)$ for $t \in [0, T]$

To calculate $K(t)$, we will need to march the **Differential Riccati Equation (DRE)** for X backward in time for $t = T \rightarrow 0$, with $X(T) = X$ from Step 1.

$$-E^H \frac{dX}{dt} E = A^H XE + E^H XA - E^H XBR^{-1}B^H XE + Q$$

$$\mathbf{f}(A, E, X) = -E^{-H}A^H X - XAE^{-1} + XBR^{-1}B^H X - E^{-H}QE^{-1}$$

Taking $Q = I$ and $R = I$, and utilizing the fourth-order Runge-Kutta (**RK4**) from §10.4.1.1 of Numerical Renaissance, we have

$$\begin{aligned}
\mathbf{f}_1 &= \mathbf{f}(A_n, E_n, X_n) \\
\mathbf{f}_2 &= \mathbf{f}((A_n + A_{n-1})/2, (E_n + E_{n-1})/2, X_n - (h/2)\mathbf{f}_1) \\
\mathbf{f}_3 &= \mathbf{f}((A_n + A_{n-1})/2, (E_n + E_{n-1})/2, X_n - (h/2)\mathbf{f}_2) \\
\mathbf{f}_4 &= \mathbf{f}(A_{n-1}, E_{n-1}, X_n - h\mathbf{f}_3) \\
X_{n-1} &= X_n - h[\frac{1}{6}\mathbf{f}_1 + \frac{1}{3}\mathbf{f}_2 + \frac{1}{3}\mathbf{f}_3 + \frac{1}{6}\mathbf{f}_4]
\end{aligned}$$

Using the value of X for each timestep from $T \rightarrow 0$, we can now calculate

$$K(t) = -R^{-1}B^H X(t)E$$

Step 4: Calculating $L(t)$ for $t \in [0,\ T]$

To calculate $L(t)$, we will need to march the **DRE** for P forward in time for $t = 0 \rightarrow T$, with $P(0) = P_0$. P_0 is the covariance matrix of our initial state estimate, $\hat{\mathbf{x}}_0$. Assuming we have a known initial state, we will set $P_0 = 0$.

$$E^H \frac{dP}{dt} E = APE + E^H P A^H - E^H P C^H Q_2^{-1} C P E + Q_1$$

$$\mathbf{f}(A, E, P) = E^{-H} A P + P A^H E^{-1} - P C^H Q_2^{-1} C P + E^{-H} Q_1 E^{-1}$$

Utilizing **RK4** marching forward in time, and taking $Q_1 = I$ and $Q_2 = I$ we have

$$\begin{aligned}
\mathbf{f}_1 &= \mathbf{f}(A_n, E_n, P_n) \\
\mathbf{f}_2 &= \mathbf{f}((A_n + A_{n+1})/2, (E_n + E_{n+1})/2, P_n + (h/2)\mathbf{f}_1) \\
\mathbf{f}_3 &= \mathbf{f}((A_n + A_{n+1})/2, (E_n + E_{n+1})/2, P_n + (h/2)\mathbf{f}_2) \\
\mathbf{f}_4 &= \mathbf{f}(A_{n+1}, E_{n+1}, P_n + h\mathbf{f}_3) \\
P_{n+1} &= P_n + h[\frac{1}{6}\mathbf{f}_1 + \frac{1}{3}\mathbf{f}_2 + \frac{1}{3}\mathbf{f}_3 + \frac{1}{6}\mathbf{f}_4]
\end{aligned}$$

Keeping the value of P for each timestep from $0 \rightarrow T$, we can now calculate

$$L(t) = -E^H P(t) C^H Q_2^{-1}$$

Step 5: Putting it all together

The Eigenvalues of the linearized $E^{-1}A$ at upright ($\mathbf{x} = [0\ 0\ 0\ 0\ 0\ 0]^T$), $E^{-1}(A + BK)$, and $E^{-1}(A + LC)$ are shown in Table 1. As expected, the Eigenvalues of the system without the controller are unstable, but the Eigenvalues of $E^{-1}(A + BK)$ and $E^{-1}(A + LC)$ are now stable given the solution of the **CARE**.

Table 1: Eigenvalues with all gains set to I

$\lambda_{E^{-1}A}$	$\lambda_{E^{-1}(A+BK)}$	$\lambda_{E^{-1}(A+LC)}$
6.0435	-6.2315 + 0i	-9.7593 - 7.6303i
-6.0435	-5.8608 + 0i	-9.7593 + 7.6303i
-4.2918	-4.3814 + 0i	-4.7562 - 1.9235i
4.2918	-4.2047 + 0i	-4.7562 + 1.9235i
0	-0.2256 - 0.2149i	-0.9953 + 0i
0	-0.2256 + 0.2149i	-0.0975 + 0i

Now that we have K, L, $K(t)$ and $L(t)$ for $t \in [0, \infty)$, we can simulate the swingup and stabilization. Utilizing **RK4** marching forward in time, we can implement the plant utilizing the nonlinear equations of motion, including state disturbances ($\mathbf{w}_1$) and measurement noise ($\mathbf{w}_2$).

$$\frac{d\mathbf{x}}{dt} = E(\mathbf{x})^{-1} N(\mathbf{x}, \mathbf{u}) + \mathbf{w}_1$$

$$\mathbf{y} = C\mathbf{x} + \mathbf{w}_2$$

Using $\bar{\mathbf{u}}$ and $\bar{\mathbf{x}}$ from Example 21.1 Step 1 in Numerical Renaissance, as well as K, L, $K(t)$, and $L(t)$ from Steps 1 through 4, we can implement our **Extended Kalman Filter (EFK)** estimator and controller

Step 5: Putting it all together continued on next page...

$$\frac{d\hat{\mathbf{x}}}{dt} = E(\hat{\mathbf{x}})^{-1}N(\hat{\mathbf{x}}, \mathbf{u}) - E(\hat{\mathbf{x}})^{-1}L(t)(\mathbf{y} - C\hat{\mathbf{x}})$$

$$\mathbf{u} = \bar{\mathbf{u}} + K(t)(\hat{\mathbf{x}} - \bar{\mathbf{x}})$$

where, after time T, $\bar{\mathbf{u}} = 0$, $\bar{\mathbf{x}} = 0$, and *K(t)* and *L(t)* become K and L, the equations simplify to

$$\frac{d\hat{\mathbf{x}}}{dt} = E(\hat{\mathbf{x}})^{-1}N(\hat{\mathbf{x}}, \mathbf{u}) - E(\hat{\mathbf{x}})^{-1}L(\mathbf{y} - C\hat{\mathbf{x}})$$

$$\mathbf{u} = K\hat{\mathbf{x}}$$

At this point, taking $\mathbf{w}_1 = 0$ and $\mathbf{w}_2 = 0$, and setting $\hat{\mathbf{x}}_0 = \mathbf{x}_0$, we should have a stable solution given the Eigenvalues above, and our state estimate error $\tilde{\mathbf{x}} = \hat{\mathbf{x}}$ - $\mathbf{x} = 0 \; \forall$ t $\in [0, \infty)$. Figure 1 shows the evolution of the position of the cart and the pendulums with no noise or perturbations, and confirms that $\tilde{\mathbf{x}} = 0$. Adding Gaussian state disturbances, Gaussian measurement noise, and a perturbation to $\mathbf{x}_0 \neq \hat{\mathbf{x}}_0$, P_0 and the gains on R, Q, Q_1, and Q_2 are tuned to obtain the best state estimate and keep the system convergent. While a rule of thumb is to try to get the fastest Eigenvalues of $E^{-1}(A + LC)$ to be 2-5x faster then than fastest Eigenvalues of $E^{-1}(A + BK)$, the best results for this system were found at approximately 1.6x, as shown in table 2. Figure 2 shows the evolution of the position of the cart and the pendulums with tuned gains, perturbations and noise. Comparing figure 1 with figure 2 shows how much faster the tuned system converges, despite the perturbation and noise. It is also worth noting that all of the initial state errors converge within the first second. Figure 3 shows the evolution of the state velocities, and figure 4 shows the control input to the perturbed and noisy system.

Table 2: Eigenvalues with tuned gains

$\lambda_{E^{-1}(A+BK)}$	$\lambda_{E^{-1}(A+LC)}$
-9.9825 + 0.0000i	-16.0358 -14.6842i
-4.7962 - 0.6936i	-16.0358 +14.6842i
-4.7962 + 0.6936i	-6.5445 - 4.4055i
-2.7766 + 0.0000i	-6.5445 + 4.4055i
-1.1779 - 0.4461i	-3.1608 + 0i
-1.1779 + 0.4461i	-0.0971 + 0i

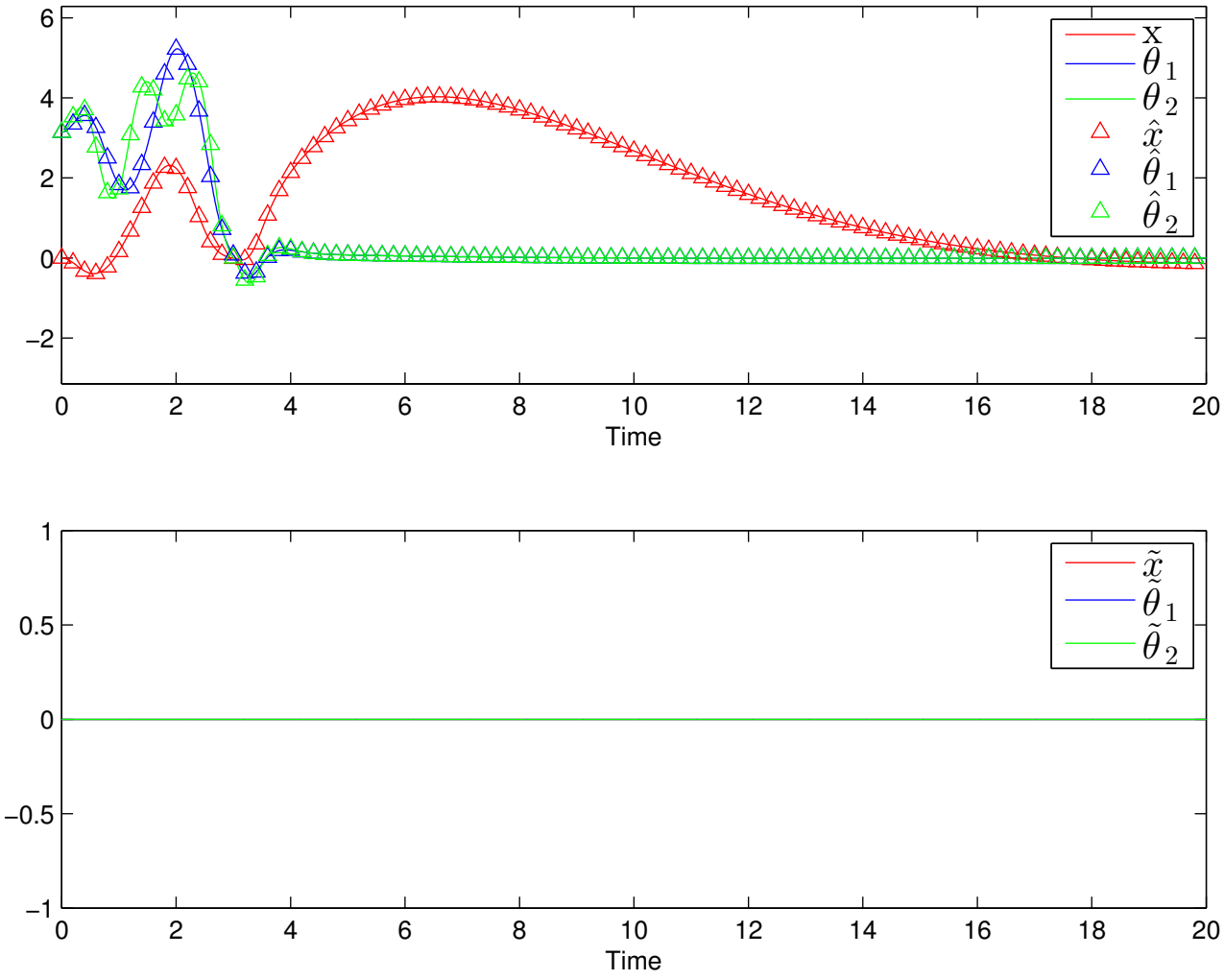

Figure 1: Position components of $\mathbf{x}$, $\hat{\mathbf{x}}$ and $\tilde{\mathbf{x}}$ with no disturbances, noise, or perturbations

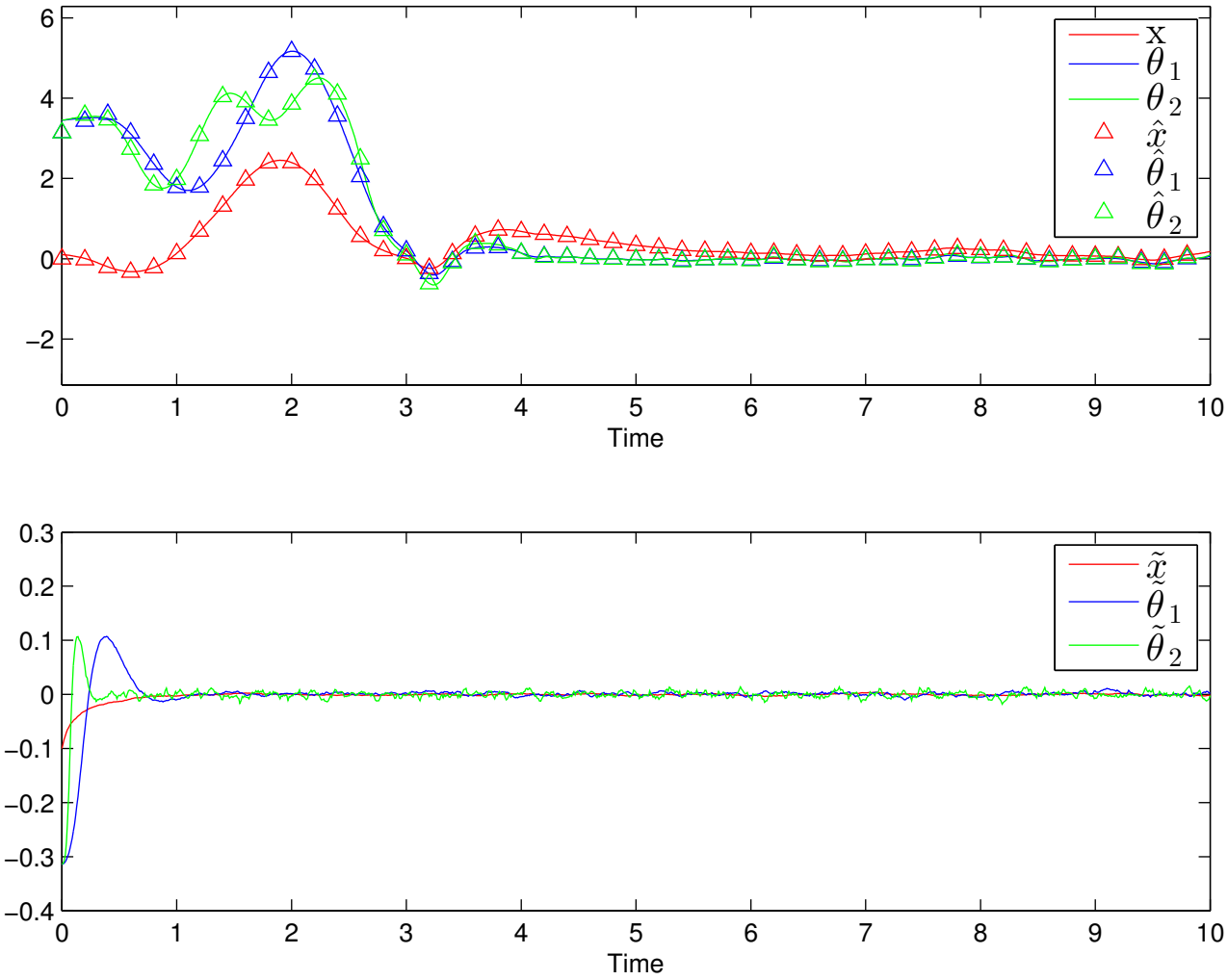

Figure 2: Position components of $\mathbf{x}$, $\hat{\mathbf{x}}$ and $\tilde{\mathbf{x}}$ with disturbances, noise, and initial perturbation

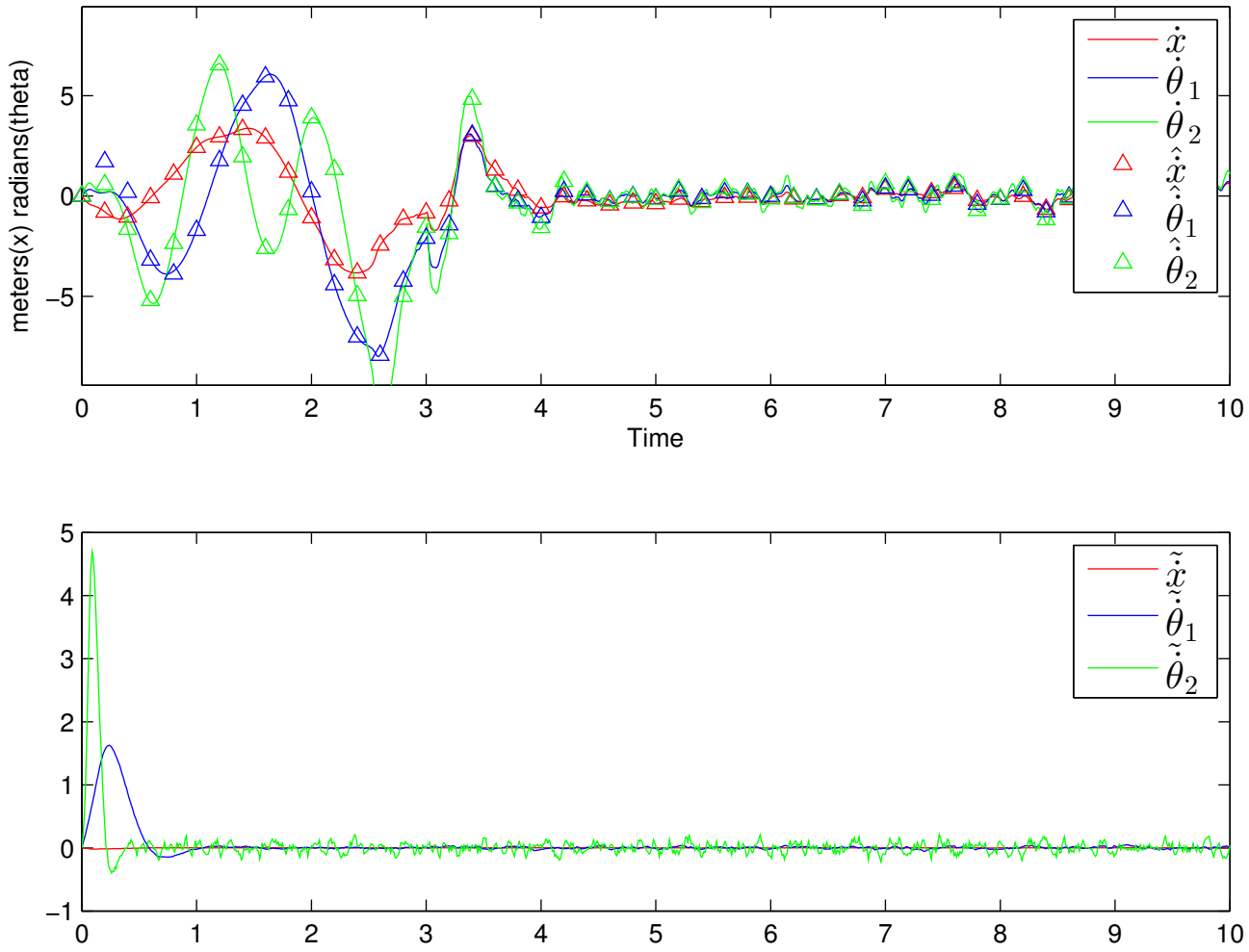

Figure 3: Velocity components of $\mathbf{x}$, $\hat{\mathbf{x}}$ and $\tilde{\mathbf{x}}$ with disturbances, noise, and initial perturbation

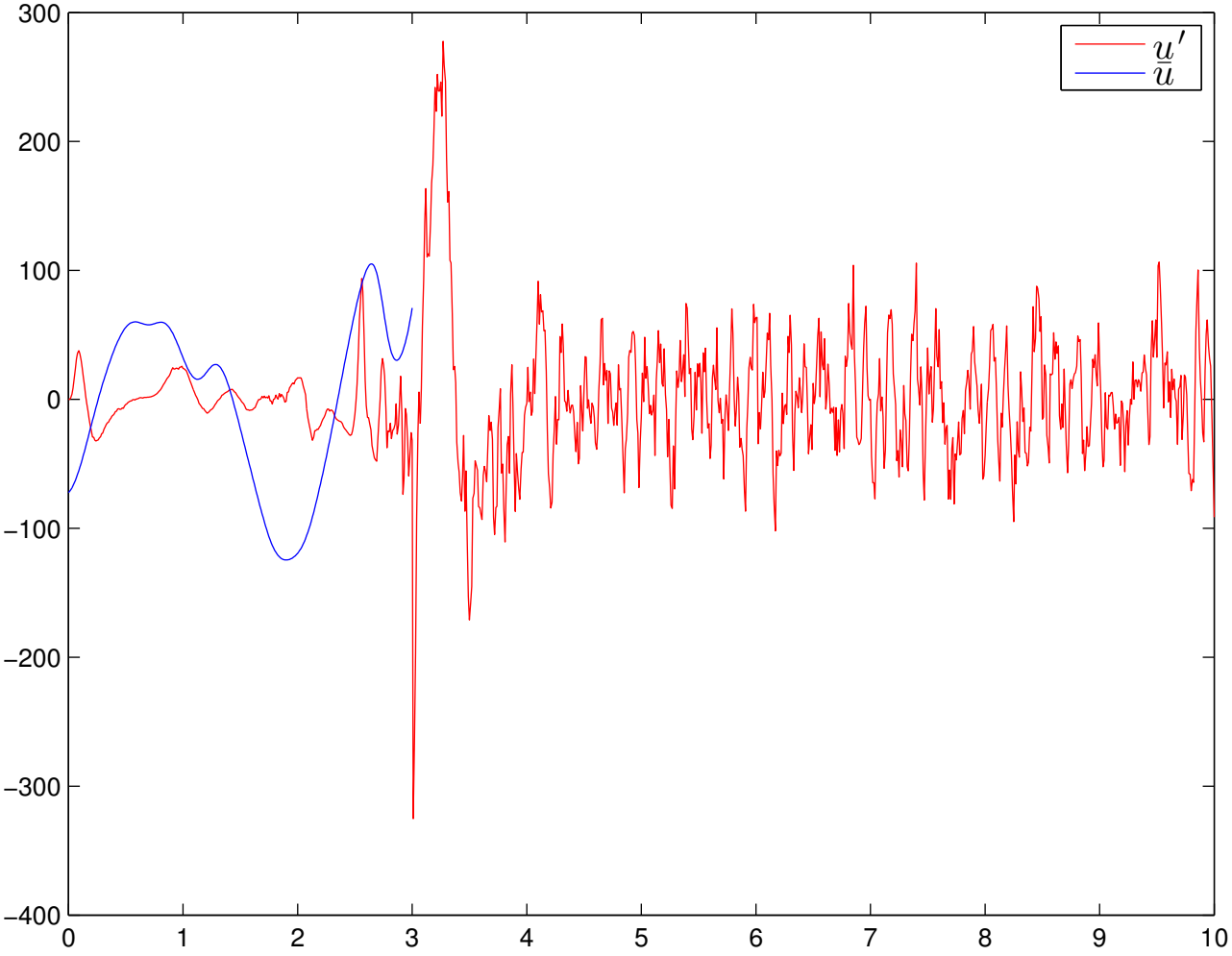

Figure 4: $\mathbf{u}'$ and $\bar{\mathbf{u}}$ with disturbances, noise, and initial perturbation

MATLAB Code

```
function [x_kp]=Dual_Pendulum_Cart(u_bar,x_bar,T)
s.h=0.01;                                          %Timestep .01 sec
s.N=T/s.h;                                         %N = Time/Timestep
s.N1=size(u_bar,1)-1;                              %Swingup time
t=(0:s.N)*s.h;                                     %time vector from 0 to T
s.mc=10;                                           %Mass Cart = 10
s.m1=0.2;                                          %Pendulum 1 Mass .2
s.L1=1;                                            %Pendulum 1 Length 1
s.ell1=s.L1/2;                                     %Pendulum 1 CG half of length
s.I1=s.m1*s.ell1^2/12;                             %Pendulum 1 Moment of Inertia
s.m2=0.1;                                          %Pendulum 2 Mass .1
s.L2=0.5;                                          %Pendulum 2 Length .5
s.ell2=s.L2/2;                                     %Pendulum 2 CG half of length
s.I2=s.m2*s.ell2^2/12;                             %Pendulum 2 Moment of Inertia
s.B=[0; 0; 0; 1; 0; 0];                            %B Matrix from State Space form.
s.C=[eye(3) zeros(3)];                             %C Matrix from State Space form.
x_plant=[.1; 1.1*pi; 1.1*pi; 0; 0; 0];             %Perturbed X_plant
x_hat=[0; pi; pi; 0; 0; 0];                        %Perurbation not reflected
x_kp = zeros(6,s.N+1);
x_kh = zeros(6,s.N+1);
x_kt = zeros(6,s.N+1);
u_prime = zeros(s.N+1,1);
L_k=zeros(6,3,301);
S1=zeros(6,1);
S2=zeros(6,3);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Gains
R_swingup=1;
Q_swingup= 250*diag([5 5 5 1 1 1]);
Q1_swingup= diag([1 1 1 1 1 1]);
Q2_swingup= .1 * diag([1 1 1]);

R_upright=1;
Q_upright= 250*diag([1 1 1 1 1 1]);
Q1_upright= diag([1 1 1 1 1 1]);
Q2_upright= .1 * diag([1 1 1]);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Compute L and K Offline
A_upright=Compute_A([0;0;0;0;0;0;0;0;0],s);
E_upright=Compute_E([0;0;0;0;0;0;0;0;0],s);
X=care(A_upright,s.B,Q_upright,R_upright,S1,E_upright);     %X to march back from
K_upright = -inv(R_upright)*s.B'*X*E_upright;               %Infinite time K
for n=s.N1:-1:0
    A1=Compute_A(x_bar(:,n+1),s);                            %As and Es for RK4
    E1=Compute_E(x_bar(:,n+1),s);
    if n>0
        A2=Compute_A(x_bar(:,n),s);
        E2=Compute_E(x_bar(:,n),s);
    else
        A2=Compute_A(x_bar(:,n+1),s);
        E2=Compute_E(x_bar(:,n+1),s);
    end
    f1 = -DRE(A1,s.B,E1,X,Q_swingup,R_swingup);             %Runge Kutta back marching
    f2 = -DRE((A1+A2)./2,s.B,(E1+E2)./2,X-s.h*f1/2,Q_swingup,R_swingup);
    f3 = -DRE((A1+A2)./2,s.B,(E1+E2)./2,X-s.h*f2/2,Q_swingup,R_swingup);
    f4 = -DRE(A2,s.B,E2,X-s.h*f3,Q_swingup,R_swingup);
    X=X-s.h*(f1/6+(f2+f3)/3+f4/6);                           %Updated X for timestep
    K_k(n+1,:)=-inv(R_swingup)*s.B'*X*E1;                    %Updated K for timestep
```

MATLAB Code continued on next page. . .

```
end
P=diag([1 0 0 0 0 0]);                                       %P0
for n=0:s.N1
    if n<s.N1
        A1=Compute_A(x_bar(:,n+1),s);                         %As and Es for RK4
        A2=Compute_A(x_bar(:,n+2),s);
        E1=Compute_E(x_bar(:,n+1),s);
        E2=Compute_E(x_bar(:,n+2),s);
    else
        A2=Compute_A(x_bar(:,n+1),s);
        E2=Compute_E(x_bar(:,n+1),s);
    end
    f1 = DRE(A1',s.C',E1,P,Q1_swingup,Q2_swingup);          %Runge Kutta forward marching
    f2 = DRE((A1'+A2')./2,s.C',(E1+E2)./2,P+s.h*f1/2,Q1_swingup,Q2_swingup);
    f3 = DRE((A1'+A2')./2,s.C',(E1+E2)./2,P+s.h*f2/2,Q1_swingup,Q2_swingup);
    f4 = DRE(A2',s.C',E2,P+s.h*f3,Q1_swingup,Q2_swingup);
    P=P+s.h*(f1/6+(f2+f3)/3+f4/6);                            %Updated P for timestep
    L_k(:,:,n+1)=E1'*-P*s.C'/Q2_swingup;                      %Updated L for timestep
end
P = care(A_upright',s.C',Q1_upright,Q2_upright,S2,E_upright);
L_upright=E_upright'*-P*s.C'/Q2_upright;                      %Infinite L
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%March forward for simulation
for n=0:s.N
    %Swingup phase
    if n<=s.N1
        u_prime(n+1,1)=K_k(n+1,:)*(x_hat-x_bar(1:6,n+1));   %Calculate u_prime
        u=u_bar(n+1,1) + u_prime(n+1,1);                     %Calculate u
        L=L_k(:,:,n+1);                                      %Pull L(t)
    %Infinite Horizon phase
    else
        u_prime(n+1,1) = K_upright*(x_hat);                  %Optimal control feedback
        u=u_prime(n+1,1);                                    %u just u prime
        L=L_upright;                                         %Use L
    end

    x_tilde=x_hat-x_plant;                                    %state error
    x_kp(:,n+1)=x_plant;                                      %plant state
    x_kh(:,n+1)=x_hat;                                        %estimator state
    x_kt(:,n+1)=x_tilde;                                      %store state error

    w1=normrnd(0,.01,6,1);                                    %Gaussian state disturbance
    w2=normrnd(0,.01,3,1);                                    %Gaussian measurement noise

    y_plant=(s.C*x_plant+w2);                                 %Measurements with noise
    f1_plant=RHS(x_plant,u,s);                                %RK4 for plant
    f2_plant=RHS(x_plant+s.h*f1_plant/2,u,s);
    f3_plant=RHS(x_plant+s.h*f2_plant/2,u,s);
    f4_plant=RHS(x_plant+s.h*f3_plant,u,s);
    x_plant=x_plant+s.h*(f1_plant/6+(f2_plant+f3_plant)/3+f4_plant/6+w1);

    E=Compute_E(x_hat,s);
    f1_hat=RHS(x_hat,u,s);                                    %RK4 for estimator
    f2_hat=RHS(x_hat+s.h*f1_hat/2,u,s);
    f3_hat=RHS(x_hat+s.h*f2_hat/2,u,s);
    f4_hat=RHS(x_hat+s.h*f3_hat,u,s);
    x_hat=x_hat+s.h*(f1_hat/6+(f2_hat+f3_hat)/3+f4_hat/6+w1-E\L*(y_plant-s.C*x_hat));
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Plots
figure(1);
clf;
```

```
subplot(2,1,1);
plot(t,x_kp(1,:),'r-',t,x_kp(2,:),'b-',t,x_kp(3,:),'g-');
hold on
plot(t(1,1:20:s.N),x_kh(1,1:20:s.N),'r^',t(1,1:20:s.N),x_kh(2,1:20:s.N),...
    'b^',t(1,1:20:s.N),x_kh(3,1:20:s.N),'g^');
legend({'x','${\theta}_{1}$','${\theta}_{2}$','$\hat{x}$','$\hat{\theta}_{1}$',...
    '$\hat{\theta}_{2}$'},'Interpreter','LaTex','fontsize',15)
xlabel('Time')
ylim([-pi 2*pi])
subplot(2,1,2);
plot(t,x_kt(1,:),'r-',t,x_kt(2,:),'b-',t,x_kt(3,:),'g-');
legend({'$\tilde{x}$','$\tilde{\theta}_{1}$','$\tilde{\theta}_{2}$'},'Interpreter',...
    'LaTex','fontsize',15)
xlabel('Time')
print -depsc epsFig1

figure(2);
clf;
subplot(2,1,1);
plot(t,x_kp(4,:),'r-',t,x_kp(5,:),'b-',t,x_kp(6,:),'g-');
hold on
plot(t(1,1:20:s.N),x_kh(4,1:20:s.N),'r^',t(1,1:20:s.N),x_kh(5,1:20:s.N),'b^',...
    t(1,1:20:s.N),x_kh(6,1:20:s.N),'g^');
legend({'$\dot{x}$','$\dot{\theta}_{1}$','$\dot{\theta}_{2}$','$\hat{\dot{x}}$',...
    '$\hat{\dot{\theta}}_{1}$','$\hat{\dot{\theta}}_{2}$'},'Interpreter','LaTex','fontsize',15)
xlabel('Time')
ylabel('meters(x) radians(theta)')
ylim([-3*pi 3*pi])
subplot(2,1,2);
plot(t,x_kt(4,:),'r-',t,x_kt(5,:),'b-',t,x_kt(6,:),'g-');
legend({'$\tilde{\dot{x}}$','$\tilde{\dot{\theta}}_{1}$','$\tilde{\dot{\theta}}_{2}$'},...
    'Interpreter','LaTex','fontsize',15)
print -depsc epsFig2

figure(3);
plot(t,u_prime(:,1),'red',t(1,1:301),u_bar(:,1));
legend({'$u^{\prime}$','$\bar{u}$'},'Interpreter','LaTex','fontsize',15)
print -depsc epsFig3
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function dX = DRE(A,B,E,X,Q,R)
dX = (E'\A'*X+X*A/E-X*B*inv(R)*B'*X+E'\Q/E);                %DRE used for marching X and P
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function A=Compute_A(x,s)                                    %Build A per Step 1
g=9.8;
a42=s.m1*s.ell1*(x(8)*sin(x(2))+x(5)^2*cos(x(2)));
a45=2*s.m1*s.ell1*x(5)*sin(x(2));
a43=s.m2*s.ell2*(x(9)*sin(x(3))+x(6)^2*cos(x(3)));
a46=2*s.m2*s.ell2*x(6)*sin(x(3));
a52=s.m1*s.ell1*(g*cos(x(2))-x(7)*sin(x(2)));
a63=s.m2*s.ell2*(g*cos(x(3))-x(7)*sin(x(3)));
A=[zeros(3) eye(3); 0 -a42 -a43 0 -a45 -a46; 0 a52 0 0 0 0; 0 0 a63 0 0 0];
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function E=Compute_E(x,s)                                    %Build E per Step 1
I=eye(3);
Z=zeros(3);
E=[I Z; Z [s.mc+s.m1+s.m2         -s.m1*s.ell1*cos(x(2)) -s.m2*s.ell2*cos(x(3));
    -s.m1*s.ell1*cos(x(2))  s.I1+s.m1*s.ell1^2            0            ;
    -s.m2*s.ell2*cos(x(3))          0              s.I2+s.m2*s.ell2^2   ]];
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function R=RHS(x,u,s)
E=Compute_E(x,s);                                            %Build E per Step 1
N=Compute_N(x,u,s);                                          %Build N per Step 1
R=E\N;                                                       %Compute state derivative per 21.1a
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function N=Compute_N(x,u,s)                                  %Build N per Step 1
N=[x(4); x(5); x(6); -s.m1*s.ell1*sin(x(2))*x(5)^2-s.m2*s.ell2*sin(x(3))*x(6)^2+u;
    s.m1*9.8*s.ell1*sin(x(2)); s.m2*9.8*s.ell2*sin(x(3)) ];
end
```