Taking the required values from Funwork 1

```
syms x1 x2 x3 x4 x5 x6 u
syms M m1 m2 l1 l2 g
M = 1.5;
m1 = 0.5;
l1 = 0.5;
m2 = 0.75;
l2 = 0.75;
g = 9.81;
```

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

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

```
% Taking derivative of the values
dL_dx1 = diff(L, x1);
dL_dx2 = diff(L, x2);
dL_dx3 = diff(L, x3);
dL_dx4 = diff(L, x4);
dL_dx5 = diff(L, x5);
dL_dx6 = diff(L, x6);
q = [x1 ;x2; x3];
qd = [x4; x5; x6];
syms qdd1 qdd2 qdd3 real
qdd = [qdd1; qdd2; qdd3];
Jaco = jacobian([dL_dx4;dL_dx5;dL_dx6],[q;qd]);
d_dt_dLdqdot = Jaco*[qd; qdd];
dLdq = [dL_dx1;dL_dx2;dL_dx3];
Force = [u; 0; 0];
Lagrange_eq = simplify(d_dt_dLdqdot - dLdq -Force);
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

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

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
```

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

```
f_sym =
```

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

where

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

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

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

FUNWORK 2

Problem 1. (4 pts) Start with the non-linear model of the double inverted pendulum on a cart (DIPC) from FunWork 1.

Solve the Lyapunov matrix equation for the linearized open-loop system. Take $Q = I_6$. Is the open-loop system asymptotically stable, that is, is the equilibrium state of interest asymptotically stable in the sense of Lyapunov? Explain.

For the first step towrad this problem we linearize the system around 0, just like it was done in Problem 1

```
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)));
```

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

LYAPUNOV STABILITY

$A^T P + \mathrm{PA} = -Q$

Where **P must be a symmetric positive-definite matrix**, and **Q must be a symmetric positive-definite matrix**. This condition has to be satisfied for asymptotic stability of A.

In this case A for the linear system that we defined if A_lin_syms, and B is B_lin_syms.

But for checking asymptotic stability we will only require A_lin syms.

Q is given to us as I_6

```
Q = eye(6);
```

```
poles_A = eig(A_lin_syms)
```

```
poles_A = 6×1
         0
         0
   -9.1715
   -3.7394
    9.1715
    3.7394
```

```
%P = lyap(A_lin_syms', Q)
% As in MATLAB it's AP+PA'+ Q = 0 solving Lyapunov for this case guves us an error.
```

This happens because A is not asymptotically stable. We can see that all eigenvalues are not in the left hand plane. A stable system would have all eigenvalues with strictly negative real parts.

Problem 2. (4 pts) Design a linear state-feedback controller for the linearized model.

We can proceed with using LQR, it will give us 3 Values, for the gain Matrix K, Solution S of the Riccati equation and the closed loop poles P.

$\dot{x}(t) = \mathrm{Ax}(t) + \mathrm{Bu}(t)$

The Equation for the cost function is given by:

$$J = \int_0^\infty (x^T \mathrm{Qx} + u^T \mathrm{Ru})\, \mathrm{dt}$$

The **optimal control law** is a linear state feedback:

$u(t) = -\mathrm{K}\mathrm{x}(t)$

$K = R^{-1}B^{T}P$

$A^{T}P + \mathrm{PA} - \mathrm{PB}R^{-1}B^{T}P + Q = 0$

Q is given to us. We need to specify our R matrix, in this case I will take it as 1 as we don't want our controller to be aggressive

Let us proceed.

The choice of R and Q depends on the one defining the cost function, for a more aggressive controller we can take R to be small, But in this case I have chosen a moderate value, Q is given to us in the question.

```
R = 1
```

```
R =
1
```

```
[K,S,Poles] = lqr(A_lin_syms, B_lin_syms, Q, R)
```

```
K = 1×6
    1.0000 -268.9918  291.4237    3.3277  -18.4729   48.3751
S = 6×6
10^3 ×
    0.0033   -0.0185    0.0484    0.0050    0.0018    0.0105
   -0.0185    3.2945   -4.2500   -0.0632    0.1701   -0.7503
    0.0484   -4.2500    6.3253    0.1505   -0.1433    1.1696
    0.0050   -0.0632    0.1505    0.0137    0.0044    0.0321
    0.0018    0.1701   -0.1433    0.0044    0.0160   -0.0202
    0.0105   -0.7503    1.1696    0.0321   -0.0202    0.2195
Poles = 6×1 complex
  -0.4648 + 0.3810i
  -0.4648 - 0.3810i
  -3.4074 + 0.0000i
  -4.1313 + 0.0000i
  -8.6088 + 0.0000i
  -9.7720 + 0.0000i
```

```
% Calculate the closed-loop system dynamics
A_cl = A_lin_syms - B_lin_syms * K;
eig(A_cl)
```

```
ans = 6×1 complex
  -9.7720 + 0.0000i
  -8.6088 + 0.0000i
  -4.1313 + 0.0000i
  -3.4074 + 0.0000i
  -0.4648 + 0.3810i
  -0.4648 - 0.3810i
```

```
sys_cl = ss(A_cl, B_lin_syms, C_output, D_sys)
```

```
sys_cl =

  A =
          x1       x2       x3       x4       x5       x6
   x1      0        0        0        1        0        0
   x2      0        0        0        0        1        0
   x3      0        0        0        0        0        1
```

```
   x4  -0.6667    171.2   -194.3   -2.218    12.32   -32.25
   x5   1.333   -293.3    359.1    4.437   -24.63     64.5
   x6       0    -32.7     32.7        0        0        0

  B =
         u1
   x1     0
   x2     0
   x3     0
   x4  0.6667
   x5  -1.333
   x6     0

  C =
      x1  x2  x3  x4  x5  x6
   y1  1   0   0   0   0   0
   y2  0   1   0   0   0   0
   y3  0   0   1   0   0   0

  D =
      u1
   y1  0
   y2  0
   y3  0

Continuous-time state-space model.
Model Properties
```

Problem 3. (4 pts) Find the transfer function of the closed-loop system composed of the linearized model driven by the linear state-feedback controller.

```
tf_cl = tf(sys_cl);
tf_cl(1)
```

```
ans =

      0.6667 s^4 + 5.24e-14 s^3 - 54.5 s^2 - 2.463e-12 s + 427.7
  ---------------------------------------------------------------
  s^6 + 26.85 s^5 + 261.2 s^4 + 1122 s^3 + 2100 s^2 + 1423 s + 427.7

Continuous-time transfer function.
Model Properties
```

```
tf_cl(2)
```

```
ans =

    -1.333 s^4 - 8.628e-16 s^3 + 43.6 s^2 - 1.051e-14 s - 1.21e-27
  ---------------------------------------------------------------
  s^6 + 26.85 s^5 + 261.2 s^4 + 1122 s^3 + 2100 s^2 + 1423 s + 427.7

Continuous-time transfer function.
Model Properties
```

```
tf_cl(3)
```

```
ans =

                 43.6 s^2 - 8.947e-15 s - 3.958e-16
  ---------------------------------------------------------------
  s^6 + 26.85 s^5 + 261.2 s^4 + 1122 s^3 + 2100 s^2 + 1423 s + 427.7
```

```
Continuous-time transfer function.
Model Properties
```

Problem 4. (4 pts) Construct a Lyapunov function for the closed-loop system comprised of the linearized model driven by the state-feedback controller, solve the Lyapunov matrix equation, and check if the equilibrium state of interest of the closed-loop system is asymptotically stable in the sense of Lyapunov.

```
P = lyap(A_cl', Q)
```

```
P = 6×6
    2.7977   -8.9424   27.0644    3.2730    1.2615    6.0087
   -8.9424  131.5370 -276.6926  -21.2280   -5.9323  -57.8914
   27.0644 -276.6926  651.7088   58.4136   18.9727  138.6696
    3.2730  -21.2280   58.4136    6.3746    2.3369   12.7689
    1.2615   -5.9323   18.9727    2.3369    0.9479    4.2268
    6.0087  -57.8914  138.6696   12.7689    4.2268   29.6998
```

```
eig(P)
```

```
ans = 6×1
    0.0285
    0.0740
    0.2391
    1.1650
   14.3165
  807.2427
```

The closed-loop system is asymptotically stable if a unique **positive-definite** matrix P exists.

The eigenvalues of your matrix P are all positive, which means that **P is positive-definite**.

Yes, the equilibrium state of interest of the closed-loop system is **asymptotically stable** in the sense of Lyapunov.

Problem 5. (4 pts) Add an extra input in the double inverted pendulum on a cart (DIPC) non-linear model from the previous FunWork, namely, the torque at the first joint. Thus the system's overall input is

$$u = \begin{bmatrix} u_1 & u_2 \end{bmatrix}^\top,$$

where u_1 is the force applied to the cart and u_2 is the torque applied at the first joint. In summary, we have now a three-output two-input system.

Design a stabilizing state-feedback controller using the linearized model.

Adding an extra input to the initial system which gives torque.

We will now rearrange the previous equations to get this new equation which takes in two inputs and redefine f_sys; and get a new solution for our lagrangian equation.

```
syms u_1 u_2
u_new = [u_1;u_2];
D_new = sym(zeros(3,3));
for i=1:3
  for j=1:3
    D_new(i,j) = simplify(diff(diff(T_total, qd(i)), qd(j)));% Computing D(q)
  end

end
G_new = simplify([ diff(V_total, x1);
      diff(V_total, x2);
      diff(V_total, x3) ]);

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

C_new = 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_new(i,j) = simplify( reshape(c_new(i,j,:),[n,1]).' * qd );
  end
end

H_new = [1 0;
    0 1;
    0 0];
qdd_new = simplify(D_new\(H_new*u_new - C_new*qd-G_new));
X_new = [q;qd];
f_sym_new = [qd; qdd_new]
```

```
f_sym_new =
```

$$\begin{pmatrix} x_4 \\ x_5 \\ x_6 \\ -\frac{280\,u_1 - 981\sin(2\,x_2) + 100\,x_5^2\sin(x_2) + 45\,x_6^2\sin(x_3) - 120\,u_1\cos(\sigma_4)}{20\,\sigma_3} \\ \frac{1200\,u_2\cos(2\,x_3) - 8829\sin(x_2 - 2\,x_3) - 30411\sin(x_2) - 7600\,u_2 + 1575\,x_6^2\sin(x_2 - x_3) + 250\,x_5^2\sin(2\,x_2}{50\,\sigma_3} \\ -\frac{300\sin(x_2 - x_3)\,x_5^2 + 135\sin(\sigma_4)\,x_6^2 - 2943\,\sigma_1 + 2943\sin(x_3) + 200\,u_1\cos(2\,x_2 - x_3) + 40}{75\cos(2\,x_2) + 135\cos(\sigma_4) - 390} \end{pmatrix}$$

where

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

$$\sigma_2 = \cos(x_2 - 2\,x_3)$$

$$\sigma_3 = 5\cos(2\,x_2) + 9\cos(\sigma_4) - 26$$

$$\sigma_4 = 2\,x_2 - 2\,x_3$$

In this system, again we linearize our system around 0, we have our new equations for the linearized matrices. Now we will proceed as we did earlier, we will get the gain matrix by LQR and ensure that poles of $A - \mathrm{BK}$ are negative

```
A_sym_new = simplify(jacobian(f_sym_new, X));
B_sym_new = simplify(jacobian(f_sym_new, u_new));
vars_new = [x1; x2; x3; x4; x5; x6; u_1; u_2];
vals_new = [0; 0; 0; 0; 0; 0; 0; 0];
A_lin_syms_new = double(simplify(subs(A_sym_new, vars_new, vals_new)))
```

```
A_lin_syms_new = 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_new = double(simplify(subs(B_sym_new, vars_new, vals_new)))
```

```
B_lin_syms_new = 6×2
         0         0
         0         0
         0         0
    0.6667   -1.3333
   -1.3333   10.6667
         0   -5.3333
```

```
poles_A_new = eig(A_lin_syms_new)
```

```
poles_A_new = 6×1
         0
         0
   -9.1715
   -3.7394
    9.1715
    3.7394
```

Here we use $R = I_2$ as we need to account for 2 inputs instead of 1 for our LQR; I have multiplied it by 5 to make it less aggressive to minimize the overall cost, the LQR algorithm will find a solution that uses smaller, smoother control inputs.

Also we define a new D for the equation, as the columns of D and B should match.

y = Cx+Du

x_dot = Ax+Bu

B will have 2 columns instead of 1 because of new input

We will now calculate the controller for our system and redesign it with the new input

```
R_new = 5*eye(2)
```

```
R_new = 2×2
     5     0
     0     5
```

```
D_new_sys = zeros(3,2)
```

```
D_new_sys = 3×2
     0     0
     0     0
     0     0
```

```
[K_new,S_new,Poles_new] = lqr(A_lin_syms_new, B_lin_syms_new,Q, R_new)
```

```
K_new = 2×6
    0.0326   -6.1312  -24.0310   -0.0243   -2.7385   -5.8278
   -0.4460    5.6835  -38.3330   -1.7740   -2.3147   -8.6090
S_new = 6×6
10^3 ×
    0.0040    0.0047    0.0183    0.0074    0.0036    0.0057
    0.0047    0.0891    0.1979    0.0173    0.0316    0.0536
    0.0183    0.1979    1.0819    0.0714    0.1258    0.2697
    0.0074    0.0173    0.0714    0.0272    0.0137    0.0223
    0.0036    0.0316    0.1258    0.0137    0.0171    0.0330
    0.0057    0.0536    0.2697    0.0223    0.0330    0.0685
Poles_new = 6×1 complex
  -0.2972 + 0.2721i
  -0.2972 - 0.2721i
  -3.4325 + 0.0000i
  -4.0935 + 0.0000i
  -6.8711 + 0.0000i
 -12.2338 + 0.0000i
```

```
% Calculate the closed-loop system dynamics for the new control inputs
A_cl_new = A_lin_syms_new - B_lin_syms_new * K_new;
eig(A_cl_new)
```

```
ans = 6×1 complex
 -12.2338 + 0.0000i
  -6.8711 + 0.0000i
  -4.0935 + 0.0000i
  -3.4325 + 0.0000i
  -0.2972 + 0.2721i
  -0.2972 - 0.2721i
```

```
sys_cl_new = ss(A_cl_new, B_lin_syms_new, C_output, D_new_sys);
tf_cl_new = tf(sys_cl_new);
P_new = lyap(A_cl_new', Q)
```

```
P_new = 6×6
    3.0992    2.5783    9.5897    4.0026    1.8817    2.9707
    2.5783    5.0504   14.2172    5.6156    2.8098    4.4181
    9.5897   14.2172   79.6163   22.3903   13.2953   22.3230
    4.0026    5.6156   22.3903    8.6700    4.2771    6.8628
    1.8817    2.8098   13.2953    4.2771    2.3936    3.9106
    2.9707    4.4181   22.3230    6.8628    3.9106    6.5214
```

```
% Calculate the eigenvalues of the closed-loop system for the new control inputs
eig(P_new)
```

```
ans = 6×1
    0.0229
    0.1208
    0.6647
    1.3268
    4.3921
   98.8236
```

Eigenvalues of P are positvie definite, and eigenvalues of our A matrix are all in the open left hand plane.

We have designed a new stable system.

```
tf_cl_new(1,1)
```

```
ans =

         0.6667 s^4 + 18.26 s^3 + 112.1 s^2 + 161.5 s + 31.24
  ------------------------------------------------------------------
  s^6 + 27.23 s^5 + 257.9 s^4 + 1049 s^3 + 1756 s^2 + 848.5 s + 191.8

Continuous-time transfer function.
Model Properties
```

```
tf_cl_new(1,2)
```

```
ans =

          -1.333 s^4 - 6.116 s^3 - 9.144 s^2 - 191 s - 427.7
  ------------------------------------------------------------------
  s^6 + 27.23 s^5 + 257.9 s^4 + 1049 s^3 + 1756 s^2 + 848.5 s + 191.8

Continuous-time transfer function.
Model Properties
```

```
tf_cl_new(2,1)
```

```
ans =
```

```
        -1.333 s^4 - 51.76 s^3 - 226.6 s^2 - 123.8 s - 31.11
  ------------------------------------------------------------------
  s^6 + 27.23 s^5 + 257.9 s^4 + 1049 s^3 + 1756 s^2 + 848.5 s + 191.8

Continuous-time transfer function.
Model Properties
```

```
tf_cl_new(2,2)
```

```
ans =

         10.67 s^4 + 41.31 s^3 - 20.78 s^2 + 1.694 s - 2.272
  ------------------------------------------------------------------
  s^6 + 27.23 s^5 + 257.9 s^4 + 1049 s^3 + 1756 s^2 + 848.5 s + 191.8

Continuous-time transfer function.
Model Properties
```

```
tf_cl_new(3,1)
```

```
ans =

            10.15 s^3 + 1.598 s^2 + 5.067e-14 s + 1.174e-14
  ------------------------------------------------------------------
  s^6 + 27.23 s^5 + 257.9 s^4 + 1049 s^3 + 1756 s^2 + 848.5 s + 191.8

Continuous-time transfer function.
Model Properties
```

```
tf_cl_new(3,2)
```

```
ans =

     -5.333 s^4 - 19.39 s^3 - 43.72 s^2 - 7.818e-14 s + 3.334e-27
  ------------------------------------------------------------------
  s^6 + 27.23 s^5 + 257.9 s^4 + 1049 s^3 + 1756 s^2 + 848.5 s + 191.8

Continuous-time transfer function.
Model Properties
```

Problem 6: Use the linearized model to design a Luenberger observer.

We know that for designing a Luenberger observer, we need $A - \mathrm{LC}$ where L is the Luenberger gain. For a stable system, we need the poles of this system to be in the open left hand plane.

$e^{\cdot} = (A - \mathrm{LC})e = A\_\mathrm{Luen}$ (Error dynamics are given by this equation)

Let us first take our new linearized A and C matrices, and check if they are observable

```
Obs = obsv(A_lin_syms_new, 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
```

So we have that they are observable, we will now proceed to design the observe's gain matrix L. The poles for the Luenberger observer should be 2-5 times of that of the controller(Rule of thumb), we can take the new values by multiplying the Poles_new by 4.

```
Luenberger_poles = Poles_new*4
```

```
Luenberger_poles = 6×1 complex
  -1.1890 + 1.0883i
  -1.1890 - 1.0883i
 -13.7301 + 0.0000i
 -16.3740 + 0.0000i
 -27.4842 + 0.0000i
 -48.9351 + 0.0000i
```

```
L = place(A_lin_syms_new', C_output', Luenberger_poles)' %Transposing again to get L
```

```
L = 6×3
    2.3780         0         0
         0   43.8583         0
         0         0   62.6652
    2.5981   -8.1750         0
         0  515.4278  -29.4300
         0  -32.7000  704.5854
```

```
A_Luen = A_lin_syms_new-L*C_output
```

```
A_Luen = 6×6
   -2.3780         0         0    1.0000         0         0
         0  -43.8583         0         0    1.0000         0
         0         0  -62.6652         0         0    1.0000
   -2.5981         0         0         0         0         0
         0 -450.0278         0         0         0         0
         0         0 -671.8854         0         0         0
```

```
eig(A_Luen)
```

```
ans = 6×1 complex
  -1.1890 + 1.0883i
  -1.1890 - 1.0883i
 -48.9351 + 0.0000i
 -13.7301 + 0.0000i
 -16.3740 + 0.0000i
 -27.4842 + 0.0000i
```

All eigenvalues of the Luenberger Observer are in the left hand plane, we have successfully defined it.

Problem 7

Problem 7. (4 pts) Construct a Lyapunov function for the closed-loop system composed of the linearized model driven by the combined controller-observer compensator, solve the Lyapunov matrix equation, using $Q = I_{12}$ and check if the equilibriu state of interest of the closed-loop system is asymptotically stable in the sense of Lyapunov.

After getting the Luenberger observer, we now need to construct a Lyapunov function for the closed loop system.

New Q = I_{12}

For our Combined controller observer compensator we take L and K together for the new system.

The new system will be given by the following equation:

$$\begin{bmatrix} \dot{\boldsymbol{x}}(t) \\ \dot{\tilde{\boldsymbol{x}}}(t) \end{bmatrix} = \begin{bmatrix} \boldsymbol{A} & -\boldsymbol{BK} \\ \boldsymbol{LC} & \boldsymbol{A}-\boldsymbol{LC}-\boldsymbol{BK} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}(t) \\ \tilde{\boldsymbol{x}}(t) \end{bmatrix} + \begin{bmatrix} \boldsymbol{B} \\ \boldsymbol{B} \end{bmatrix} \boldsymbol{v}(t)$$

$$\boldsymbol{y}(t) = \begin{bmatrix} \boldsymbol{C} & \boldsymbol{O} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}(t) \\ \tilde{\boldsymbol{x}}(t) \end{bmatrix}$$

We need to create new A,B,C,D matrices for the system following this:

```
A_cl_new = A_lin_syms_new - B_lin_syms_new * K_new;
A_Luen = A_lin_syms_new-L*C_output
```

```
A_Luen = 6×6
   -2.3780         0         0    1.0000         0         0
         0  -43.8583         0         0    1.0000         0
         0         0  -62.6652         0         0    1.0000
   -2.5981         0         0         0         0         0
         0 -450.0278         0         0         0         0
         0         0 -671.8854         0         0         0
```

```
BK = B_lin_syms_new*K_new;
```

```
O = 6×6
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
```

```
A_combined = [A_lin_syms_new, -B_lin_syms_new*K_new;
              L*C_output, (A_lin_syms_new - B_lin_syms_new*K_new - L*C_output)];

Q_new = eye(12)
```

```
Q_new = 12×12
     1     0     0     0     0     0     0     0     0     0     0     0
     0     1     0     0     0     0     0     0     0     0     0     0
     0     0     1     0     0     0     0     0     0     0     0     0
     0     0     0     1     0     0     0     0     0     0     0     0
     0     0     0     0     1     0     0     0     0     0     0     0
     0     0     0     0     0     1     0     0     0     0     0     0
     0     0     0     0     0     0     1     0     0     0     0     0
     0     0     0     0     0     0     0     1     0     0     0     0
     0     0     0     0     0     0     0     0     1     0     0     0
     0     0     0     0     0     0     0     0     0     1     0     0
     0     0     0     0     0     0     0     0     0     0     1     0
     0     0     0     0     0     0     0     0     0     0     0     1
     ⋮
```

```
P_combined = lyap(A_combined', Q_new);
```

```
eig(P_combined)
```

```
ans = 12×1
    0.0060
    0.0119
    0.0897
    0.2467
    0.3778
    0.5498
    1.3712
    4.3882
    8.3479
   14.4907
  133.6257
  500.7602
      ⋮
```

All eigenvalues of P_combined are positive the combined controller-observer system is **asymptotically stable** in the sense of Lyapunov

Problem 8

Finding transfer function of this closed loop system. We need to scale the matrices, B,C and D to account for the new dimesions of the system.

```
B_combined = [B_lin_syms_new; B_lin_syms_new]
```

```
B_combined = 12×2
         0         0
         0         0
         0         0
    0.6667   -1.3333
   -1.3333   10.6667
         0   -5.3333
         0         0
         0         0
         0         0
    0.6667   -1.3333
   -1.3333   10.6667
         0   -5.3333
      ⋮
```

```
C_combined = [C_output, zeros(3,6)]
```

```
C_combined = 3×12
     1     0     0     0     0     0     0     0     0     0     0     0
     0     1     0     0     0     0     0     0     0     0     0     0
     0     0     1     0     0     0     0     0     0     0     0     0
```

```
D_combined = zeros(3,2)
```

```
D_combined = 3×2
     0     0
     0     0
     0     0
```

```
sys_combined = ss(A_combined, B_combined, C_combined, D_combined)
```

```
sys_combined =

  A =
           x1       x2       x3       x4       x5       x6       x7       x8       x9      x10      x11      x12
  x1        0        0        0        1        0        0        0        0        0        0        0        0
  x2        0        0        0        0        1        0        0        0        0        0        0        0
  x3        0        0        0        0        0        1        0        0        0        0        0        0
  x4        0   -8.175        0        0        0        0  -0.6164    11.67   -35.09   -2.349   -1.261   -7.593
  x5        0     65.4   -29.43        0        0        0    4.801    -68.8    376.8    18.89    21.04    84.06
  x6        0    -32.7     32.7        0        0        0   -2.379    30.31   -204.4   -9.461   -12.34   -45.91
  x7    2.378        0        0        0        0        0   -2.378        0        0        1        0        0
  x8        0    43.86        0        0        0        0        0   -43.86        0        0        1        0
  x9        0        0    62.67        0        0        0        0        0   -62.67        0        0        1
  x10   2.598   -8.175        0        0        0        0   -3.214    11.67   -35.09   -2.349   -1.261   -7.593
  x11       0    515.4   -29.43        0        0        0    4.801   -518.8    376.8    18.89    21.04    84.06
  x12       0    -32.7    704.6        0        0        0   -2.379    30.31   -876.3   -9.461   -12.34   -45.91

  B =
          u1      u2
  x1       0       0
  x2       0       0
  x3       0       0
  x4   0.6667  -1.333
  x5   -1.333   10.67
  x6       0  -5.333
  x7       0       0
  x8       0       0
  x9       0       0
  x10  0.6667  -1.333
  x11  -1.333   10.67
  x12      0  -5.333

  C =
       x1   x2   x3   x4   x5   x6   x7   x8   x9  x10  x11  x12
  y1    1    0    0    0    0    0    0    0    0    0    0    0
  y2    0    1    0    0    0    0    0    0    0    0    0    0
  y3    0    0    1    0    0    0    0    0    0    0    0    0

  D =
      u1  u2
  y1   0   0
  y2   0   0
  y3   0   0

Continuous-time state-space model.
Model Properties
```

```
% Calculate the transfer function for the combined system
sys_combined_min = minreal(sys_combined, 1e-6);
```

```
6 states removed.
```

```
tf_combined_min = tf(sys_combined_min);

tf_combined_min(1,1)
```

```
ans =

         0.6667 s^4 + 18.26 s^3 + 112.1 s^2 + 161.5 s + 31.24
  ------------------------------------------------------------------
  s^6 + 27.23 s^5 + 257.9 s^4 + 1049 s^3 + 1756 s^2 + 848.5 s + 191.8
```

```
Continuous-time transfer function.
Model Properties
```

```
tf_combined_min(1,2)
```

```
ans =

          -1.333 s^4 - 6.116 s^3 - 9.144 s^2 - 191 s - 427.7
  ------------------------------------------------------------------
  s^6 + 27.23 s^5 + 257.9 s^4 + 1049 s^3 + 1756 s^2 + 848.5 s + 191.8

Continuous-time transfer function.
Model Properties
```

```
tf_combined_min(2,1)
```

```
ans =

         -1.333 s^4 - 51.76 s^3 - 226.6 s^2 - 123.8 s - 31.11
  ------------------------------------------------------------------
  s^6 + 27.23 s^5 + 257.9 s^4 + 1049 s^3 + 1756 s^2 + 848.5 s + 191.8

Continuous-time transfer function.
Model Properties
```

```
tf_combined_min(2,2)
```

```
ans =

          10.67 s^4 + 41.31 s^3 - 20.78 s^2 + 1.694 s - 2.272
  ------------------------------------------------------------------
  s^6 + 27.23 s^5 + 257.9 s^4 + 1049 s^3 + 1756 s^2 + 848.5 s + 191.8

Continuous-time transfer function.
Model Properties
```

```
tf_combined_min(3,1)
```

```
ans =

            10.15 s^3 + 1.598 s^2 + 2.591e-12 s + 5.296e-13
  ------------------------------------------------------------------
  s^6 + 27.23 s^5 + 257.9 s^4 + 1049 s^3 + 1756 s^2 + 848.5 s + 191.8

Continuous-time transfer function.
Model Properties
```

```
tf_combined_min(3,2)
```

```
ans =

     -5.333 s^4 - 19.39 s^3 - 43.72 s^2 - 9.393e-12 s - 4.184e-12
  ------------------------------------------------------------------
  s^6 + 27.23 s^5 + 257.9 s^4 + 1049 s^3 + 1756 s^2 + 848.5 s + 191.8

Continuous-time transfer function.
Model Properties
```

Problem 9, Part 1. (4 pts) Implement and perform animation of the closed-loop system composed of the combined controller-observer compensator driving the DIPC nonlinear model.

```
vars_for_f = {x1, x2, x3, x4, x5, x6, u_1, u_2};
nonlinear_dynamics_fun = matlabFunction(f_sym_new, 'Vars', vars_for_f);
```

```
Running simulation...
Simulation complete.
```

```


t_span = [0 20];

% Initial condition for the plant (e.g., slightly disturbed from vertical)
x_plant_0 = [1; 0.2; -0.2; 0; 0; 0];

% Initial condition for the observer (starts at zero, assuming no initial info)
x_observer_0 = zeros(6,1);

% Combine into a single initial state vector for the simulation
X_sim_0 = [x_plant_0; x_observer_0];



```

```
Running simulation...
```

```
[T, X_out] = ode45(@(t, X_sim) dynamics_with_compensator(t, X_sim, ...
    K_new, L, A_lin_syms_new, B_lin_syms_new, C_output, nonlinear_dynamics_fun), ...
    t_span, X_sim_0);
disp('Simulation complete.');
```

```
Simulation complete.
```

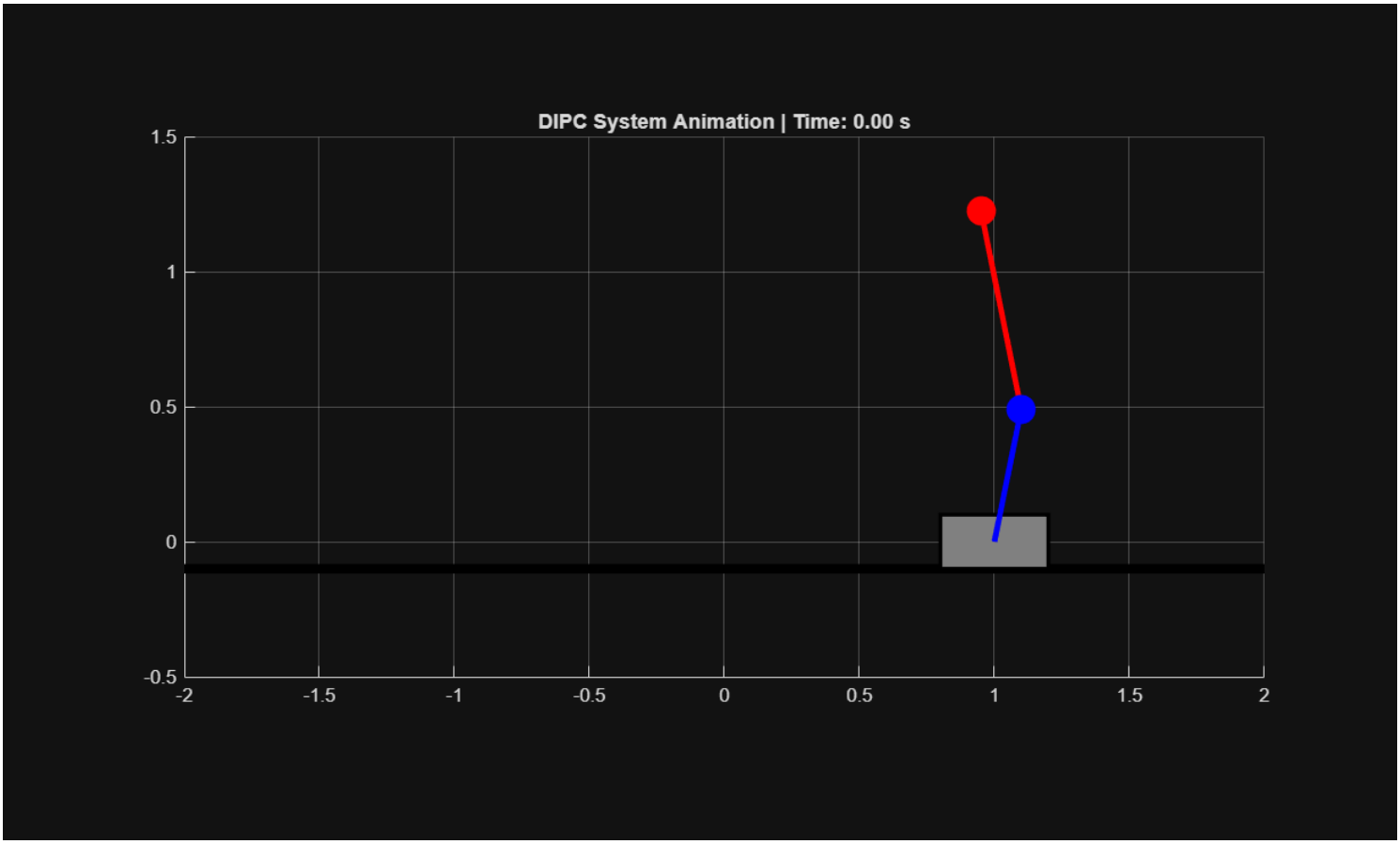
```


function dXdt = dynamics_with_compensator(t, X_sim, K, L, A, B, C, nonlin_fun)

    % 1. Unpack the state vector
    x_plant    = X_sim(1:6);
    x_observer = X_sim(7:12);

    % 2. Compute the control input using the OBSERVED states
    u = -K * x_observer;

    % 3. Compute the plant dynamics (CORRECTED CALL)
    % Pass the elements of the vectors individually as required by the function
handle.
    dx_plant = nonlin_fun(x_plant(1), x_plant(2), x_plant(3), x_plant(4),
x_plant(5), x_plant(6), u(1), u(2));

    % 4. Compute the observer dynamics (CORRECTED EQUATION)
    y_measured  = C * x_plant;
```

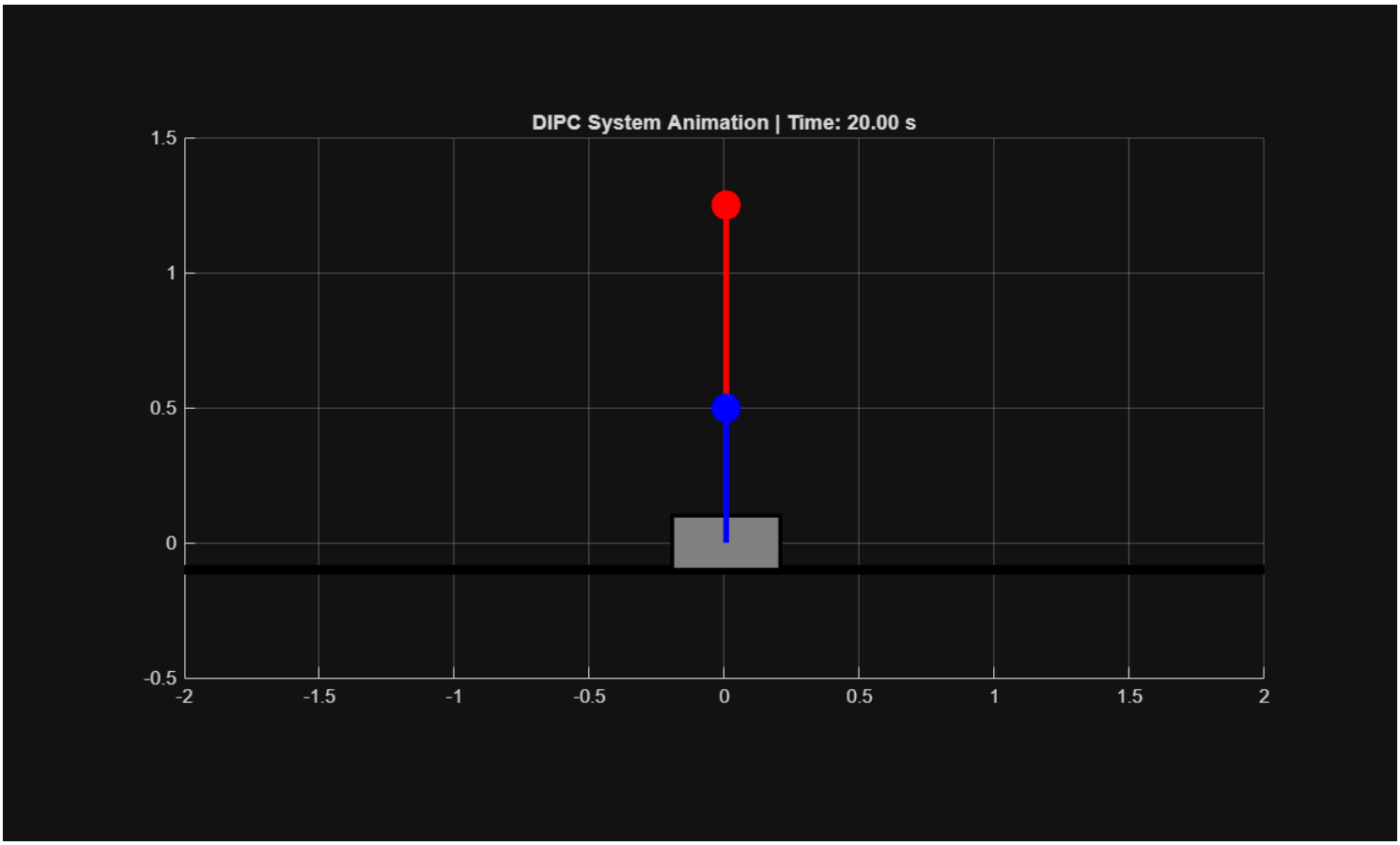
```
    y_estimated = C * x_observer; % This was the missing part
    dx_observer = A * x_observer + B * u + L * (y_measured - y_estimated);

    % 5. Combine the derivatives
    dXdt = [dx_plant; dx_observer];
end
% Extract the plant states for animation
x1_sim = X_out(:, 1);
x2_sim = X_out(:, 2);
x3_sim = X_out(:, 3);


%% 4. Create and Save the Animation
disp('Creating animation...');
```

```
Creating animation...
```

```
% Setup video writer
video = VideoWriter('dipc_animation.avi', 'Motion JPEG AVI');
video.FrameRate = 30; % Adjust for desired playback speed
open(video);

% Setup figure
figure('Name', 'DIPC Animation', 'Position', [100 100 1000 600]);
ax = gca;
grid on;
hold on;

% Animation loop
for i = 1:length(T)
    x_c = x1_sim(i); % cart position
    th1 = x2_sim(i); % pendulum 1 angle
    th2 = x3_sim(i); % pendulum 2 angle

    % Calculate coordinates
    cart_pos = [x_c, 0];
    bob1_pos = cart_pos + [l1*sin(th1), l1*cos(th1)];
    bob2_pos = bob1_pos + [l2*sin(th2), l2*cos(th2)];

    % Clear previous frame
    cla(ax);

    % Set plot limits
    axis(ax, [-2 2 -0.5 1.5]);
    ax.DataAspectRatio = [1 1 1];

    % Draw ground
    plot([-5 5], [-0.1 -0.1], 'k', 'LineWidth', 5);

    % Draw cart
```

```
    cart_width = 0.4;
    cart_height = 0.2;
    rectangle('Position', [cart_pos(1)-cart_width/2, cart_pos(2)-cart_height/2,
cart_width, cart_height], ...
              'FaceColor', [0.5 0.5 0.5], 'EdgeColor', 'k', 'LineWidth', 2);

    % Draw pendulum links
    plot([cart_pos(1), bob1_pos(1)], [cart_pos(2), bob1_pos(2)], 'b', 'LineWidth',
3);
    plot([bob1_pos(1), bob2_pos(1)], [bob1_pos(2), bob2_pos(2)], 'r', 'LineWidth',
3);

    % Draw pendulum bobs
    plot(bob1_pos(1), bob1_pos(2), 'bo', 'MarkerFaceColor', 'b', 'MarkerSize', 15);
    plot(bob2_pos(1), bob2_pos(2), 'ro', 'MarkerFaceColor', 'r', 'MarkerSize', 15);

    title(sprintf('DIPC System Animation | Time: %.2f s', T(i)));
    % Force the figure to update
    drawnow;

    % PAUSE HERE: Give the renderer time to catch up
    pause(0.05); % A small pause of 0.01 seconds is usually enough

    % Now, capture the frame
    frame = getframe(gcf);
    writeVideo(video, frame);
end
```


DIPC System Animation | Time: 0.00 s
1.5
1
0.5
0
-0.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2


DIPC System Animation | Time: 20.00 s
1.5
1
0.5
0
-0.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

```
% Close the video file
close(video);
disp('Animation video "dipc_animation.avi" saved successfully!');
```

```
Animation video "dipc_animation.avi" saved successfully!
```

$$=$$

$$\begin{pmatrix} x_4 \\ x_5 \\ x_6 \\ -\dfrac{280\,u_1 - 981\sin(2\,x_2) + 100\,x_5^2\sin(x_2) + 45\,x_6^2\sin(x_3) - 120\,u_1\cos(\sigma_4) - 560\,u_2\cos(x_2) + 45\,x_6^2\,\sigma_1 + 240\,u_2\,\sigma_2}{20\,\sigma_3} \\ \dfrac{\text{:os}(2\,x_3) - 8829\sin(x_2 - 2\,x_3) - 30411\sin(x_2) - 7600\,u_2 + 1575\,x_6^2\sin(x_2 - x_3) + 250\,x_5^2\sin(2\,x_2) + 1400\,u_1\cos(x_2) + 450\,x_5^2\sin(\sigma_4) - 600\,u_1\,\sigma_2 + 225\,x_6^2\sin(x_2 + x_3)}{50\,\sigma_3} \\ -\dfrac{300\sin(x_2 - x_3)\,x_5^2 + 135\sin(\sigma_4)\,x_6^2 - 2943\,\sigma_1 + 2943\sin(x_3) + 200\,u_1\cos(2\,x_2 - x_3) + 400\,u_2\cos(x_2 + x_3) - 200\,u_1\cos(x_3) - 1360\,u_2\cos(x_2 - x_3)}{75\cos(2\,x_2) + 135\cos(\sigma_4) - 390} \end{pmatrix}$$

$2\,x_2 - x_3)$

$(x_2 - 2\,x_3)$

$\text{os}(2\,x_2) + 9\cos(\sigma_4) - 26$

$: - 2\,x_3$