1) To practice your vector-matrix factorization and manipulation, please repeat the derivations on page 8 of Handout 5 as follows. Consider the state space representation of a general LTI system below:

$$\begin{cases} \dfrac{d\mathbf{x}}{dt} = \mathbf{Ax} + \mathbf{Bu} \\ \mathbf{y} = \mathbf{Cx} + \mathbf{Du} \end{cases}$$

Where $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$ are matrices and $\mathbf{x}, \mathbf{u}, \mathbf{y}$ are column vectors.

a. Identify the state vector, input vector and output vector.
   Answer: State vector: $\mathbf{x}$; Input vector: $\mathbf{u}$.

b. Use matrix multiplication inner dimension compatibility rule to find the dimensions (no. of rows × no. of columns) of the matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$ when dimensions of $\mathbf{x}, \mathbf{u}, \mathbf{y}$ are n, m, p; respectively.
   Answer: Each matrix has as many rows as the entity on the left-hand-side of the equal sign and as many columns as the multiplying vector. Hence:
   A: dx/dt size = n (same as x), x size = n → A is n X n.
   B: dx/dt size = n, u size = m → B is n X m.
   C: y size = p, x size = n → C is p X n.
   D: y size = p, u size = m → D is p X m.

c. How many *scalar* state variables, inputs and outputs does the system have given the dimensions in part (b) above?
   Answer: x is of size n hence n elements in x → n scalar state variables, m elements in u → m scalar inputs, p elements in y → p scalar outputs.

d. Take the Laplace transform of the above vector/matrix differential equations as done in Handout 5, where the initial condition for $\mathbf{x}$ is $\mathbf{x}_0$.
   Answer: $\mathcal{L}\begin{bmatrix} \frac{d\mathbf{x}}{dt} = \mathbf{Ax} + \mathbf{Bu} \\ \mathbf{y} = \mathbf{Cx} + \mathbf{Du} \end{bmatrix} = \begin{cases} s\boldsymbol{X}(s) - \mathbf{x}_0 = \mathbf{A}\boldsymbol{X}(s) + \mathbf{B}\boldsymbol{U}(s) \\ \boldsymbol{Y}(s) = \mathbf{C}\boldsymbol{X}(s) + \mathbf{D}\boldsymbol{U}(s) \end{cases} \Rightarrow \begin{cases} s\boldsymbol{X}(s) - \mathbf{A}\boldsymbol{X}(s) = \mathbf{B}\boldsymbol{U}(s) + \mathbf{x}_0 \\ \boldsymbol{Y}(s) = \mathbf{C}\boldsymbol{X}(s) + \mathbf{D}\boldsymbol{U}(s) \end{cases}$

$$\Rightarrow \begin{cases} (s\mathbf{I} - \mathbf{A})\boldsymbol{X}(s) = \mathbf{B}\boldsymbol{U}(s) + \mathbf{x}_0 \\ \boldsymbol{Y}(s) = \mathbf{C}\boldsymbol{X}(s) + \mathbf{D}\boldsymbol{U}(s) \end{cases} \Rightarrow \begin{cases} \boldsymbol{X}(s) = (s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}\,\boldsymbol{U}(s) + (s\mathbf{I} - \mathbf{A})^{-1}\mathbf{x}_0 \\ \boldsymbol{Y}(s) = \mathbf{C}\boldsymbol{X}(s) + \mathbf{D}\boldsymbol{U}(s) \end{cases}$$

e. As done in the handout, find an expression for output $\boldsymbol{Y}(s)$ in terms of input $\boldsymbol{U}(s)$ and $\mathbf{x_0}$.
   Substitute $\boldsymbol{X}(s)$ from 1st equation into 2nd equation to find $\boldsymbol{Y}(s)$:

$$\begin{cases} \boldsymbol{X}(s) = (s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}\,\boldsymbol{U}(s) + (s\mathbf{I} - \mathbf{A})^{-1}\mathbf{x}_0 \\ \boldsymbol{Y}(s) = \mathbf{C}[(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}\,\boldsymbol{U}(s) + (s\mathbf{I} - \mathbf{A})^{-1}\mathbf{x}_0] + \mathbf{D}\boldsymbol{U}(s) \end{cases}$$

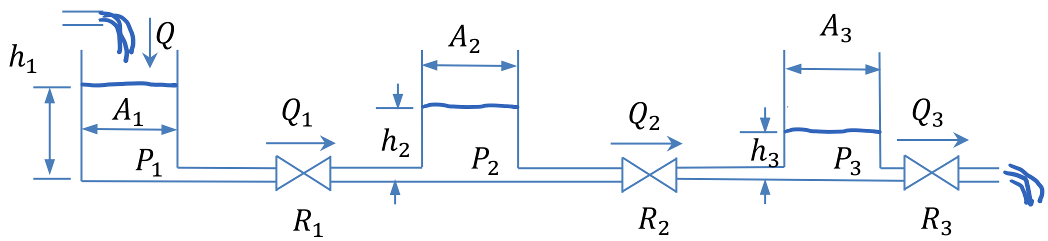
   Right factor $\boldsymbol{U}(s)$:

$$\begin{cases} \boldsymbol{X}(s) = (s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}\,\boldsymbol{U}(s) + (s\mathbf{I} - \mathbf{A})^{-1}\mathbf{x}_0 \\ \boldsymbol{Y}(s) = \{\mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} + \mathbf{D}\}\boldsymbol{U}(s) + \mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{x}_0 \end{cases}$$

f. Write the expression for the matrix of transfer functions. (Recall transfer functions are defined when all initial conditions are zero.)
   Matrix of transfer functions $\mathbf{G}(\boldsymbol{s})$ is the expression multiplying $\boldsymbol{U}(s)$ in the 2nd equation:

$$\mathbf{G}(\boldsymbol{s}) = \mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} + \mathbf{D}$$

2) **Derivation of system state space equations by computer algebra** (symbolic manipulations in MATLAB): The three tank system below is slightly different from the one discussed in Handout 5 in that the exit pressures at the linear flow resistances for the first two tanks are not zero. Recall that the equation for a linear flow resistance is: $\Delta p = R\, q$ where Δp is the pressure difference across the flow resistance and q is the volume flow rate through the resistance. The cross sectional areas of the tanks are A_1, A_2, A_3; the heights of the liquid in the tanks are h_1, h_2, h_3 and the pressures at the bottoms of the tanks are P_1, P_2, P_3. The linear flow resistances at the tank exits are R_1, R_2, R_3 and the volume flow rate through them are Q_1, Q_2, Q_3 respectively. An input flow rate equal to Q enters the first tank, hence the input $\mathbf{u}$ = Q. The system outputs are $\mathbf{y} = \begin{bmatrix} Q_1 \\ Q_2 \\ Q_3 \end{bmatrix}$ and the state variables are the liquid heights $\mathbf{x} = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix}$.



a. Use MATLAB symbolic package to carry out the derivations to find the above system equation in the following form (i.e. find the $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$ matrices symbolically; see Hint on next page):

$$\begin{cases} \dfrac{d\mathbf{x}}{dt} = \mathbf{Ax} + \mathbf{Bu} \\ \mathbf{y} = \mathbf{Cx} + \mathbf{Du} \end{cases}$$

b. Using MATLAB "`subs`" command find numerical expressions for $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$ given: $R_1 = R_2 = R_3 = 10^7\ (\mathrm{Pa/m^3}/s)$; $A_1 = A_2 = A_3 = 0.1\ (\mathrm{m^2})$; density of liquid $\rho = 1000\ (\mathrm{kg/m^3})$; acceleration of gravity $g = 9.81\ (\mathrm{m/s^2})$.

c. Find the matrix of transfer functions using MATLAB. (See Handout 7 equation 7 – 7.)

d. Given $Q =$ step input of magnitude $0.0005\ (\mathrm{m^3}/s)$ and 0 initial conditions (i.e. no liquid in tanks at $t = 0$), use MATLAB symbolic commands to find system outputs vs. time: $\mathbf{y}(t) = \begin{bmatrix} Q_1(t) \\ Q_2(t) \\ Q_3(t) \end{bmatrix}$.

(Hint: Use MATLAB example on page 9 of Handout 5 as a guide).

e. What are the steady state values of h_1, h_2, h_3 (i.e. tank liquid heights when $t \to \infty$)? (Hint: Steady state means no change, i.e. all derivatives are zero, hence solve $0 = \mathbf{Ax}_{ss} + \mathbf{Bu}$ for $\mathbf{x}_{ss}$. The solution is: $\mathbf{x}_{ss} = -\mathbf{A}^{-1}\mathbf{Bu}$. Use MATLAB to do the computation.)

Answer: Note that the equation of liquid height for a tank is

$$\frac{dh}{dt} = A(Q_{in} - Q_{out}; \quad Q_{out} = \frac{1}{R}\Delta P; \quad \Delta P = \rho g h - P_{exit}$$

MATLAB:

Part a:

```
clear                                % Clear all variables to avoid stale data.

% Define symbols for all variables:
syms h1 h2 h3 p1 p2 p3 q q1 q2 q3 r1 r2 r3 a1 a2 a3 rho g

% Also define symbols for time derivatives of h1, h2, h3:
syms dh1dt dh2dt dh3dt

% Enter expressions for pressures at the bottom of tanks:
p1 = rho * g * h1
p2 = rho * g * h2
p3 = rho * g * h3

% Enter expressions for flow rates through the resistances:
q1 = (p1 - p2)/r1
q2 = (p2 - p3)/r2
q3 = (p3 - 0)/r3

% MATLAB does substitutions for you, no need to do any derivation by hand.
% But remember: MATLAB would not do back substitutions; that is: x = a; a = 1 % if now ask for value of
x it would still return x = a.

% Enter expressions for the time derivatives of liquid heights in the tanks:
dh1dt = (q - q1)/a1           % Recall dh/dt = (Qin – Qout)/Area
dh2dt = (q1 - q2)/a2
dh3dt = (q2 - q3)/a3

% Define column vectors: state x, derivative dx/dt, output y and input u:

x = [h1; h2; h3]
y = [q1; q2; q3]
u = q

dxdt = [dh1dt; dh2dt; dh3dt]

dxdt = expand(dxdt)           % “expand” --> Multiply out.

% At this point you are ready to extract A, B, C, D matrices. Easiest way is
% to use “jacobian(g, h)” command where g and h are column vectors. jacobian
% takes all possible partial derivatives of the elements of its 1st argument
% g with respect to elements of the second argument h and hence extracts
% various coefficients of variables such as h1,h2,h3,q and stacks them into
% a matrix. “jacobian” is also used to linearize nonlinear systems and you
% will encounter it later in the coming weeks.

AA = jacobian(dxdt, x)        % When x = state, u = input and y = output,
BB = jacobian(dxdt, u)        % <-- these 4 commands always result A,B,C,D.
CC = jacobian(y, x)
DD = jacobian(y, u)
```

```
p1 =

g*h1*rho


p2 =

g*h2*rho


p3 =

g*h3*rho


q1 =

(g*h1*rho - g*h2*rho)/r1


q2 =

(g*h2*rho - g*h3*rho)/r2


q3 =

(g*h3*rho)/r3


dh1dt =

(q - (g*h1*rho - g*h2*rho)/r1)/a1


dh2dt =

((g*h1*rho - g*h2*rho)/r1 - (g*h2*rho - g*h3*rho)/r2)/a2


dh3dt =

((g*h2*rho - g*h3*rho)/r2 - (g*h3*rho)/r3)/a3


x =

 h1
 h2
 h3
```

```
y =

 (g*h1*rho - g*h2*rho)/r1
 (g*h2*rho - g*h3*rho)/r2
            (g*h3*rho)/r3


u =

q


dxdt =

                        (q - (g*h1*rho - g*h2*rho)/r1)/a1
 ((g*h1*rho - g*h2*rho)/r1 - (g*h2*rho - g*h3*rho)/r2)/a2
            ((g*h2*rho - g*h3*rho)/r2 - (g*h3*rho)/r3)/a3


dxdt =

                                    q/a1 - (g*h1*rho)/(a1*r1) + (g*h2*rho)/(a1*r1)
 (g*h1*rho)/(a2*r1) - (g*h2*rho)/(a2*r1) - (g*h2*rho)/(a2*r2) + (g*h3*rho)/(a2*r2)
                      (g*h2*rho)/(a3*r2) - (g*h3*rho)/(a3*r2) - (g*h3*rho)/(a3*r3)


AA =

[ -(g*rho)/(a1*r1),                     (g*rho)/(a1*r1),                                  0]
[  (g*rho)/(a2*r1), - (g*rho)/(a2*r1) - (g*rho)/(a2*r2),                    (g*rho)/(a2*r2)]
[                0,                     (g*rho)/(a3*r2), - (g*rho)/(a3*r2) - (g*rho)/(a3*r3)]


BB =

 1/a1
    0
    0


CC =

[ (g*rho)/r1, -(g*rho)/r1,           0]
[          0,  (g*rho)/r2, -(g*rho)/r2]
[          0,           0,  (g*rho)/r3]


DD =

 0
 0
 0
```

Part b:

Numerical Substitution: r1=r2=r3=1e7; a1=a2=a3=.1; rho=1000; g=9.81 “subs” command uses curly brackets, be careful of the variables order:

```
A = subs(AA, {r1,r2,r3,a1,a2,a3,rho,g}, {1e7,1e7,1e7,.1,.1,.1,1000,9.81});
A = double(A)  % Convert answer to matrix of double precision numbers.

B = subs(BB, {r1,r2,r3,a1,a2,a3,rho,g}, {1e7,1e7,1e7,.1,.1,.1,1000,9.81});
B = double(B)  % Convert answer to matrix of double precision numbers.

C = subs(CC, {r1,r2,r3,a1,a2,a3,rho,g}, {1e7,1e7,1e7,.1,.1,.1,1000,9.81});
C = double(C)  % Convert answer to matrix of double precision numbers.

D = subs(DD, {r1,r2,r3,a1,a2,a3,rho,g}, {1e7,1e7,1e7,.1,.1,.1,1000,9.81});
D = double(D)  % Convert answer to matrix of double precision numbers.
```

```
A =

   -0.0098    0.0098         0
    0.0098   -0.0196    0.0098
         0    0.0098   -0.0196


B =

    10
     0
     0


C =

   1.0e-03 *

    0.9810   -0.9810         0
         0    0.9810   -0.9810
         0         0    0.9810


D =

     0
     0
     0
```

Part c:

Transfer functions matrix:

```
syms s t
Gs = C * (s * eye(3) - A)^-1 * B + D;
Gs = simplify(Gs);
vpa(Gs,4)

% Alternative method using control system toolbox:
sys = ss(A,B,C,D)       % define state space system.
Gofs = tf(sys)
```

```
ans =    (Transfer Function Matrix, divide numerator denominator by 1.0e15.)

 (981.0*(1.0e10*s^2 + 2.943e8*s + 9.624e5))/(1.0e15*s^3 + 4.905e13*s^2 + 5.774e11*s + 9.441e8)
                (9.624e5*(1.0e5*s + 981.0))/(1.0e15*s^3 + 4.905e13*s^2 + 5.774e11*s + 9.441e8)
                                    9.441e8/(1.0e15*s^3 + 4.905e13*s^2 + 5.774e11*s + 9.441e8)


sys =

  A =
             x1         x2         x3
   x1  -0.00981    0.00981          0
   x2   0.00981   -0.01962    0.00981
   x3         0    0.00981   -0.01962

  B =
       u1
   x1  10
   x2   0
   x3   0

  C =
              x1          x2          x3
   y1   0.000981   -0.000981           0
   y2          0    0.000981   -0.000981
   y3          0           0    0.000981

  D =
       u1
   y1   0
   y2   0
   y3   0

Continuous-time state-space model.
```

```
Gofs =

  From input to output...
               0.00981 s^2 + 0.0002887 s + 9.441e-07
   1:  -----------------------------------------
       s^3 + 0.04905 s^2 + 0.0005774 s + 9.441e-07

                      9.624e-05 s + 9.441e-07
   2:  -----------------------------------------
       s^3 + 0.04905 s^2 + 0.0005774 s + 9.441e-07

                             9.441e-07
   3:  -----------------------------------------
       s^3 + 0.04905 s^2 + 0.0005774 s + 9.441e-07

Continuous-time transfer function.

Answer: transfer function matrix:
```

$$G(s) = \begin{bmatrix} \dfrac{0.00981\,\mathrm{s}^2 + 0.0002887\,\mathrm{s} + 9.441\times10^{-07}}{\mathrm{s}^3 + 0.04905\,\mathrm{s}^2 + 0.0005774\,\mathrm{s} + 9.441\times10^{-07}} \\ \dfrac{9.624\mathrm{e}-05\,\mathrm{s} + 9.441\times10^{-07}}{\mathrm{s}^3 + 0.04905\,\mathrm{s}^2 + 0.0005774\,\mathrm{s} + 9.441\times10^{-07}} \\ \dfrac{9.441\times10^{-07}}{\mathrm{s}^3 + 0.04905\,\mathrm{s}^2 + 0.0005774\,\mathrm{s} + 9.441\times10^{-07}} \end{bmatrix}$$

Part d:

Step response:

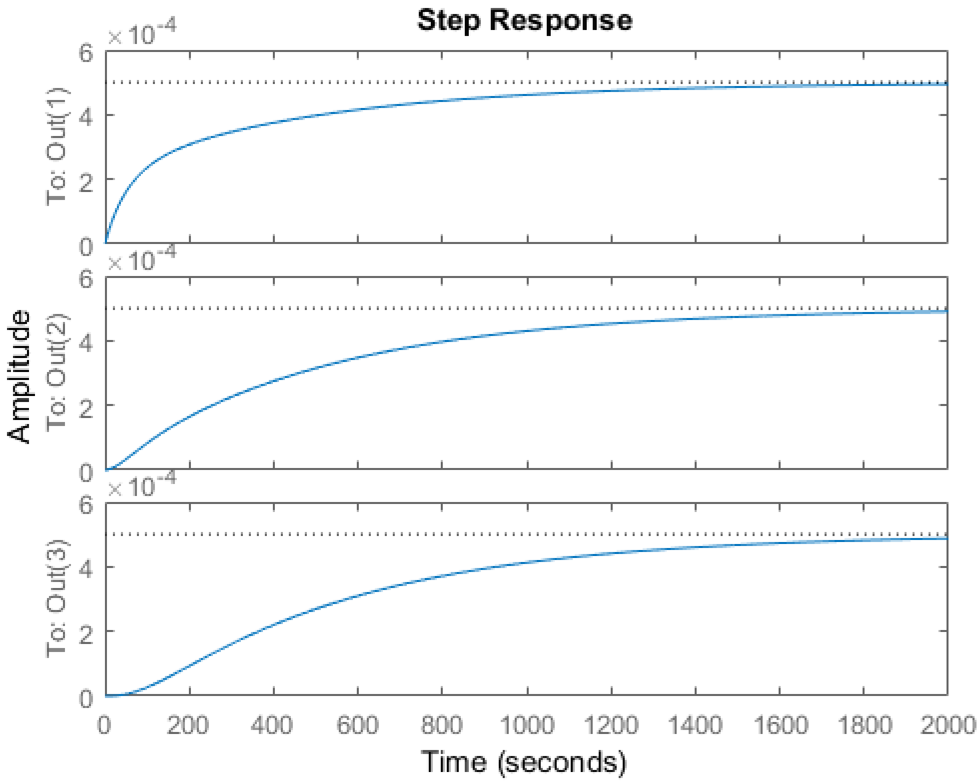
```
Qs = 0.0005 / s          % Input step of magnitude 0.0005.

Ys = Gs * Qs;
Ys = simplify(Ys);
yt = ilaplace(Ys);
yt = vpa(yt,4)

% Plots using control system toolbox:
step(0.0005 * Gofs, 2000)
```

```
Qs =

1/(2000*s)


yt =
```

0.0005 - 0.0001746*exp(-0.01525*t) - 0.0002716*exp(-0.001943*t) - 5.379e-5*exp(-0.03185*t) $= Q_1$

6.707e-5*exp(-0.03185*t) - 7.772e-5*exp(-0.01525*t) - 0.0004893*exp(-0.001943*t) + 0.0005 $= Q_2$

0.0001401*exp(-0.01525*t) - 2.985e-5*exp(-0.03185*t) - 0.0006102*exp(-0.001943*t) + 0.0005 $= Q_3$



Part e:

Steady state values of heights:

```
xss = -A^(-1) * B * 0.0005
```

```
xss =

    1.5291
    1.0194
    0.5097
```

3) The system in problem 1 of Homework 2: $\frac{d^3y}{dt^3} + 5\frac{d^2y}{dt^2} + 10\frac{dy}{dt} + 6y = 2\frac{du}{dt} + u$ has the following transfer function:

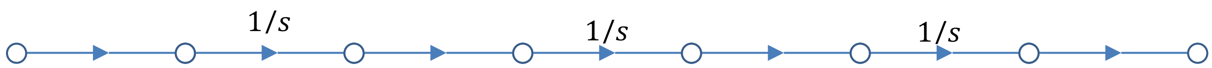
$$\frac{Y(s)}{U(s)} = \frac{2s+1}{s^3+5s^2+10s+6}$$

a. Use the method in Handout 4 to draw a signal flow graph for above transfer function using integrators ($1/s$), and gains (recall all nodes are summing junctions in signal flow graphs). (You may first draw the block diagram and then convert that to a signal flow graph, if not comfortable with signal flow graphs.)

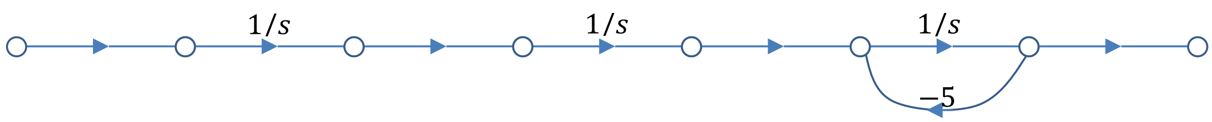
Answer: (Note: Answer not unique, others exist.) Method same as that done for block diagrams: Divide numerator and denominator by the highest power of s seen in denominator, here s^3:

$$\frac{Y(s)}{U(s)} = \frac{2s/s^3 + 1/s^3}{s^3/s^3 + 5s^2/s^3 + 10s/s^3 + 6/s^3} = \frac{2/s^2 + 1/s^3}{1 + 5/s + 10/s^2 + 6/s^3}$$

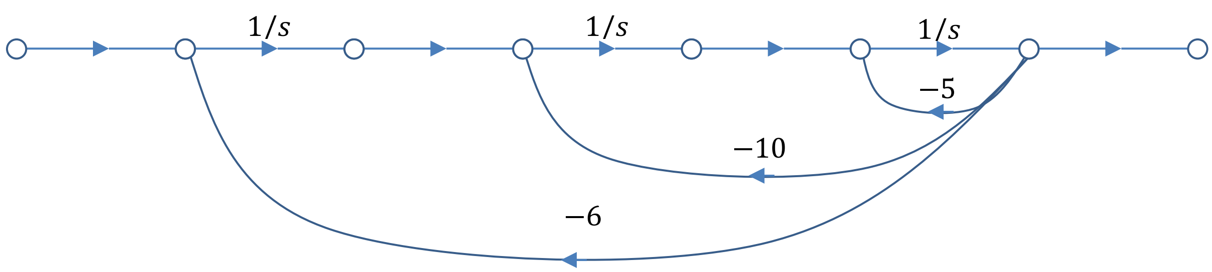
Next, draw three alternate 1's and $1/s$ arrows:



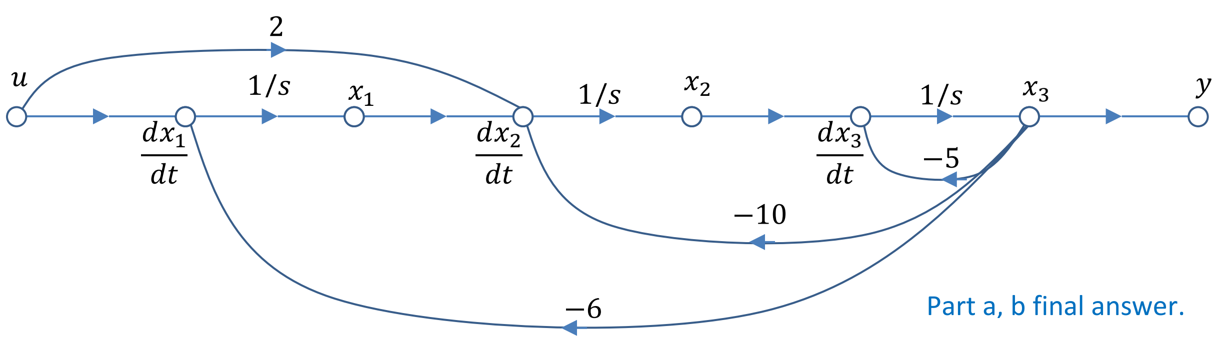
The objective is to make the terms in denominator sum of touching loop gains and the terms in numerator sum of touching forward gains. Begin with first term in denominator after the "1" and create a feedback path at the last $1/s$; this $1/s$ would be shared by all loops and forward paths:



Repeat for the next two terms in denominator:



The 2nd forward path is $1/s^3$ which corresponds to the existing forward path in above diagram, the 1st forward path, $2/s^2$ is created by bypassing a $1/s$ block (except the last one) by a gain of 2:



Part a, b final answer.

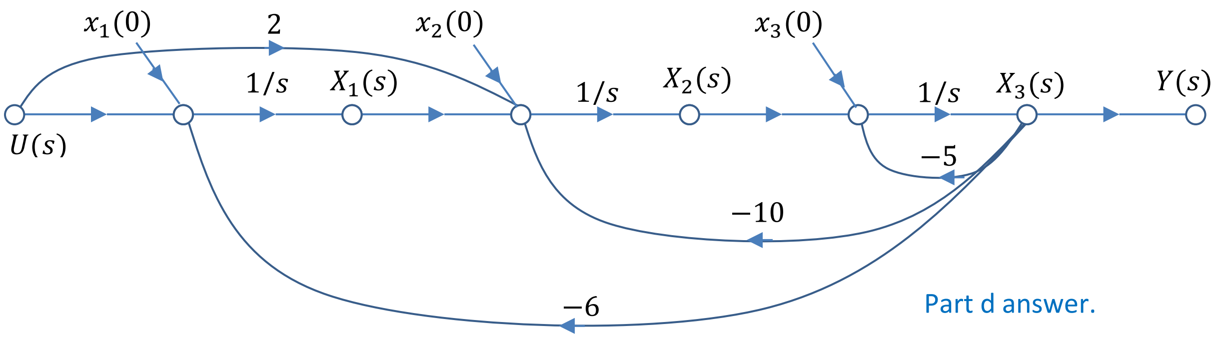
b. Label the nodes where the state variables and their derivatives are.
Answer shown in above figures, state variables are outputs of the integrators and their derivatives are the inputs of the integrators.

c. Find the state space equations from the signal flow graph in the form: $d\mathbf{x}/dt = \mathbf{A}\mathbf{x} + \mathbf{B}u$; $y = \mathbf{C}\mathbf{x} + \mathbf{D}u$; where u and y, input and output, are the same as those in the system differential equation.
Answer: Looking at above figure: $\frac{dx_1}{dt} = -6x_3 + u; \frac{dx_2}{dt} = x_1 - 10x_3 + 2u; \frac{dx_3}{dt} = x_2 - 5x_3 \Longrightarrow$

$$\begin{cases} dx_1/dt = -6x_3 + u \\ dx_2/dt = x_1 - 10x_3 + 2u \\ dx_3/dt = x_2 - 5x_3 \end{cases} \Longrightarrow \frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \overbrace{\begin{bmatrix} 0 & 0 & -6 \\ 1 & 0 & -10 \\ 0 & 1 & -5 \end{bmatrix}}^{\mathbf{A}} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \overbrace{\begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix}}^{\mathbf{B}} u$$

Also, from above figure output: $y = x_1 + 2x_2 \Longrightarrow$ $y = \overbrace{[0 \quad 0 \quad 1]}^{\mathbf{C}} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \overbrace{0}^{\mathbf{D}} u$

d. On the signal flow graph, show where the initial conditions enter. Answer: The initial condition for each state variable enters at the input of the corresponding integrator as shown below:



Part d answer.

e. Now simplify the signal flow graph so that an expression for output $Y(s)$ is found in terms of input $U(s)$ and the initial conditions $x_1(0) \equiv \left.\frac{dy^2}{dt^2}\right|_{t=0}, x_2(0) \equiv \left.\frac{dy}{dt}\right|_{t=0}, x_3(0) \equiv y(0)$ in the following form (all scalars):

$$Y(s) = T(s)U(s) + I_1(s)x_1(0) + I_2(s)x_2(0) + I_3(s)x_3(0)$$

NOTE: There is a mistake in part e please correct: there may not be one to one correspondences between state variables and output derivatives. Part e. should read:

Now simplify the signal flow graph so that an expression for output $Y(s)$ is found in terms of input $U(s)$ and the initial conditions $x_1(0), x_2(0), x_3(0)$ in the following form (all scalars):

$$Y(s) = T(s)U(s) + I_1(s)x_1(0) + I_2(s)x_2(0) + I_3(s)x_3(0)$$

Answer: Note that all loops and all forward paths, including those involving initial conditions, share the last integrators and hence are touching, as a result the formula below may be used to find the answer:

$$Y(s) = \frac{\text{Sum of touching Forward paths}}{1 - \text{sum of touching loop gains}}$$

$$Y(s)=\frac{U(s)\frac{1}{s^3}+U(s)\frac{2}{s^2}+x_1(0)\frac{1}{s^3}+x_2(0)\frac{1}{s^2}+x_3(0)\frac{1}{s}}{1-\left(-\frac{6}{s^3}-\frac{10}{s^2}-\frac{5}{s}\right)}=\frac{(s+2)U(s)+x_1(0)+s\,x_2(0)+s^2x_3(0)}{s^3+5s^2+10s+6}$$

f. Use symbolic MATLAB to find $Y(s)$ by equation (5 – 6) below to check your answer in part e.

$$\mathbf{Y}(s)=[\mathbf{C}(s\mathbf{I}-\mathbf{A})^{-1}\mathbf{B}+\mathbf{D}]\,\mathbf{U}(s)+\mathbf{C}(s\mathbf{I}-\mathbf{A})^{-1}\mathbf{x}_0;\ \text{Initial Conditions Vector } \mathbf{x}_0=\begin{bmatrix}x_1(0)\\x_2(0)\\x_3(0)\end{bmatrix}$$

MATLAB:

```
>> syms s x10 x20 x30 Us;
>> A = [0 0 -6; 1 0 -10; 0 1 -5];
>> B = [1;2;0];
>> C = [0 0 1];
>> D=0;
>> x0 = [x10; x20; x30];            % Define initial conditions vector.
>> Ys = (C * inv(s*eye(3) - A) * B + D) * Us + C * inv(s*eye(3) - A) * x0;
>> Ys = simplify(Ys)
Ys =
   (Us + x10 + 2*Us*s + s*x20 + s^2*x30)/(s^3 + 5*s^2 + 10*s + 6)     % Same as above.
```