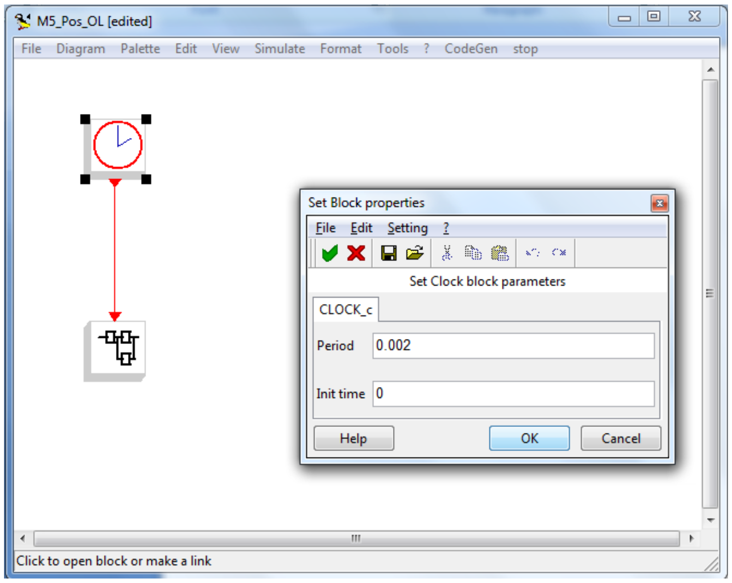
Experiment 1: Open Loop Motor Position

Objective: Understanding how sensors and actuators work.

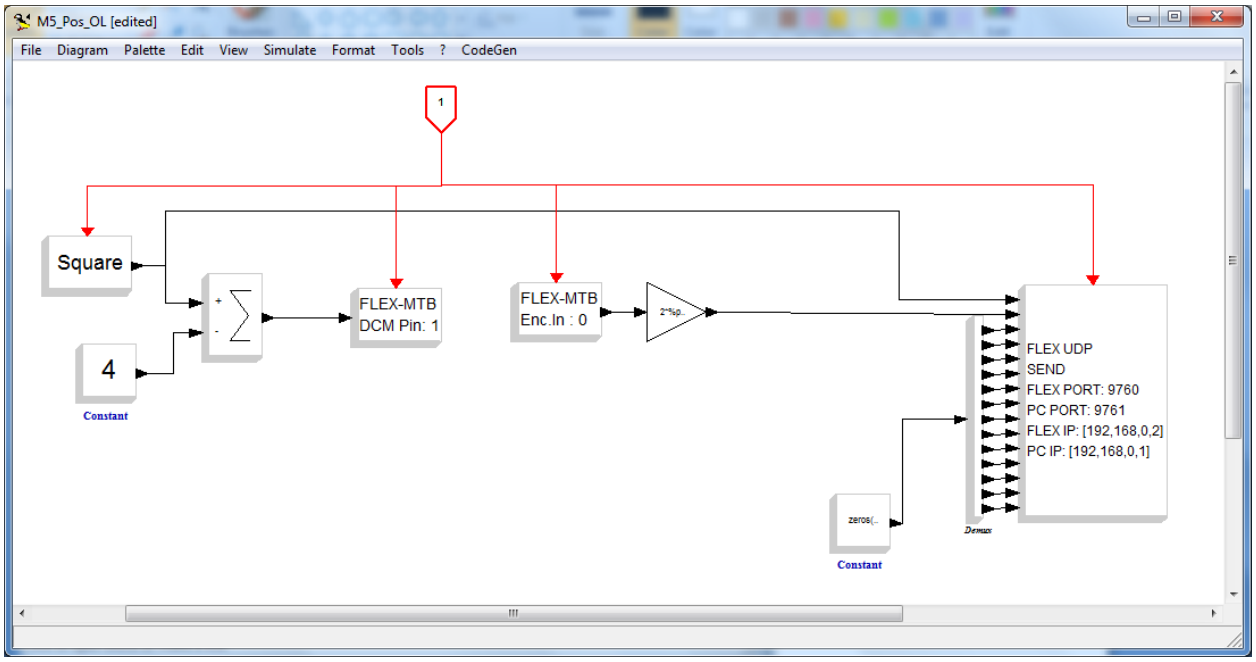

M5_Pos_OL [edited]
File Diagram Palette Edit View Simulate Format Tools ? CodeGen stop
Set Block properties
File Edit Setting ?
OK
Cancel
Help
Click to open block or make a link

Set Clock block parameters

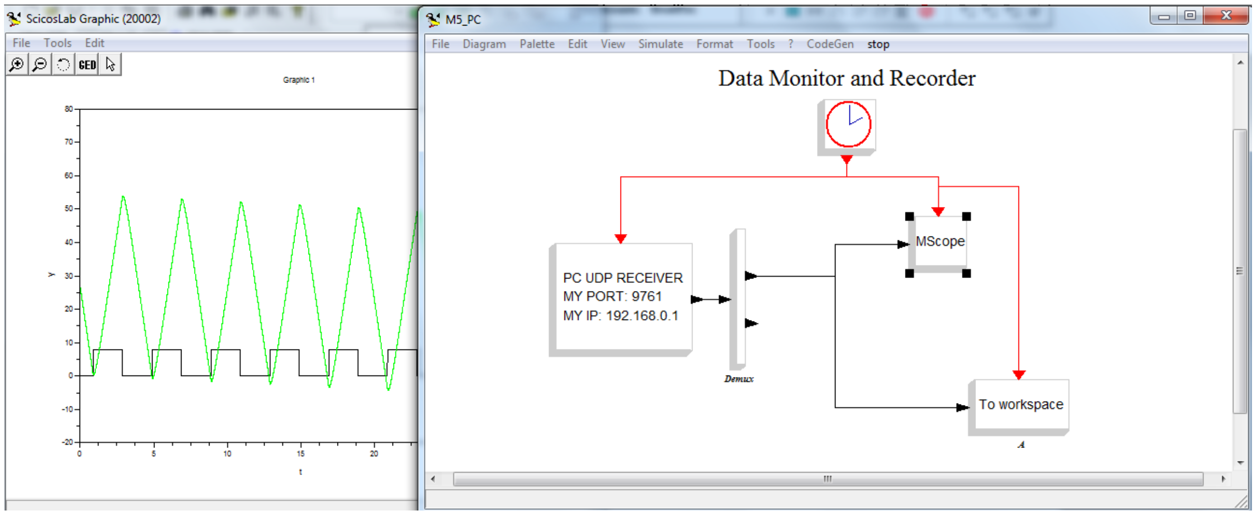
CLOCK_c

Period 0.002

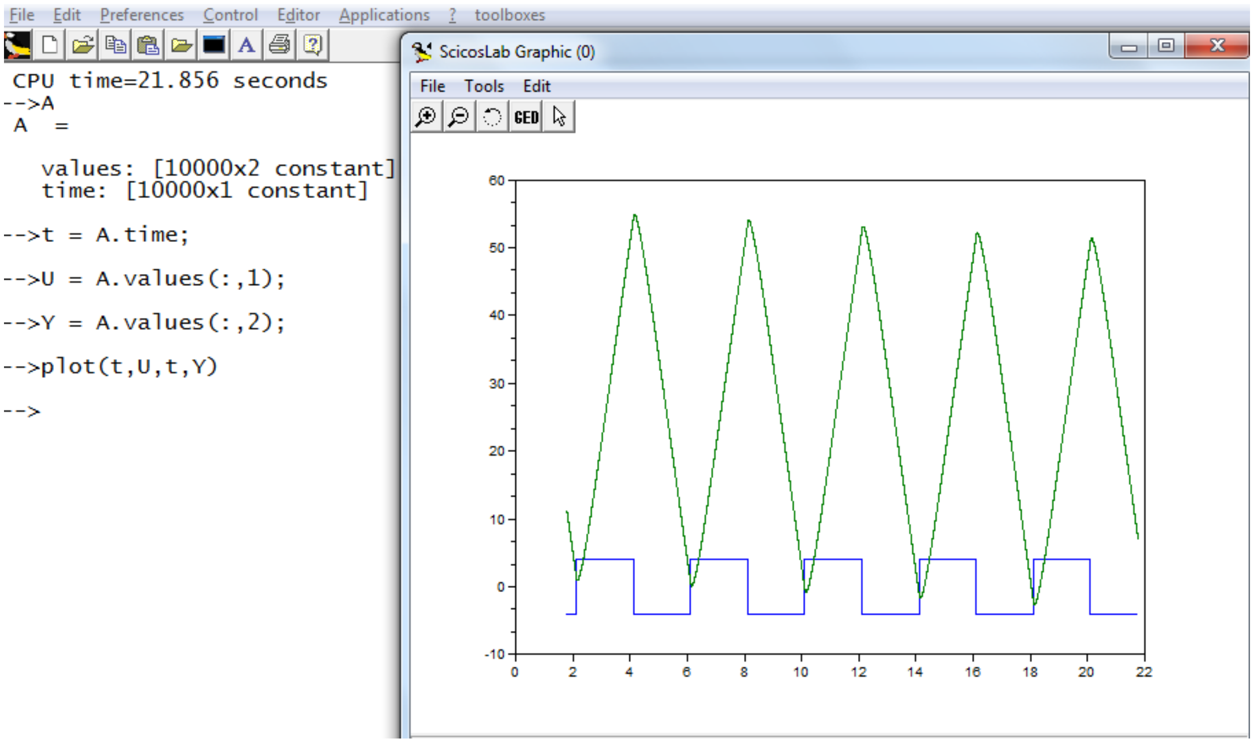
Init time 0



1. The model M5_Pos_OL.cos is compiled and run in the FLEX Demo2 Pack.
2. A square ware with amplitude -4 to 4 and period of 4 second is sent to drive the motor.
3. The position of the motor is sensed by the encoder and feed it back to the PIC.
4. The data for both input and output are sent back to computer through the TCPIP.


ScicosLab Graphic (20002)
File Tools Edit
Graphic 1
M5_PC
File Diagram Palette Edit View Simulate Format Tools ? CodeGen stop
Data Monitor and Recorder
PC UDP RECEIVER
MY PORT: 9761
MY IP: 192.168.0.1
Demux
MScope
To workspace
A

5. Model M5_PC is used to get the data back from the PIC, to scope as well as to Scicoslab workspace for system Identification.
6. The data is then extracted to variable t, U, and Y for system identification.

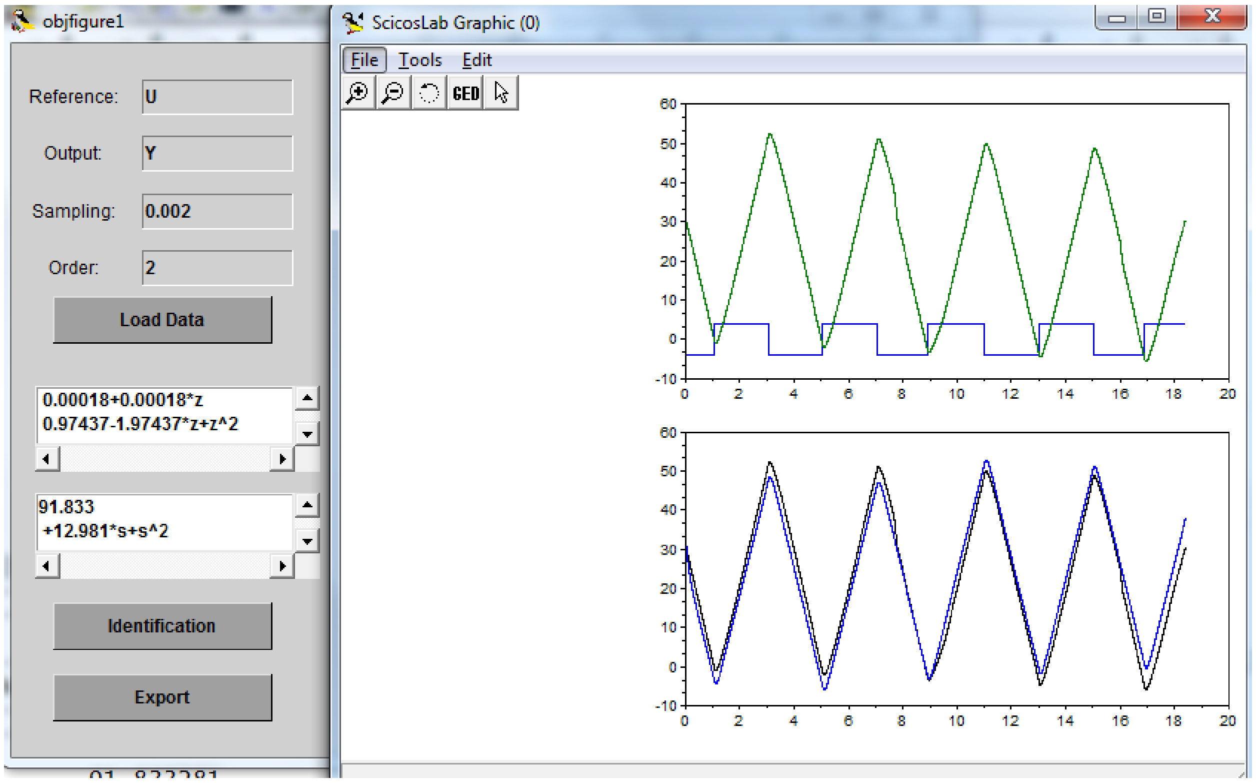

File Edit Preferences Control Editor Applications ? toolboxes
ScicosLab Graphic (0)
File Tools Edit

```
CPU time=21.856 seconds
-->A
 A  =

   values:  [10000x2 constant]
   time:  [10000x1 constant]

-->t = A.time;

-->U = A.values(:,1);

-->Y = A.values(:,2);

-->plot(t,U,t,Y)

-->
```

Experiment 2: System Identification (Data Driven Modeling) for DC Motor Position

Objective: Quick introduction to “black box” modeling in comparing with mathematic modeling.

1. With the Open Loop position data, launch the “sciIdent” for simple system identification GUI.
2. Make sure the fields Reference, Output, and Sampling match the variables you created.
3. Click Load Data button to verify that the data is imported correctly.
4. Click Identification to get the Model of the system via Identification method.
5. Click Export button to get the output echo on the scicoslab windows.



Reference: U

Output: Y

Sampling: 0.002

Order: 2

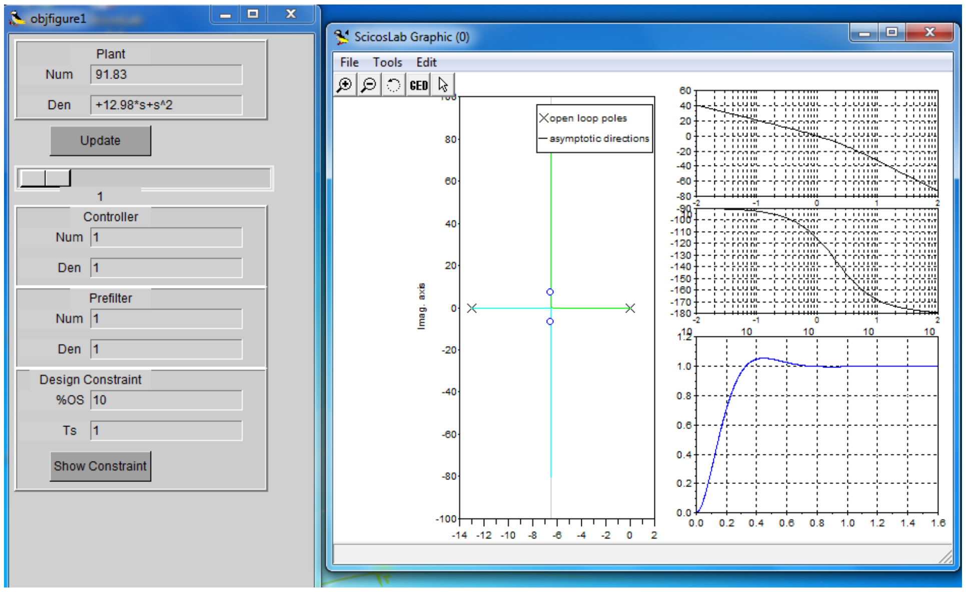
```
0.00018+0.00018*z
0.97437-1.97437*z+z^2
```

```
91.833
+12.981*s+s^2
```

Experiment 3: Controller Design

Objective: Controller Design Using RLDesign

1. By using the transfer function obtained from previous exercise, load the rldesign gui and call it using command rldesign(Gs), in which the Gs is the rational transfer function.
2. The following GUI will appear.



objfigure1

Plant
Num 91.83
Den +12.98*s+s^2

Update

1

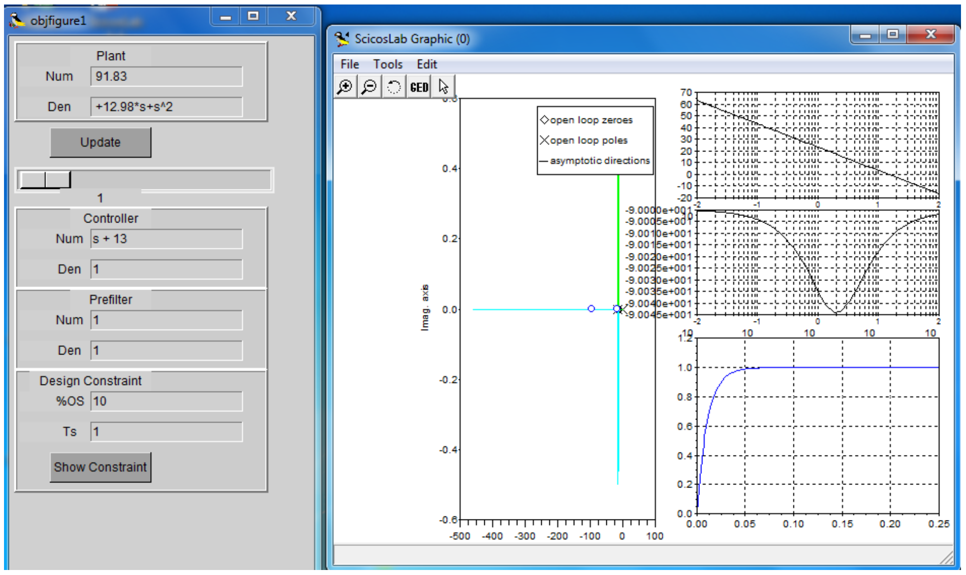
Controller
Num 1
Den 1

Prefilter
Num 1
Den 1

Design Constraint
%OS 10
Ts 1

Show Constraint

3. Try to add a zero at -13 location.



objfigure1

Plant
Num 91.83
Den +12.98*s+s^2

Update

1

Controller
Num s + 13
Den 1

Prefilter
Num 1
Den 1

Design Constraint
%OS 10
Ts 1

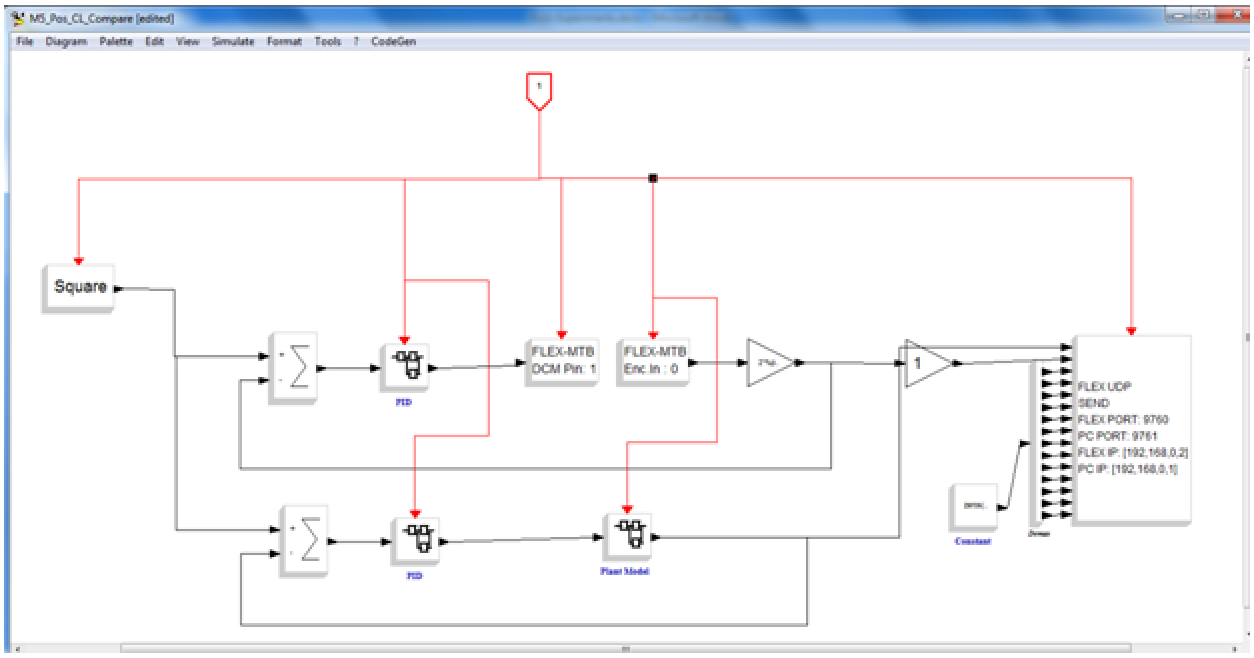
Show Constraint

4. The respond of the simulation seems to be faster and without overshoot.
5. We are going to try to implement the controller in the hardware and verify it.

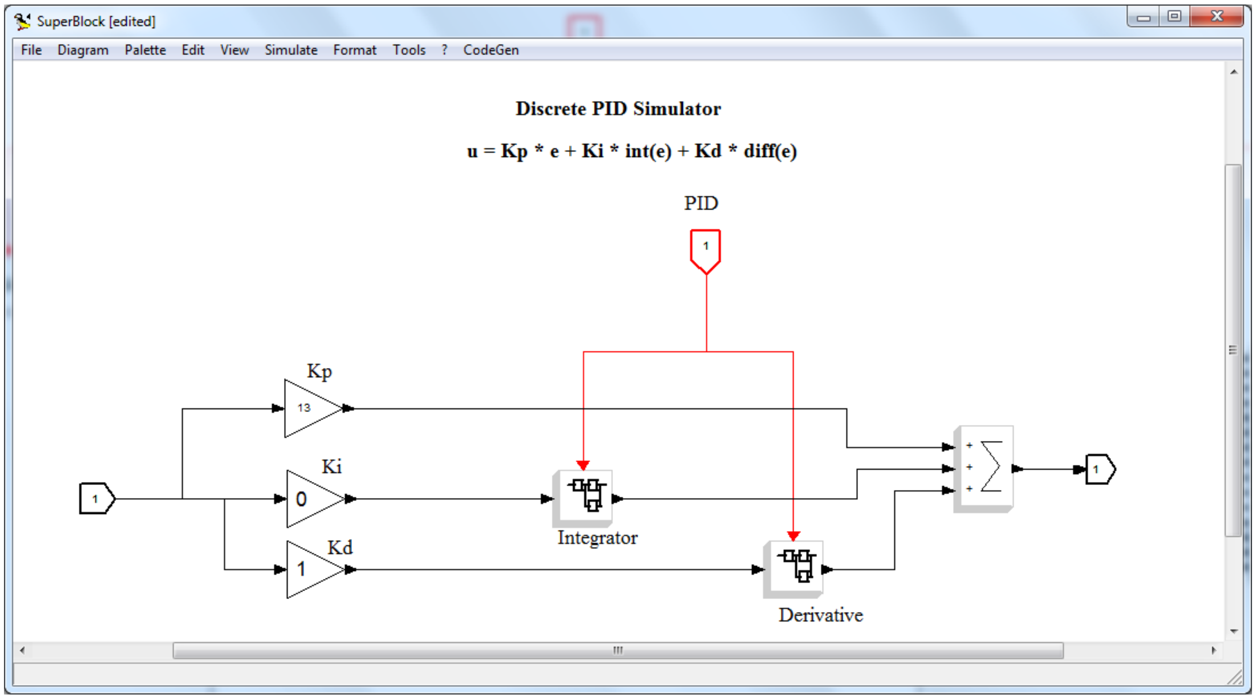
Experiment 4: Controller Implementation and Verification

Objective: To implement the controller in the hardware to verify with the simulation result

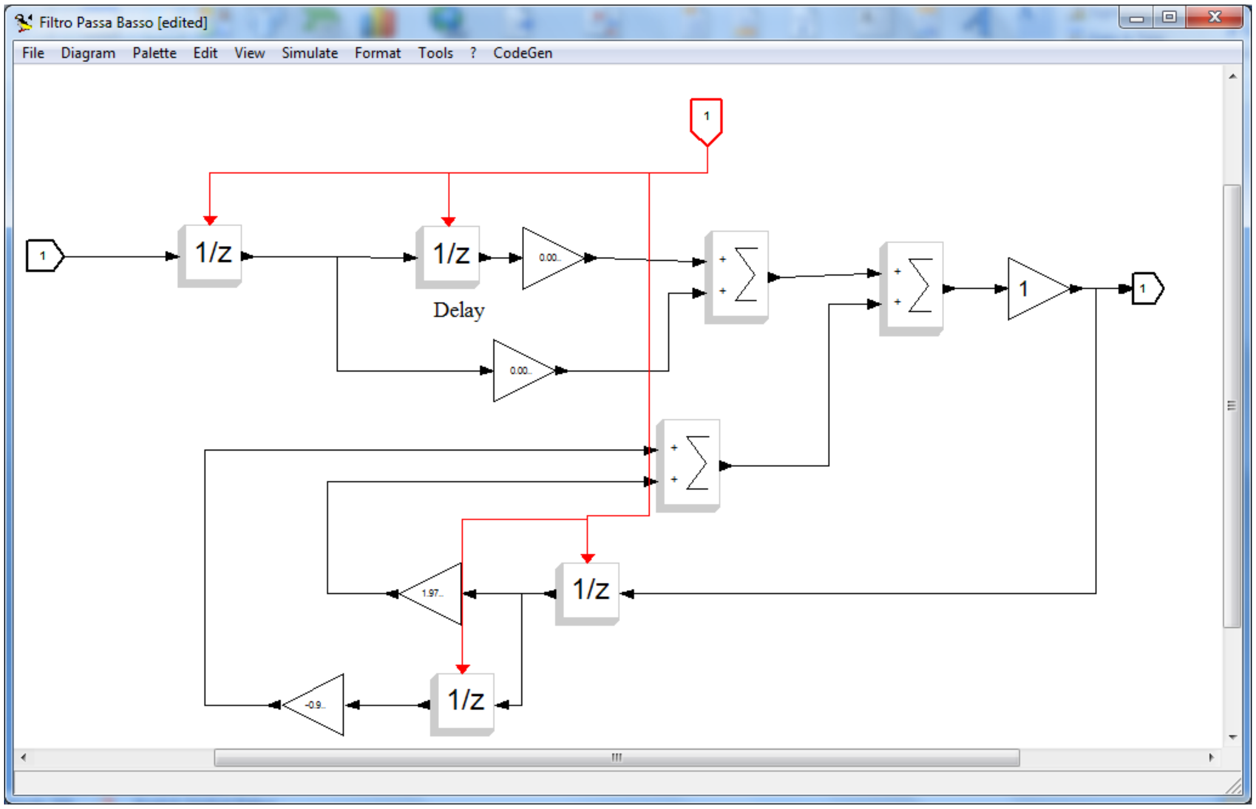
1. From the controller transfer function, we could derive that the Kp = 13 and Kd = 1.
2. Implement in the Scicos.


M5_Pos_CL_Compare [edited]
File Diagram Palette Edit View Simulate Format Tools ? CodeGen
Square
PID
FLEX-MTB DCM Pin: 1
FLEX-MTB Enc In : 0
1
FLEX UDP SEND FLEX PORT: 9760 PC PORT: 9761 FLEX IP: [192,168,0,2] PC IP: [192,168,0,1]
Constant
PID
Plant Model

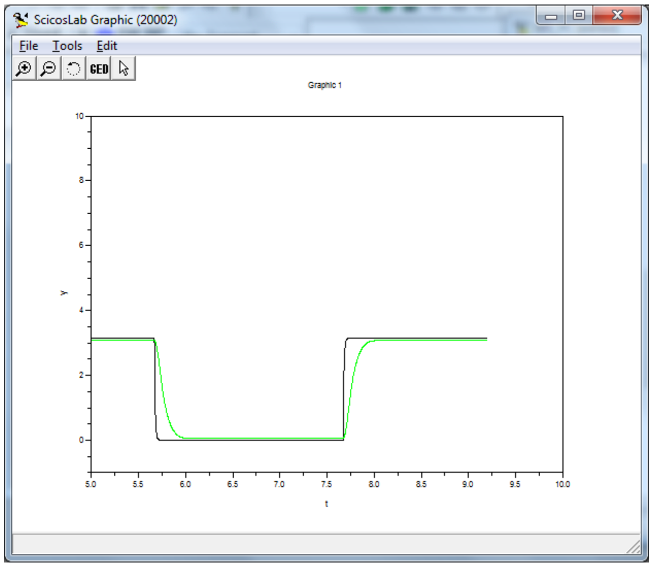
3. The PID for both real-time path and simulation path are identical.


SuperBlock [edited]
File Diagram Palette Edit View Simulate Format Tools ? CodeGen
Discrete PID Simulator
u = Kp * e + Ki * int(e) + Kd * diff(e)
PID
Kp
13
Ki
0
Kd
1
Integrator
Derivative

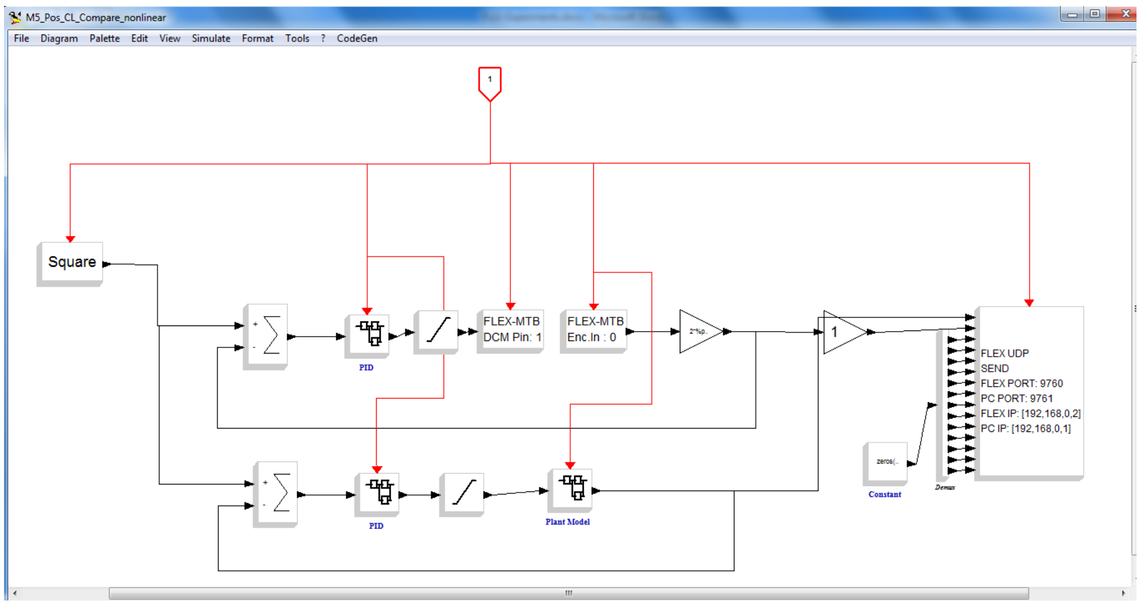
4. The model is constructed with the delay blocks, as shown below. It is equivalent to the z-domain transfer function from the previous experiment, which is:
   (0.00018 + 0.00018*z)/ (0.97437 - 1.97437*z + z^2), which could be also obtained by using dscr function to the s domain transfer function.


Filtro Passa Basso [edited]
File Diagram Palette Edit View Simulate Format Tools ? CodeGen
1
1/z
1/z
0.00..
Delay
0.00..
1
1
1.97..
1/z
-0.9..
1/z

5. Download the model into the FLEX and run the program. The motor now should move from 0 to 180 degree alternatively.
6. Using the M5_PC.cos, compare the result of real system and simulation.

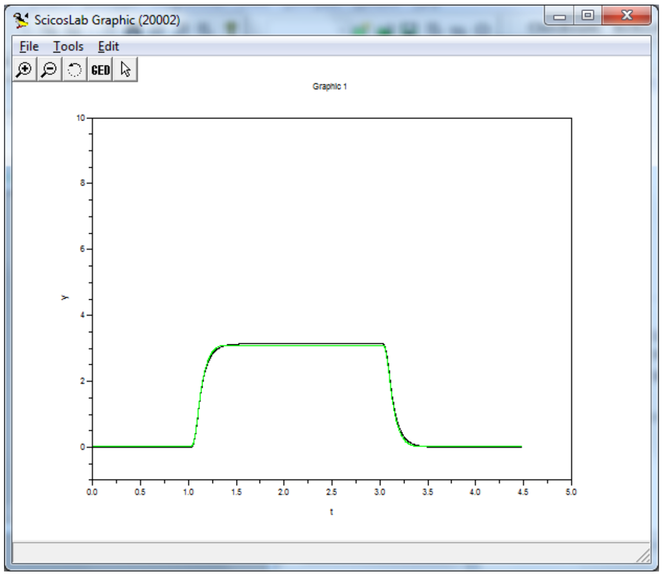


7. The black color line is the simulation result while the green is the real response. The transient response is quite different. This is due to the hardware limitation on the DC motor module, which is program to drive at maximum +- 6V.
8. To modify our model to a non linear model which includes the saturation of the controller output, we add the saturation block after the controller.



FLEX UDP
SEND
FLEX PORT: 9760
PC PORT: 9761
FLEX IP: [192,168,0,2]
PC IP: [192,168,0,1]

9. Repeat step 5-6 to compare the result again.



10. Now you could see the result match better, which concludes:
    a. Scicos is good to model non-linearity in real system
    b. The controller that we design works!