IMACS User Guide for Software-in-the-Loop (SiL) simulation

How to run the SiL?

1. complete IMACS installation as per the IMACS installation guide document.
2. open a **CoppeliaSim** application, you can find the details to open the CoppeliaSim in IMACS installation guide document. The document will be shared with the students.
3. open a v-rep scene in CoppeliaSim. V-rep scene is in the folder '**/home/imacs/Desktop/imacs/vrep_scenes**'. Please choose one of the following scenes :
    - StraightRoad.ttt
    - CurveRoad.ttt
4. start MATLAB. Please refer to IMACS installation guide document.
5. open a framework MATLAB code (**'initSimulink.m'** and **'Vrep_Simulation.slx'**) in the folder **'/home/imacs/Documents/Assg2/DASA-master/DASA_Framework/mexopencv-master'**.
6. configure SiL as per the sequential, parallel, and pipelined configurations following instructions provided in Section I to Section IV.
7. design and update the controller Simulink block as per the instructions provided in Section V.
8. run the **initSimulink.m** MATLAB script.
9. run the **Vrep_Simulation.slx** Simulink model, you can create copies of this model for different cases in the assignment.

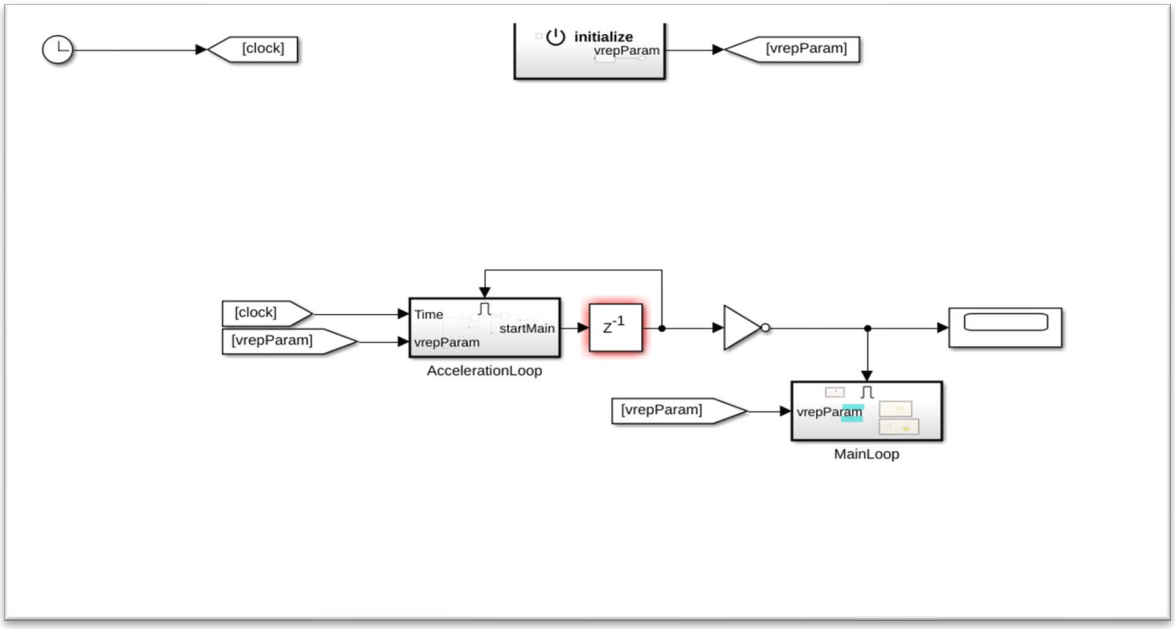
Section I: SiL configuration with sequential controller

In **initSimulink.m** file, change the initial timing parameter settings, as shown in the following screenshot. Next provide the following parameters computed as per the case in hand:

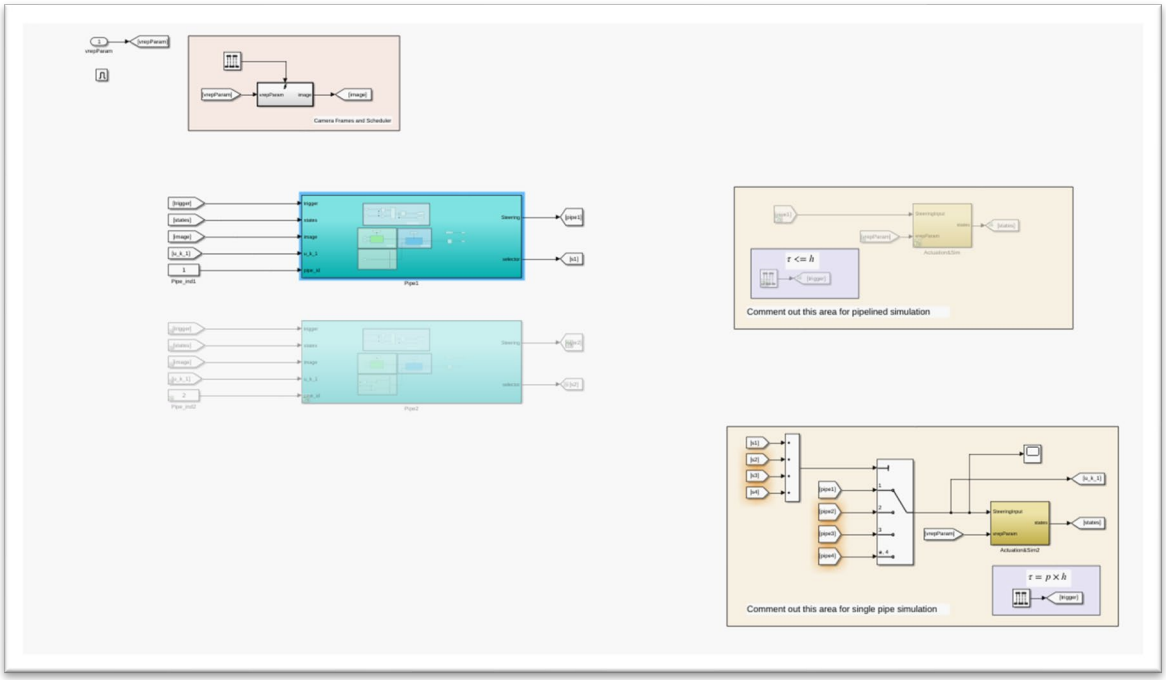
- **period_ms** : effective sampling period
- **num_parallel :** number of cores

```
%%
clear all
clc

%% initial timing parameter setting

period_ms = 0.1;  % Set to you effective sampling period
num_parallel = 1; % input the number of cores
num_pipes = 1;
```

- open **Vrep_Simulation.slx** Simulink model, you will see model as shown in the following screenshot.

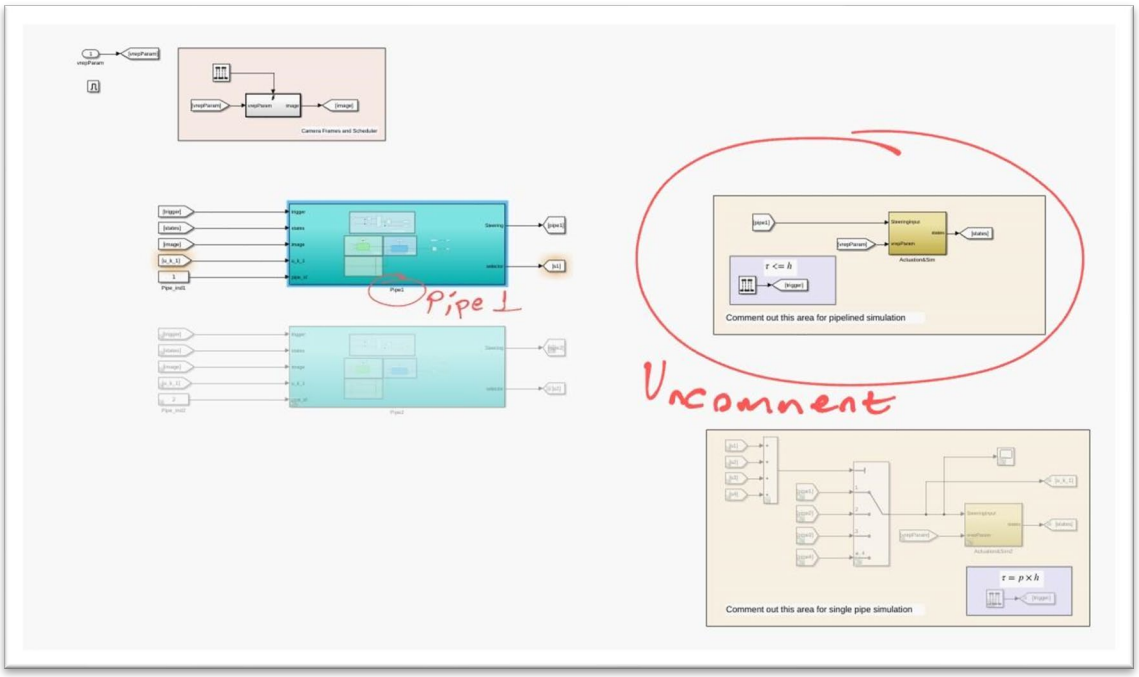


- open **MainLoop** Simulink block, which has following structure.



- **For tau <= h case:**
  - Step 1: Enter **period_ms** = effective sampling period and **num_parallel =1** in file **initSimulink.m**
  - Step 2: Inside **MainLoop** Simulink block, do the following :

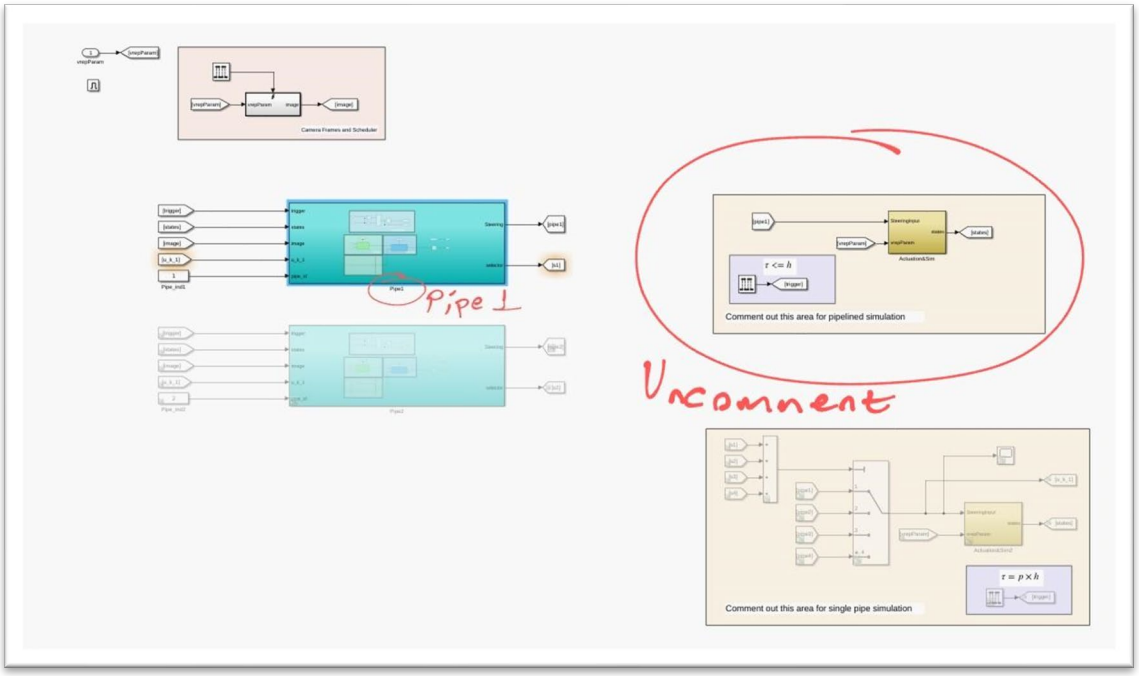
    Keep only one pipe indicated by '**pipe1'** annotation, make sure that the block with '**(tau<=h)**' trigger annotation is uncommented and the block with trigger annotation as '**(tau = p x h)'** is commented out. Refer to the following snapshot.



Section II: SiL configuration for controller with parallel implementation

In case of parallel implementation i.e., Case II in assignment, following steps needs to be done by the student

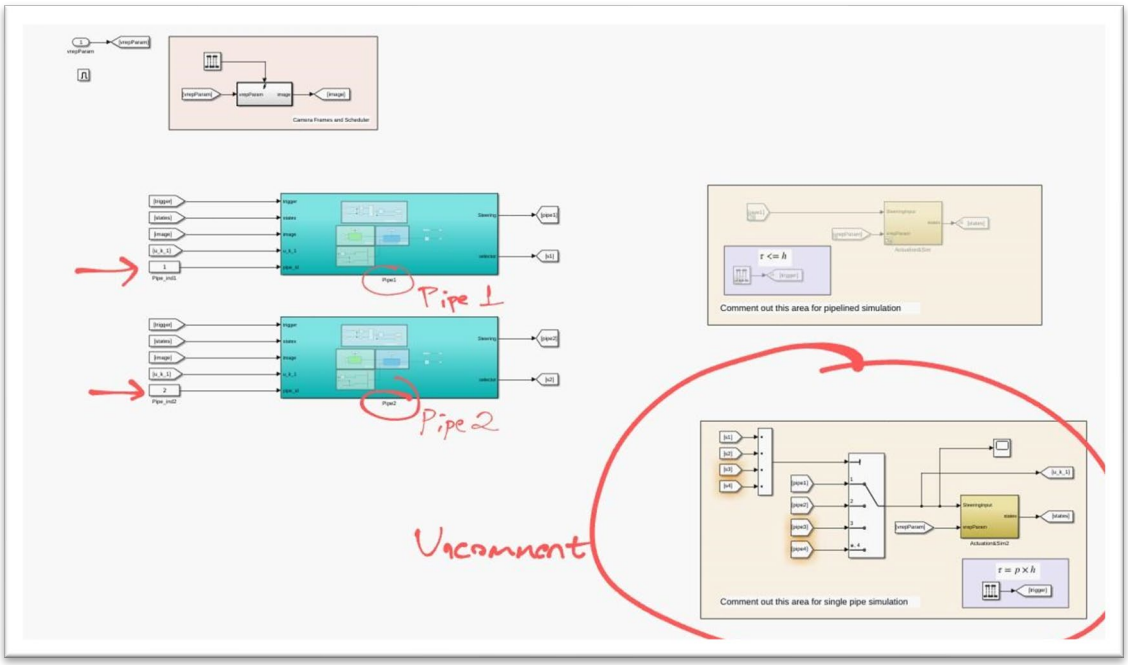
- **For tau <= h case:**
  - Step 1 : Enter **period_ms** = effective sampling period and **num_parallel =2 (Students will decide the number of cores)** in file **initSimulink.m**
  - Step 2: Keep only one pipe indicated by '**pipe1'** annotation, make sure that the block with '**(tau<=h)**' trigger annotation is uncommented and the block with trigger annotation as '**(tau = p x h)' is** commented out. Refer to the following snapshot.



Section III: SiL configuration for controller with pipelined sensing

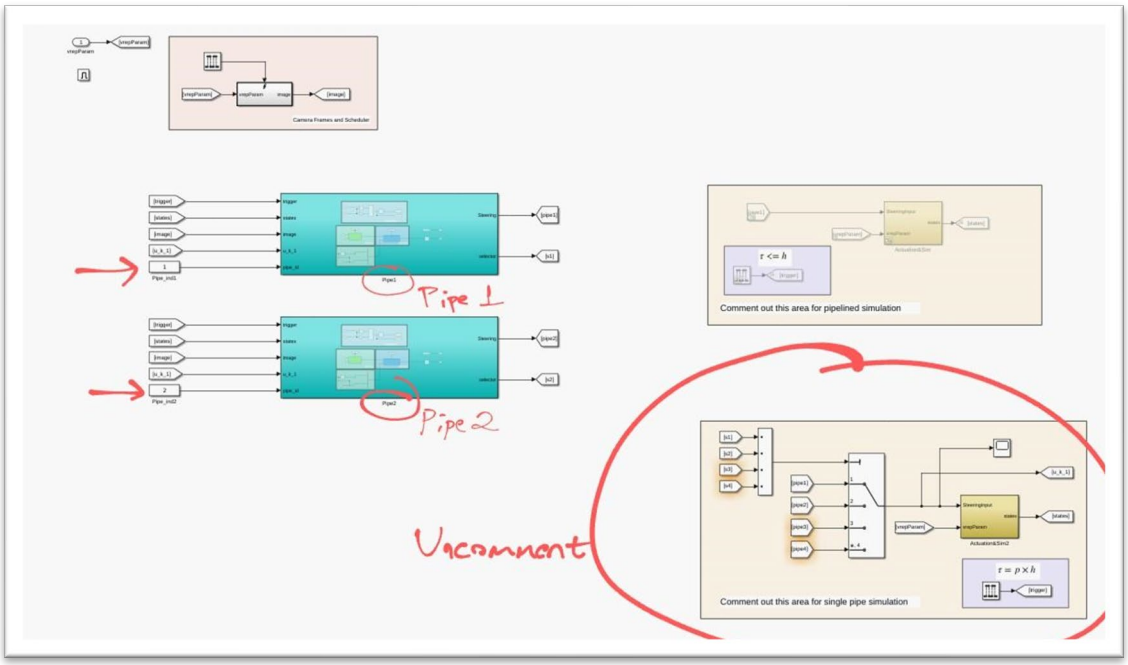
In case of pipelined implementation i.e., Case III and IV, following steps needs to be done by the student

- Step 1 : enter **period_ms** = effective sampling period, **num_parallel =1, and num_pipes = 2 (Students will decide the number of pipes)** in file **initSimulink.m**
- Step 2 **:** comment out the block with trigger annotation **'(tau<=h)'** and uncomment the block with trigger annotation **'(tau = p x h)**'. Check the following screenshot for the reference.
- Step 3 **:** copy & paste the **pipe1** annotation block and it will lead to two pipes i.e., **pipe1** and **pipe2.** Once you get additional pipe i.e., pipe2, please change the pipe index to 2 (pointed by the red arrows in the following screenshot) as pipe indexing starts from 1 for pipe1 then 2 for pipe2, so on. For a higher no of pipes, follow the same procedure to add additional pipes.
- Step 4 **:** for the new pipe (i.e., pipe2), make sure that output tags (selector tag and steering angle tag) are connected to respective locations in the yellow area (circled in the following screenshot).
- Step 5 **:** the provided model supports up to 4 pipes. For less than or equal number of pipes, leave the yellow block as is. If you want more than 4 pipes, increase the number of input ports of both the switch and summation in the yellow area, and do the necessary connections from pipes to them.



Section IV: SiL configuration for controller with parallel and pipelined implementation

- Step 1 : Enter **period_ms** = effective sampling period , **num_parallel =2, and num_pipes = 2 (Students will decide the number of pipes)** in file **initSimulink.m**
- Step 2 **:** comment out the block with trigger annotation **'(tau<=h)'** and uncomment the block with trigger annotation **'(tau = p x h)**'. Check the following screenshot for the reference.
- Step 3 **:** you need to copy & paste the **pipe1** annotation block and it will lead to two pipes i.e., **pipe1** and **pipe2.** Once you get additional pipe i.e., pipe2, please change the pipe index to 2 (pointed by the red arrows in the following screenshot) as pipe indexing starts from 1 for pipe1 then 2 for pipe2, so on. For a higher no of pipes, follow the same procedure to add additional pipes.
- Step 4 **:** for the new pipe (i.e., pipe2), make sure that output tags (selector tag and steering angle tag) are connected to respective locations in the yellow area (circled in the following screenshot).
- Step 5 **:** the provided model supports up to 4 pipes, if you have less than or equal number of pipes, you can leave the yellow block as is. If you want more than 4 pipes, increase the number of input ports of both the switch and summation in the yellow area, and do the necessary connections from pipes to them.



Section V: Controller implementation in SiL for all the cases

Open the Simulink model as **vrepSimParPipe→MainLoop→Pipe0→Controller**. For implementing a new controller, e.g. pipelined sensing with LQR, you need to modify the MATLAB Simulink model. The controller should be designed separately in MATLAB script and can be used in the Simulink in the existing **controller block highlighted by blue square** in following screenshot with your own controller.

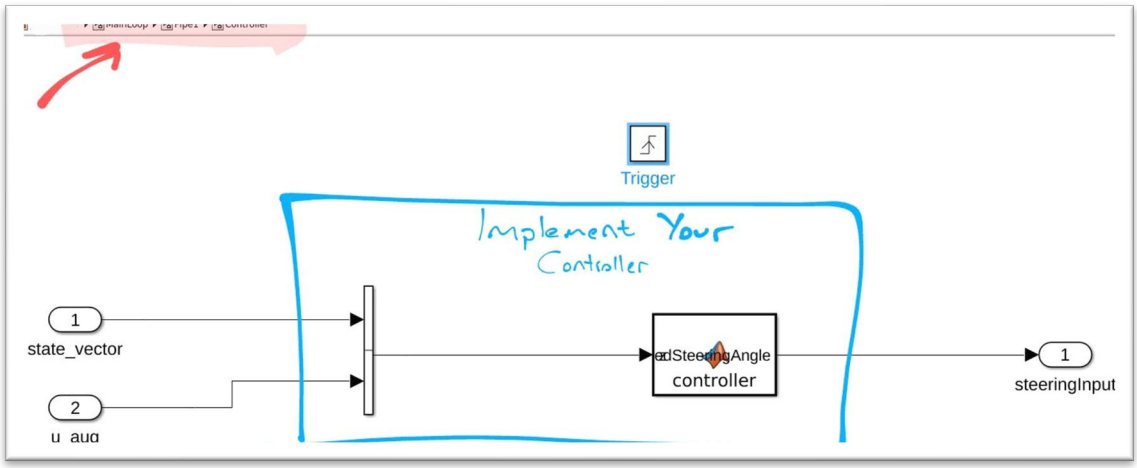
To use the LQR controller implementation in SiL for all the above cases, the following steps need to be followed:

First, test **Model-in-Loop (MiL)** (for the LKAS system) by designing LQR controller in MATLAB; As explained in the following steps

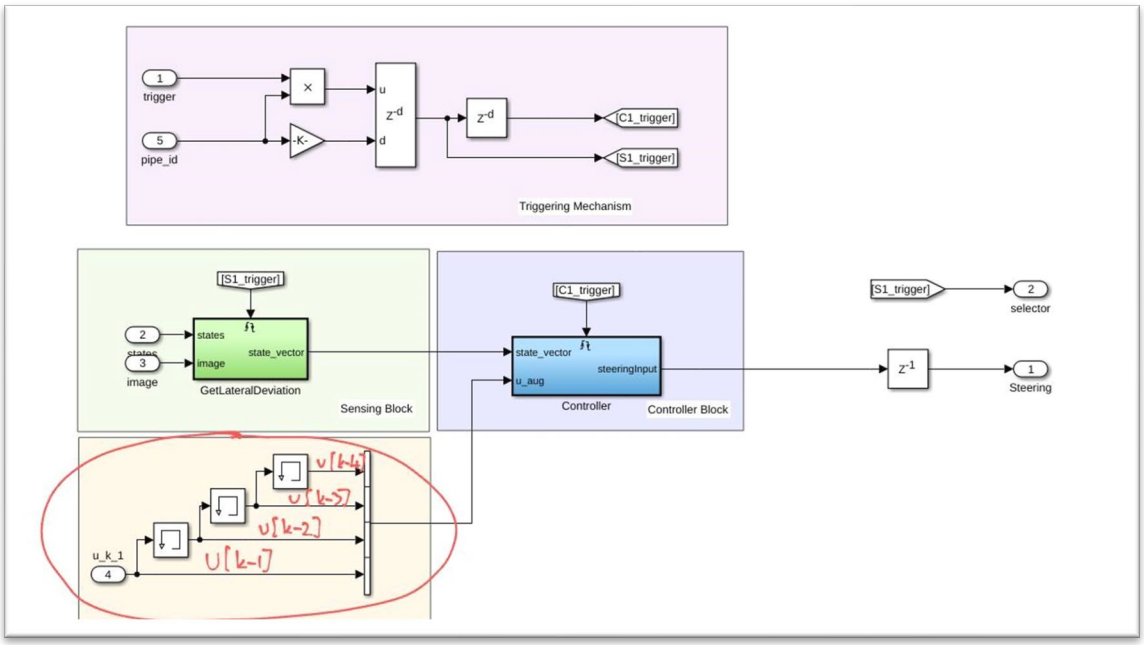
- Step 1: calculate sensor-to-actuator delay and sampling period considering the application task graph (or SDF graph) and the mapping (e.g., sequential, parallel, pipelined).
- Step 2 : use the discreteTimeController.m function, which needs delay and sampling period as the ones calculated above.
- Step 3 : designControlGainsLQR.m may be used for designing LQR controller. The dimension and weights of the Q matrix in LQR optimization should be adjusted accordingly to the specific case at hand.
- Step 4 : discreteTimeController.m returns the following variables.
    - phi_aug, Gamma_aug, C_aug, K, F, K_T, T .
- Step 5 : for simulating the MIL as closed-loop with LQR controller, use the phi_aug, Gamma_aug, C_aug matrices and K (LQR gain), F(feed-forward) for designing the control law.
- Step 6 : Plot the final output and note the performance (e.g., settling time) of the system.

Next, move to SiL with the following steps.

- Step 7 : K_T vector and T matrix are obtained from Step 4. Use both variables in the controller block, as shown in the following screenshot. **Note:** you have to omit the first element of the K_T vector. Since we have to deal with only controllable states, the first element of the K_T vector corresponds to the uncontrollable state.



In case of augmented system state or old control input, you need to add extra memory block highlighted by red circle. In order to use augmented state definitions in your controller, construct a vector of previous inputs as shown in the following figure. The following example shows a vector of 4 past inputs as an example.



As another example, if you need two old inputs for your controller, the block would look like the following image.