Name: B.Anshuman

Roll No: 200259 (e.g. 170001) **Dept.**: EE (e.g. CSE)

Indian Institute of Technology Kanpur
CS637 Embedded and Cyber-Physical Systems
Homework Assignment 3
Deadline: September 16, 2022

Total: **40 marks**

1. Write the answers **neatly** in the given boxes.
2. You may discuss the solutions with the other students, but you have to write them in your own words.

Problem 1. (10 points) Provide the state-space representation of the dynamics of a DC Motor. Assume that there is no additional load on the motor. Next, Design a Simulink model to capture the dynamics and simulate the model for an input PWM voltage signal with magnitude 1V, frequency 1 kHz and duty cycle 0.1. Assume that the kinetic friction of the motor is negligible. Take the values of the other parameters as below:

$$I = 3.88 \times 10^{-7}\, kg \cdot m^2$$
$$k_b = 2.75 \times 10^{-4}\, V/RPM$$
$$k_T = 5.9 \times 10^{-3}\, N \cdot mA^{-1}$$
$$R = 1.71\, \Omega$$
$$L = 1.1 \times 10^{-4}\, H$$

[LS15] Edward A. Lee and Sanjit A. Seshia, Introduction to Embedded Systems, A Cyber-Physical Systems Approach, Second Edition, http://LeeSeshia.org, ISBN 978-1-312-42740-2, 2015. References:

1. Page 205 - LeeSeshia Introduction to Embedded Systems.

Mathematical modelling:

Initially when the motor hasn't started rotating yet, at $t = 0^+$,

$$V(t) = Ri(t) + L\frac{di(t)}{dt} \iff V(s) = (R + sL)i(s)$$

Due to this current i, torque is generated in the motor shaft,

$$\tau(t) = k_T i(t) \iff \tau(s) = k_T i(s)$$

Which led to torque equation (with constant load torque as τ_L, which equals 0 in this question) as

$$I\frac{d\omega(t)}{dt} = \tau(t) - \tau_L \iff sI\omega(s) = \tau(s) - \tau_L \tag{1}$$

Now due to rotation of motor, there is an armature reaction leading to back e.m.f generated,

$$E_b(t) = k_b\omega(t) \iff E_b(s) = k_b\omega(s)$$

This leads to change in original equation

$$V(t) - E_b(t) = Ri(t) + L\frac{di(t)}{dt} \iff V(s) - k_b\omega(s) = Ri(s) + sLi(s) \tag{2}$$

Name: B.Anshuman

Roll No: 200259 (e.g. 170001) **Dept.**: EE (e.g. CSE)

Indian Institute of Technology Kanpur
CS637 Embedded and Cyber-Physical Systems
Homework Assignment 3
Deadline: September 16, 2022

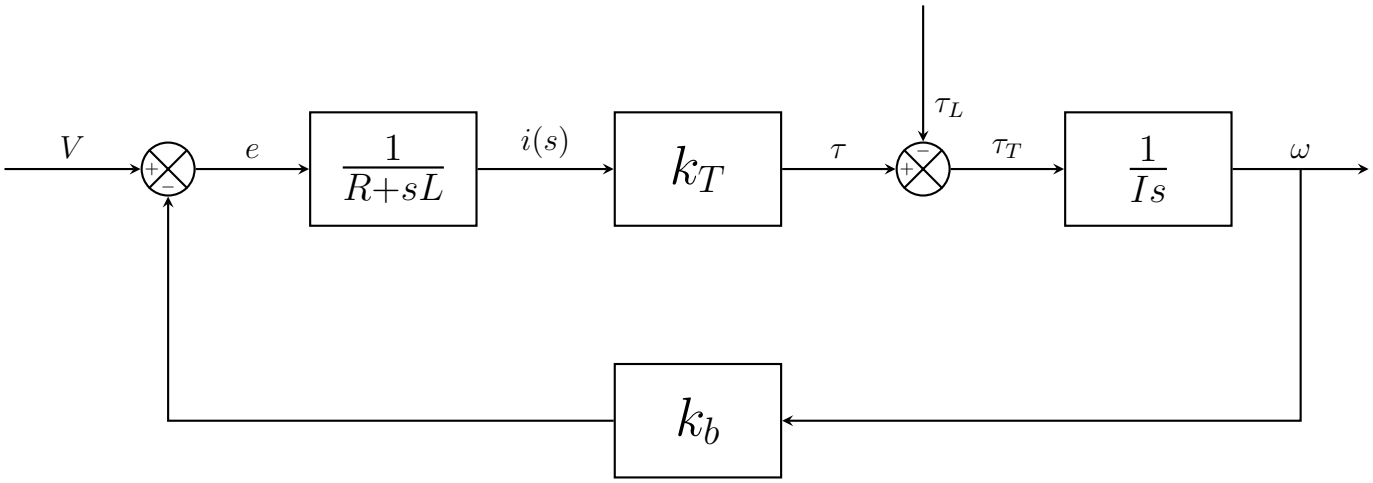


Figure 1: Block Diagram for voltage controlled DC Motor

Implementing the above in Simulink:

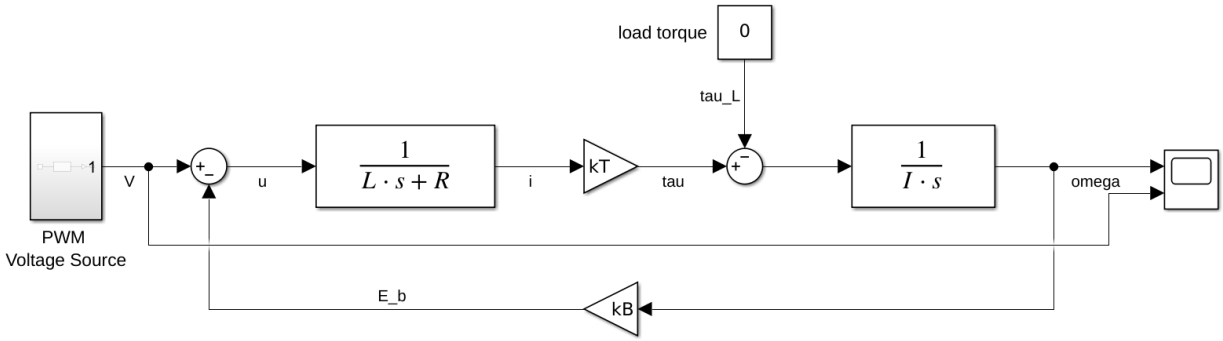


Figure 2: Transfer function implementation

For obtaining the state space equations, Consider equations (1) and (2) we get:

$$\begin{bmatrix} \dot{\omega} \\ \dot{i} \end{bmatrix} = \begin{bmatrix} 0 & \frac{k_T}{I} \\ \frac{-k_B}{L} & \frac{-R}{L} \end{bmatrix} \begin{bmatrix} \omega \\ i \end{bmatrix} + \begin{bmatrix} 0 & \frac{-1}{I} \\ \frac{1}{L} & 0 \end{bmatrix} \begin{bmatrix} V \\ \tau_L \end{bmatrix} \tag{3}$$

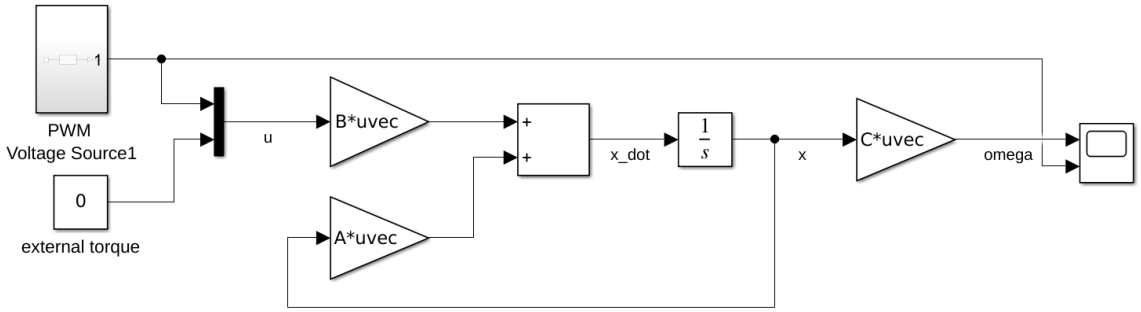


Figure 3: State space implementation

Name: B.Anshuman

Roll No: 200259 e.g. 170001 **Dept.**: EE e.g. CSE

Indian Institute of Technology Kanpur
CS637 Embedded and Cyber-Physical Systems
Homework Assignment 3
Deadline: September 16, 2022

Resultant response of both representation for $1V$ DC Input with 0.1 duty cycle at $1kHz$:

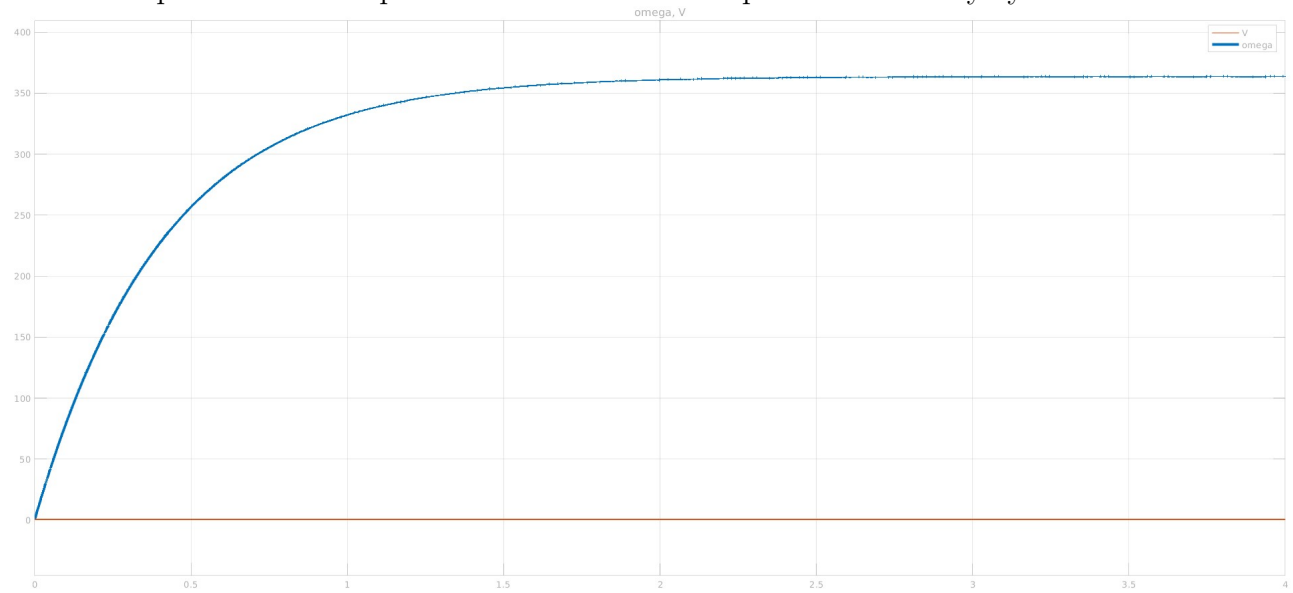


Figure 4: Response to 0.1 V (equivalent)

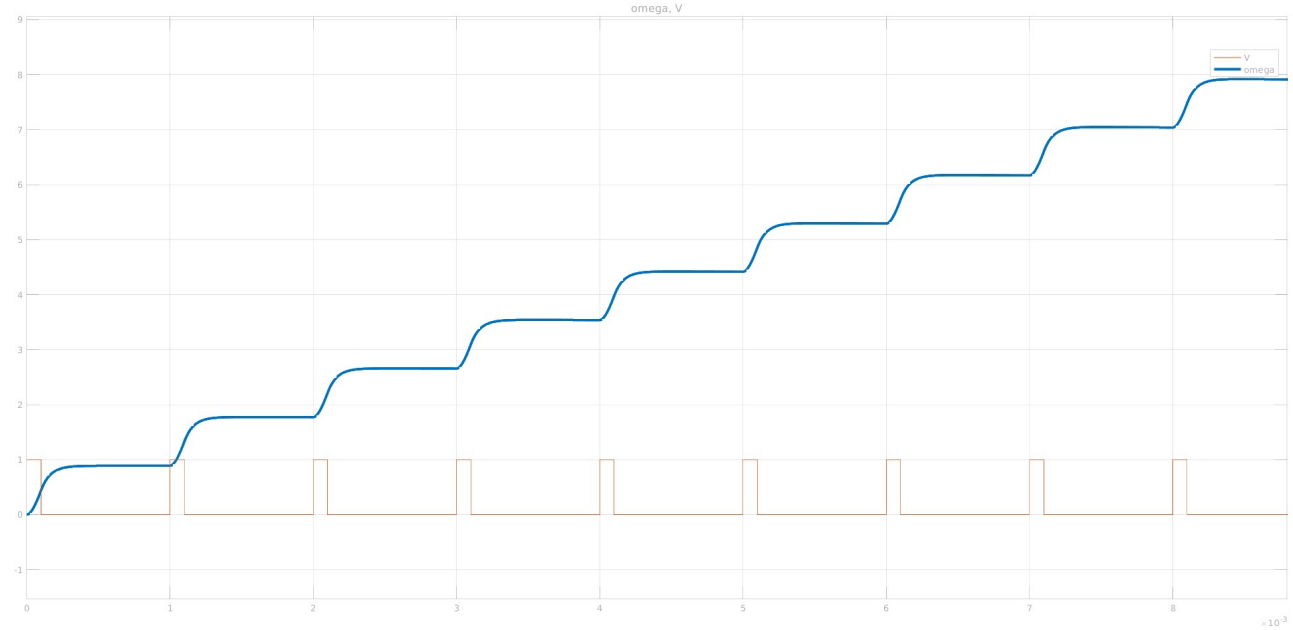


Figure 5: Initial response curve (blue), input (red)

Name: B.Anshuman

Roll No: 200259 (e.g. 170001)

Dept.: EE (e.g. CSE)

Indian Institute of Technology Kanpur
CS637 Embedded and Cyber-Physical Systems
Homework Assignment 3
Deadline: September 16, 2022

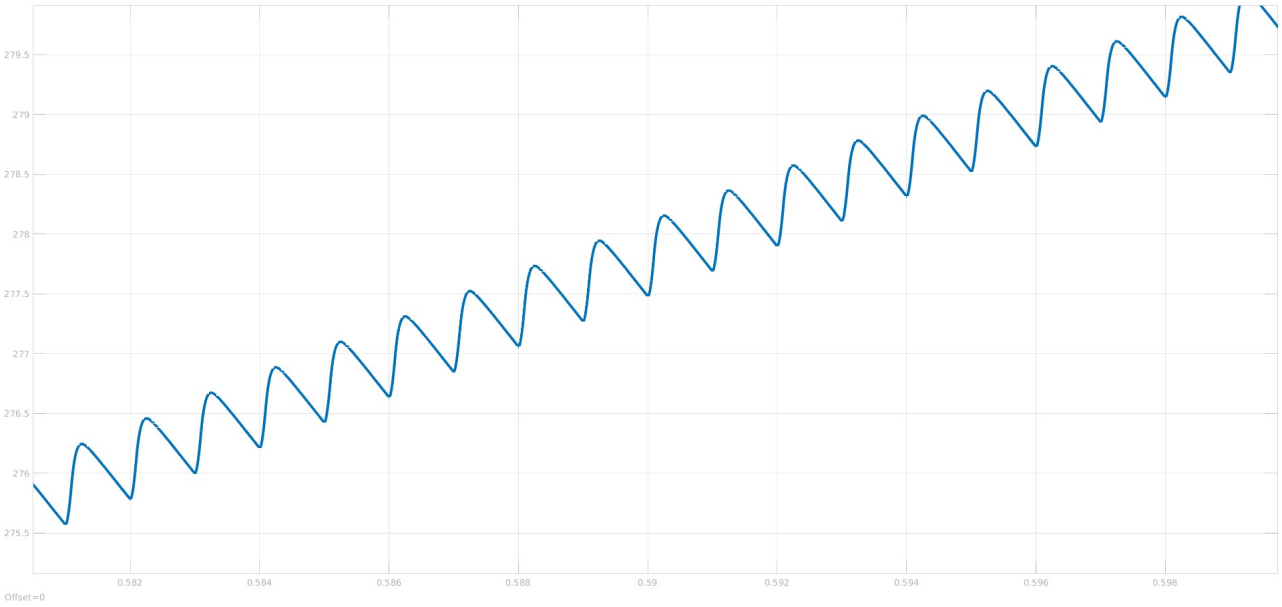


Figure 6: PWM rise and fall visible in response curve

Name: B.Anshuman

Roll No: 200259 (e.g. 170001) **Dept.**: EE (e.g. CSE)

Indian Institute of Technology Kanpur
CS637 Embedded and Cyber-Physical Systems
Homework Assignment 3
Deadline: September 16, 2022

Problem 2. (20 points) Consider the vehicle steering control problem in Example 6.4 in [AM09]. Assume that $k_1 = 1$, $k_2 = 1.6$, and $k_r = 1$. Model the control system in Simulink using double precision floating point arithmetic. Now replace the model of the controller with the ones that use 16 bit and 8-bit fixed-point arithmetic. In each case, determine the fixed-point data types precisely. Plot the difference between the first state for the floating-point controller and that for the fixed-point controllers. Generate code for both the floating point controller and the fixed-point controllers using different optimization options. Describe your experience with code generation.

[AM09] K. J. Astrom and R. M. Murray. Feedback Systems: An Introduction for Scientists and Engineers. Princeton University Press, 2009.
http://www.cds.caltech.edu/∼murray/books/AM05/pdf/am08-complete_22Feb09.pdf.

We know that plant modelling of vehicle steering problem in which we want our vehicle to move parallel to X-axis, can be done as:

$$X = \begin{bmatrix} \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v_0 sin(tan^{-1}(\frac{a \cdot tan(u)}{b}) + \theta) \\ v_0 \frac{tan(u)}{b} \end{bmatrix} \tag{4}$$

Through linearisation about X = (0, 0) and u = 0 and after normalisation with new time scale as $\tau = v_0 \frac{t}{b}$ and length scale b, state space equations come out as

$$\dot{X} = \begin{bmatrix} \dot{x_1} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} \gamma \\ 1 \end{bmatrix} u \tag{5}$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} X \tag{6}$$

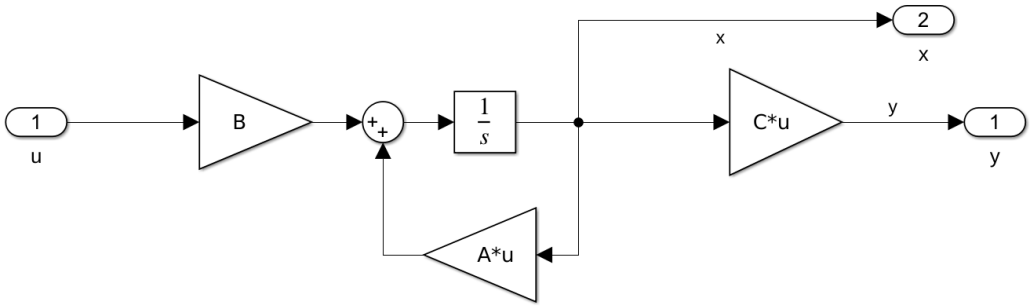


Figure 7: State Space Equations implementation

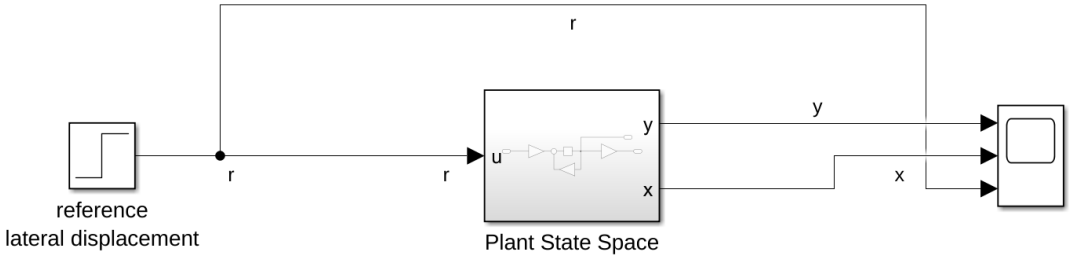


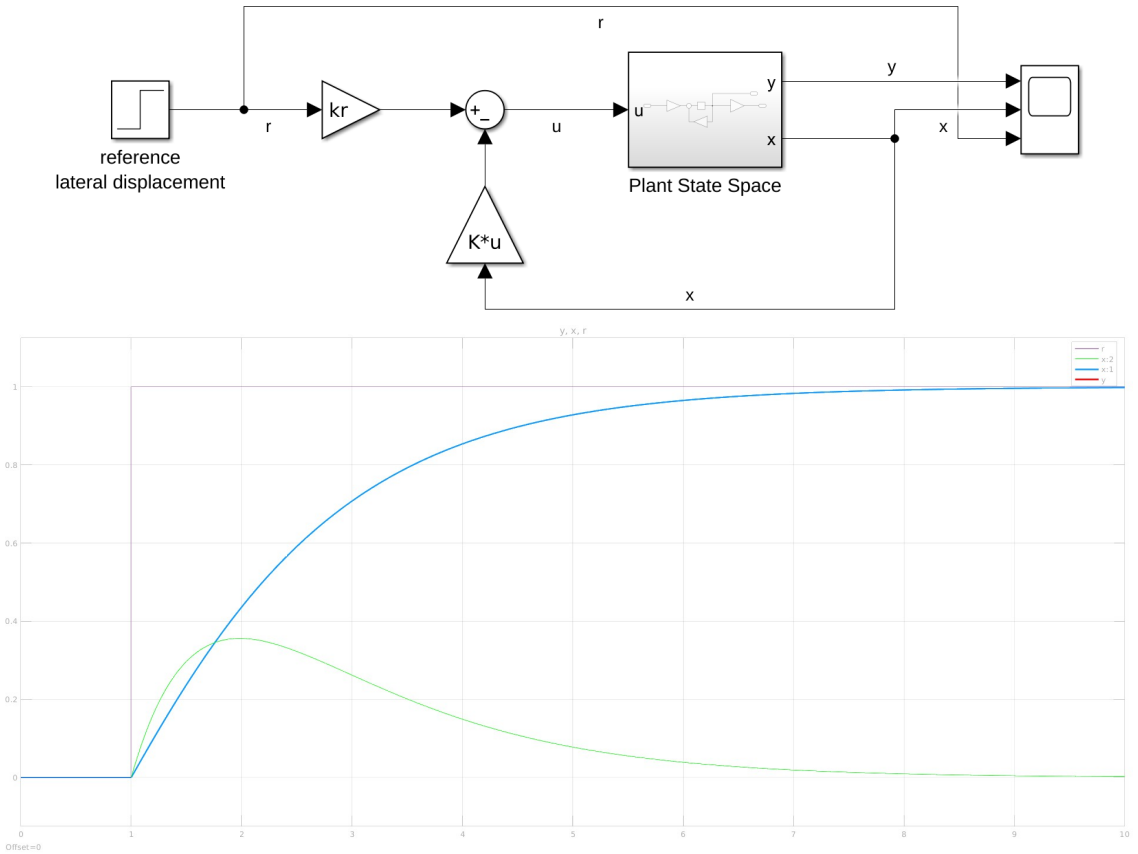
Figure 8: Open Loop System

Name: B.Anshuman

Roll No: 200259 (e.g. 170001) **Dept.**: EE (e.g. CSE)

Indian Institute of Technology Kanpur
CS637 Embedded and Cyber-Physical Systems
Homework Assignment 3
Deadline: September 16, 2022

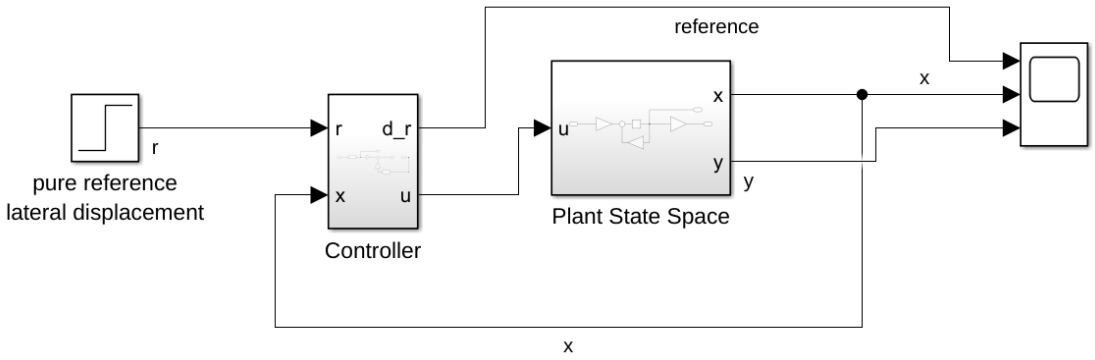
By default, Simulink uses double precision floating point numbers. Therefore using the below system:



Lateral deviation (blue), theta (green)

Figure 9: Double precision model and response

Now to decide what decimal location we want to set fixed point data types, I first chose min-max values as [-5, 5] that may be encountered by that channel, and continued with best precision, which gave 12 bits for decimal points in case of 16 bit fixed point data type, and 4 bits for decimal points in case of 8-bit fixed point data type.



Name: B.Anshuman

Roll No: 200259 e.g. 170001 **Dept.**: EE e.g. CSE

Indian Institute of Technology Kanpur
CS637 Embedded and Cyber-Physical Systems
Homework Assignment 3
Deadline: September 16, 2022

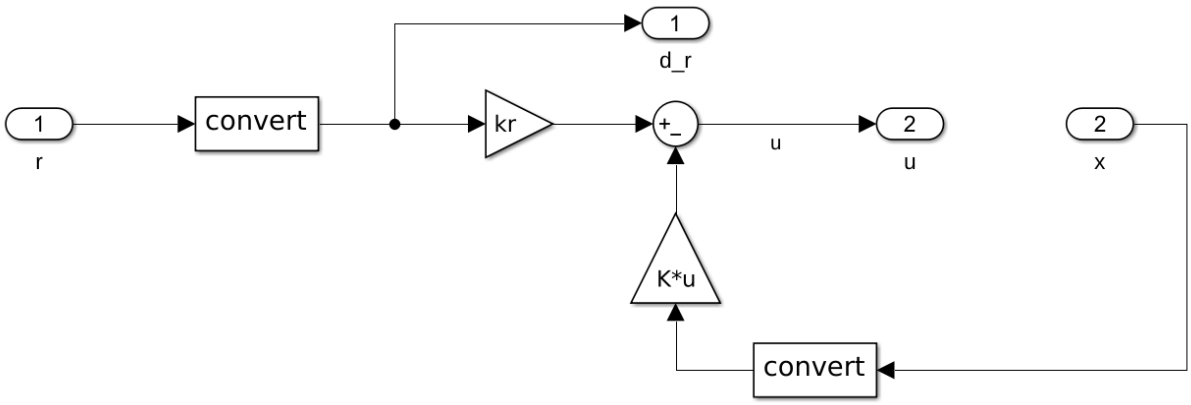
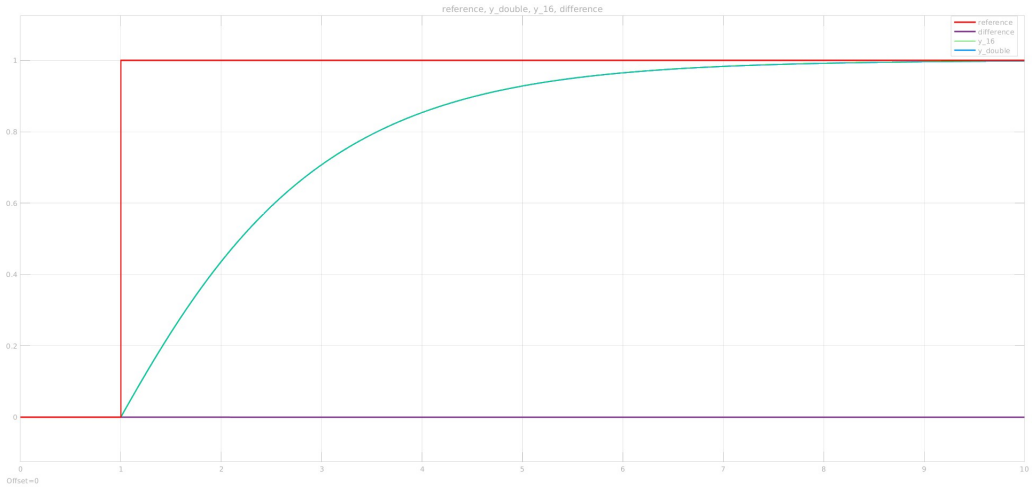


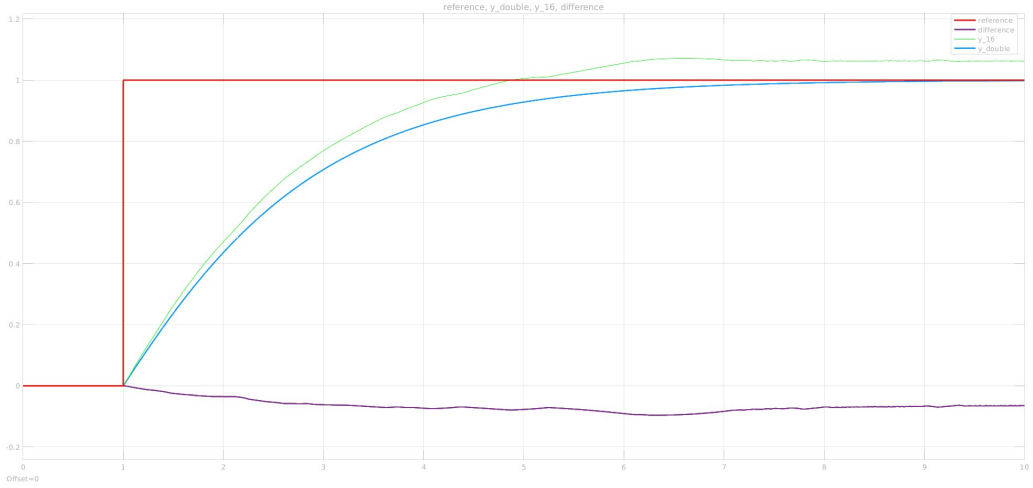
Figure 10: System with controller (above), controller handling fixed point data-type (below)

We modify the data-type of channels handled by the controller through **convert** blocks as above, as well as convert parameters to fixed-point form through sfi function in matlab.

1. First choosing 16-bit fixed point numbers, with best precision (i.e. with 12 bits for decimal as explained before)



2. Secondly, choosing 8-bit fixed point numbers with best precision (i.e. with 4 bits for decimal as explained before)



Name: B.Anshuman

Roll No: 200259 e.g. 170001 **Dept.**: EE e.g. CSE

Indian Institute of Technology Kanpur
CS637 Embedded and Cyber-Physical Systems
Homework Assignment 3
Deadline: September 16, 2022

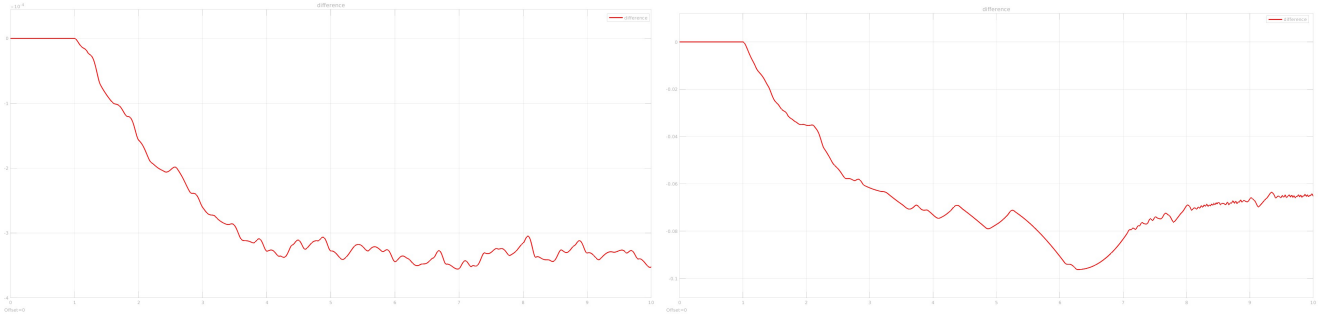


Figure 11: 16 bit fixed v/s double error (left) 8 bit fixed v/s double error (right)

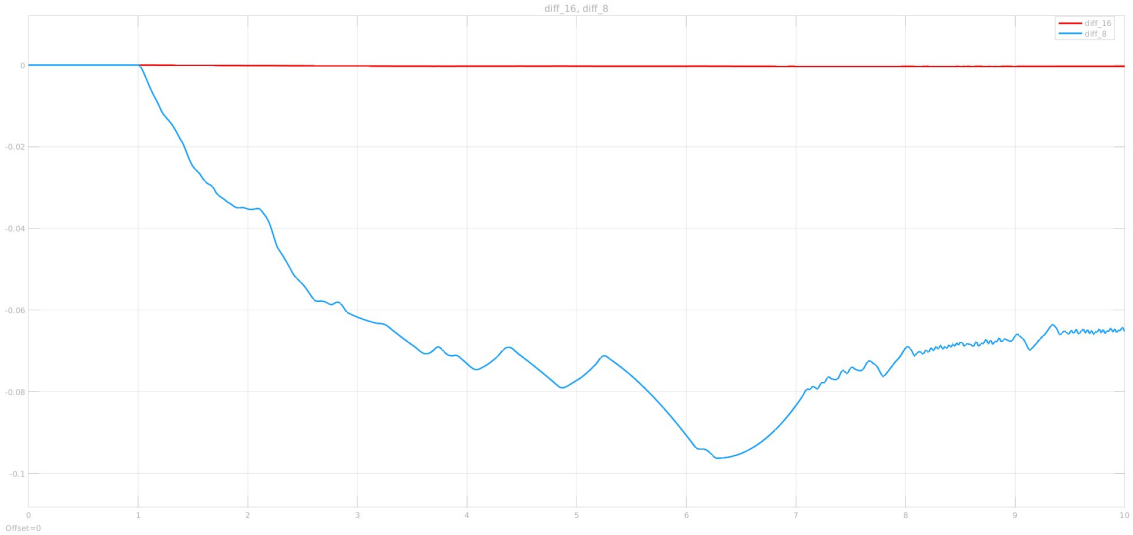


Figure 12: Comparison of above errors, 16-bit v/s double (red) 8-bit v/s double (blue)

I tried searching for code generation in SimuLink, giving results about MATLAB Coder, Simulink Coder for MPC Controllers and Embedded Coder. I wasn't able to figure out how to generate code from the control subsystem created above, all guides on the internet were outdated, and documentation didn't help much except I understood how to convert MATLAB code to ISO Cpp.

Name: B.Anshuman

Roll No: 200259 (e.g. 170001) **Dept.**: EE (e.g. CSE)

Indian Institute of Technology Kanpur
CS637 Embedded and Cyber-Physical Systems
Homework Assignment 3
Deadline: September 16, 2022

Problem 3. (10 points) Work out Problem 1 in the Exercises of Chapter 9 in [LS15].

Consider the function *compute_variance* listed below, which computes the variance of integer numbers stored in the array data.

```
1  int data[N];
2
3  int compute_variance() {
4      int sum1 = 0, sum2 = 0, result;
5      for (int i = 0; i < N; i++) {
6          sum1 += data[i];
7      }
8      sum1 /= N;
9
10     for (int i = 0; i < N; i++) {
11         sum2 += data[i] * data[i];
12     }
13     sum2 /= N;
14
15     return (sum2 - sum1*sum1);
16 }
```

Suppose this program is executing on a 32-bit processor with a direct-mapped cache with parameters (*m, S, E, B*) = (32, 8, 1, 8). We make the following additional assumptions:

- An int is 4 bytes wide.
- sum1, sum2, result, **N**, and i are all stored in registers.
- data is stored in memory starting at address 0x0.

Answer the following questions:

1. Consider the case where N is 16. How many cache misses will there be?
2. Now suppose that N is 32. Recompute the number of cache misses.
3. Now consider executing for N = 16 on a 2-way set-associative cache with parameters (*m, S, E, B*) = (32, 8, 2, 4). In other words, the block size is halved, while there are two cache lines per set. How many cache misses would the code suffer?

[LS15] Edward A. Lee and Sanjit A. Seshia, Introduction to Embedded Systems, A Cyber-Physical Systems Approach, Second Edition, http://LeeSeshia.org, ISBN 978-1-312-42740-2, 2015.

We assume a word is 4 bytes throughout our discussion below.

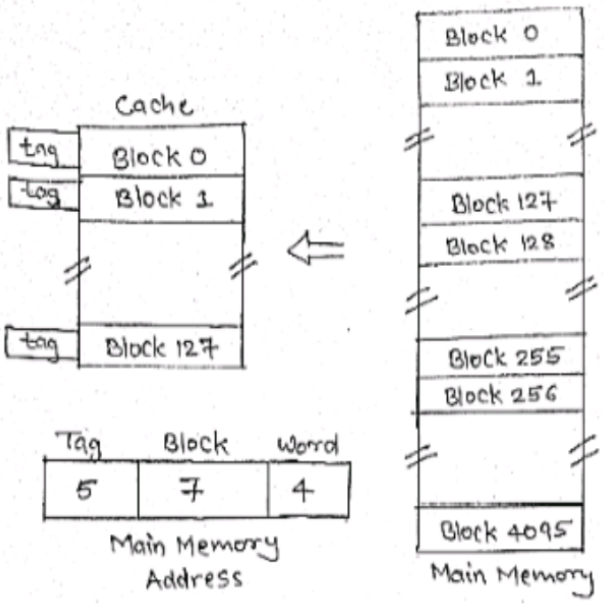
For explaining how memory blocks are mapped to sets in cache, let's go through the following scenario.
Suppose main memory is divided into blocks of 4 words each, there being 4094 blocks, $64kB$ (2^{14} words) in total. Also suppose cache is divided into 128 blocks, each having 4 words storage, giving $2kB$ (2^9 words) in total.

Name: B.Anshuman

Roll No: 200259 (e.g. 170001) **Dept.**: EE (e.g. CSE)

Indian Institute of Technology Kanpur
CS637 Embedded and Cyber-Physical Systems
Homework Assignment 3
Deadline: September 16, 2022

Mapping of cache sets to virtual memory happens in the way shown below: Here Block {i} in memory is mapped to Block $\{i \, mod(128)\}$ in cache. This comes from the fact that division of main memory address shown above comes as block then word, and the rest of bytes at start acts as tag for that block.



To explain this a bit more, suppose we have an array *data* of size 16 words starting from 0x0 in main memory. (The below is shown in little endian)

In the table below for an address, each letter represents 4 bits (0 to f), thus last letter in each address represents the word, and second last letter represents the cache block to which that memory is mapped to. Clearly data[0] to data[3] is stored in cache block 0 (0x00 to 0x0f), and the next contiguous set of data[4] to data[7] is stored in cache block 1 (0x10 to 0x1f), thus leading to contiguous memory blocks being stored in different sets instead of collision occurring.

The prefix of address '0b00000-0000000-0000' is decided by the memory block it is in, so non-contiguous data would usually have different tags.

data[0]	→	0x00	0x01	0x02	0x03	(block 0)
data[1]	→	0x04	0x05	0x06	0x07	(block 1)
data[2]	→	0x08	0x09	0x0a	0x0b	(block 2)
data[3]	→	0x0c	0x0d	0x0e	0x0f	(block 3)
data[4]	→	0x10	0x11	0x12	0x13	(block 4)
⋮	⋮	⋮	⋮	⋮	⋮	⋮
data[14]	→	0x38	0x39	0x3a	0x3b	(block 14)
data[15]	→	0x3c	0x3d	0x3e	0x3f	(block 15)

Name: B.Anshuman

Roll No: 200259 (e.g. 170001) **Dept.**: EE (e.g. CSE)

Indian Institute of Technology Kanpur
CS637 Embedded and Cyber-Physical Systems
Homework Assignment 3
Deadline: September 16, 2022

Call the first loop as Loop 1 and the second as Loop 2.

1. Given N = 16. Assuming all variables are stored in registers except *data* array given. Initially cache contains "garbage values", those not relevant to this function.
   In Loop 1, data[0] is called, leading to cache miss. Therefore memory management system would take 8 byte block from main memory at 0x00 to 0x1f. This would put data[0] and data[1] in cache set 0. The next call would find data[1] in cache, leading to cache hit in cache set 0.
   Now data[2] is called, cache miss, leading to data[2] and data[3] (0x20 to 0x3f) getting stored in set 1. The fourth call of data[3] would lead to cache hit.
   Similarly, at at 15th step, data[14] would lead to cache miss, leading to data[14] and data[15] getting stored in cache set 7, and 16th step would lead to cache hit.
   Ending of Loop 1 has led to data array getting stored completely in cache. Therefore in Loop 2, there would be no cache miss.
   There has been 8 cache misses throughout the 48 memory calls in this function.

2. Given N = 32. In Loop 1, data[0] is called, cache miss, leading to data[0] and data[1] getting stored in cache set 0. Till 16th iteration (till data[15]) cache fills out with data[0] (0x00) to data[15] (0xff).
   Moving to step 17, data[16] leads to cache miss, leading to 0x100 to 0x11f getting filled in cache set 0 again. This leads to cache getting overwritten similarly throughout from steps 17 to 32. This leads to cache miss and cache hit every two steps, giving 16 misses out of 32 calls.
   For loop 2, data[0] is called, cache miss (because data[16] to data[31] is stored in cache), leading to data[0] and data[1] being stored in cache set 0. Now data[0] is called again, and in iteration 2, data[1] is also called twice, all three leading to cache hits. Using same logic as above, in two iterations there is one cache miss and 3 hits, giving 16 misses out of 64 calls.
   Therefore cache miss rate is 32 out of 96 calls.

3. Given N = 16, $(m, S, E, B) = (32, 8, 2, 4)$, i.e. a single block in cache can store only one word, but there are 2 lines per set.

   Loop 1:
   Iteration 0: data[0] called, cache miss, block {0x00 - 0x0f} gets stored in cache set 0, line 0.
   Iteration 1: data[1] called, cache miss, block {0x10 - 0x1f} gets stored in cache set 1, line 0.
   Iteration 2: data[2] called, cache miss, block {0x20 - 0x2f} gets stored in cache set 3, line 0.
   ⋮
   Iteration 8: data[8] called, cache miss, block {0x80 - 0x8f} gets stored in cache set 0, line 1.
   Iteration 9: data[9] called, cache miss, block {0x90 - 0x9f} gets stored in cache set 1, line 1.
   ⋮
   Iteration 14: data[14] called, cache miss, block {0xe0 - 0xef} is stored in cache set 6, line 1.
   Iteration 15: data[15] called, cache miss, block {0xf0 - 0xff} gets stored in cache set 7, line 1.
   Loop 2:
   All of data array is already stored in cache, so all cache hits.
   This leads to 16 cache misses out of 48 calls.

Name: B.Anshuman

Roll No: 200259 e.g. 170001

Dept.: EE e.g. CSE

Indian Institute of Technology Kanpur
CS637 Embedded and Cyber-Physical Systems
Homework Assignment 3
Deadline: September 16, 2022