ECE486 Lab 4: DC Motor & PID Control

Scope and Objective

1. Study the basics of motion control in DC motors
   - Understand the working principle of a DC servo motor and its components
   - Model the DC servo motor as an electromechanical system with feedback control
2. Apply control theory to the design of a motion control system
   - Analysis the conditions for system stability
   - Design the appropriate PID control based on desired transient response specifications

Prelab Exercise

Please read prelab reading before coming for the lab. The information on sensors and actuators will be important to your understanding and benefits your learning outcome for this lab.

Introduction

Motion control is an important application in the automation industry involving an extensive study of system dynamics and control theory. The applications often involve a combination of electrical and mechanical systems to accomplish mechanical work. The DC servo motor is an example of such electromechanical system. The term servo refers to feedback control mechanism which typically comprises components like actuators (DC motors), sensors (encoders) and the controller.

In this lab, we will look at motion control. Starting from the understanding of the working principle of the components in the servomechanism through theoretical modeling and physical operations, we will extend the derived model to facilitate the designing of appropriate controllers. This will also pave the path for subsequent lab activities involving the application of the knowledge related to control theory for motion control.

Equipment

You will be introduced to the DC Servo System (Googol Technology Inc., China) in Control Lab @ ZJUI as shown in Figure 1. This is a servo-system that allows you to acquire hands-on experience in apply the modeling, analysis and design technique you learned in class for motion control.

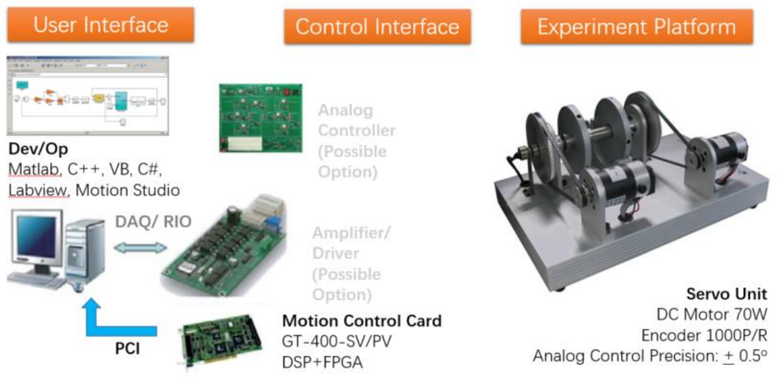


Figure 1: DC Servo System in Instructional Control Lab @ ZJUI

Lab Activities

1. Demonstration of the basics in operating the DC Servo Unit.

- Your instructor(s) will demonstrate simple calibration and acquiring of the system parameters using EasyMotion Studio. You need not learn to use the software. However, you should understand the concepts behind the operations.
- Two simple test drive modes will be demonstrated and you may later tryout yourself.

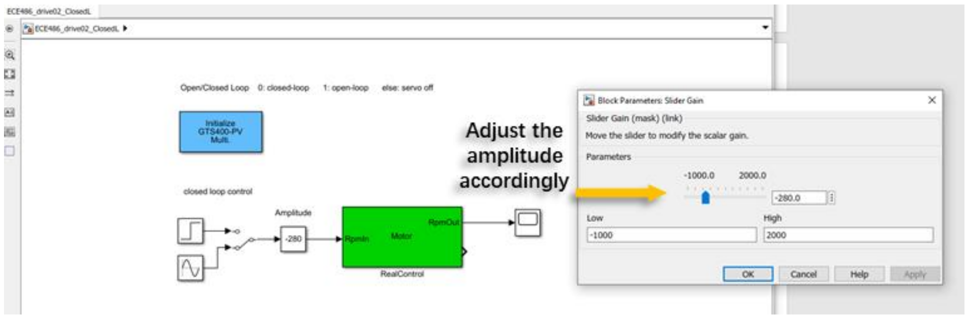

ECE486_drive02_ClosedL
Open/Closed Loop 0: closed-loop 1: open-loop else: servo off
Initialize GTS400-PV Multi
closed loop control
Amplitude
-280
RpmIn
Motor
RpmOut
RealControl
Adjust the amplitude accordingly
Block Parameters: Slider Gain
Slider Gain (mask) (link)
Move the slider to modify the scalar gain.
Parameters
-1000.0
2000.0
-280.0
Low
-1000
High
2000
OK
Cancel
Help
Apply

Figure 2: Closed-loop Test Drive Program (useful for testing and troubleshooting Error)

2. Simulation of the model and controller

- Try out Section 2 of Chapter X (Googol Technology Inc., China)
- At the end of the simulation, your program should allow you to play around with different choices of PID term to intuitively understand the effect of each of them

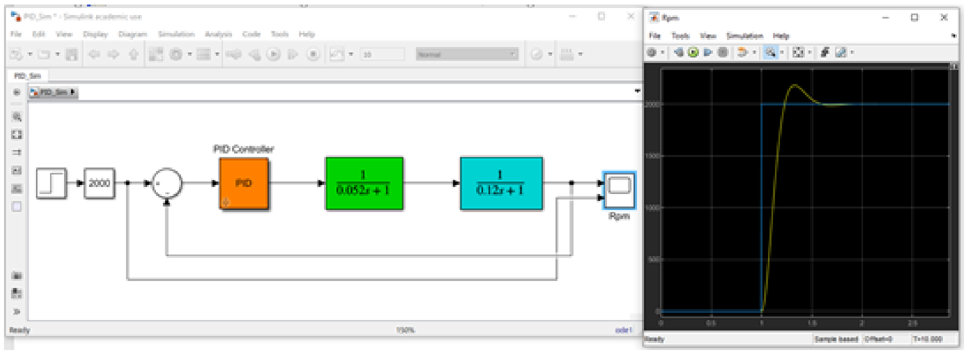


Figure 3: Your completed program should allow you to simulate different parameters

3. Implementation of PID controller and analysis on the DC Servo System

- Try out Section 3 of Activity Chapter X (Googol Technology Inc., China)
- Run ECE486_drive_OpenL and toggle the manual switch to send a ramp input (**Figure 4**)
- Observe the relationship between the voltage input and the output RPM. Given an intuitive explanation to your observation.

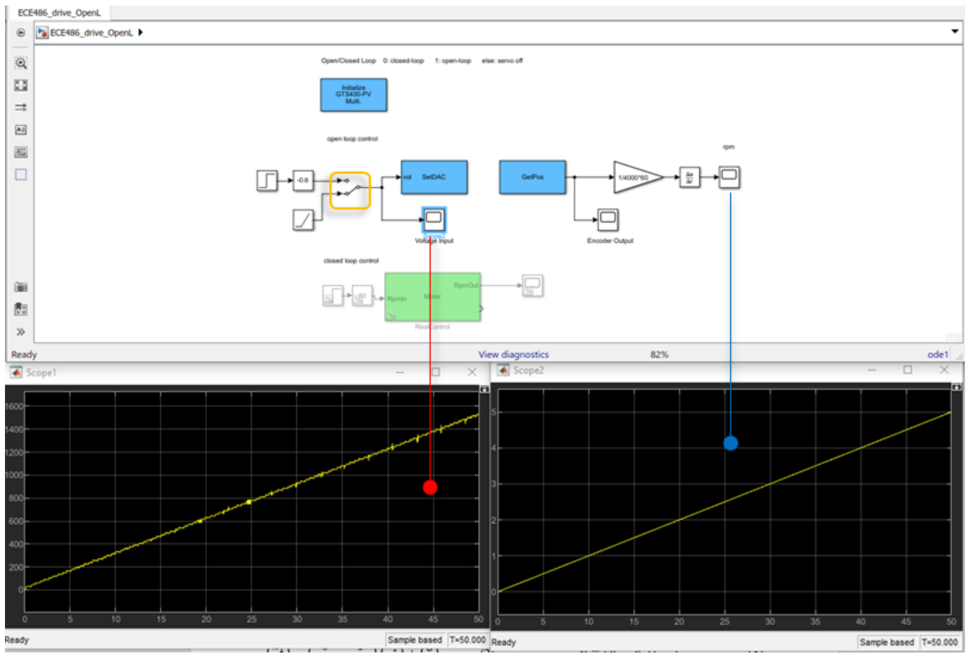
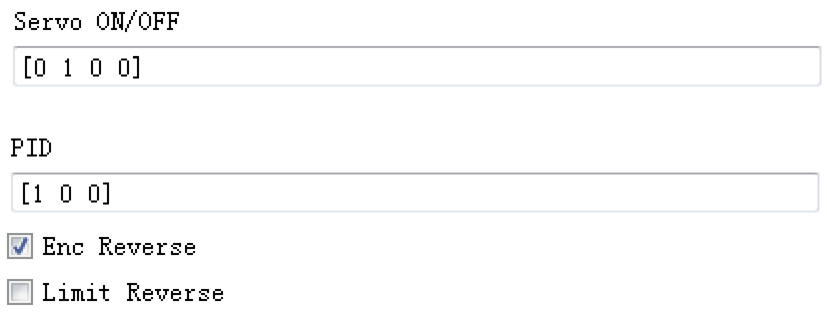


Figure 4: Input voltage and the output rpm

Chapter X PID Calibration

Attention

The "gts400-pv Initialization" module is set as follows:



Servo ON/OFF

[0 1 0 0]

PID

[1 0 0]

☑ Enc Reverse

☐ Limit Reverse

I. Experiment Purpose

1． Calibrate the DC servo motor system with PID method;

2． Design and verify such calibrated link.

II. Experiment Requirements

1. Based on the given performance index, use trial-and-error method to design the PID calibration link, and calibrate the uncorrected ones, then verify such calibration.

2. Set the open-loop transfer function of the uncorrected system as $G_0(s) = \dfrac{1}{(0.052s+1)(0.12s+1)}$, then design the PID calibration link, and make the performance index of the system reach $t_s \le 1.5$ s, $\delta_p \le 4.3\%$, and static error 0.

III. Experiment Device

1. GSMT2014 DC servo system control platform;

2. PC and MATLAB platform

IV. Experiment Principles

1. PID introduction

There are various control algorithms of PID, with each aiming at one specific filed. The Fig. 10.1, Fig. 10.2 and Fig. 10.3 present three different algorithms.

In simulating control system, the most commonly used control law is PID control. The simulated PID control system block diagram is shown as Fig. 10.1, and the system is composed of simulated PID controller and object.

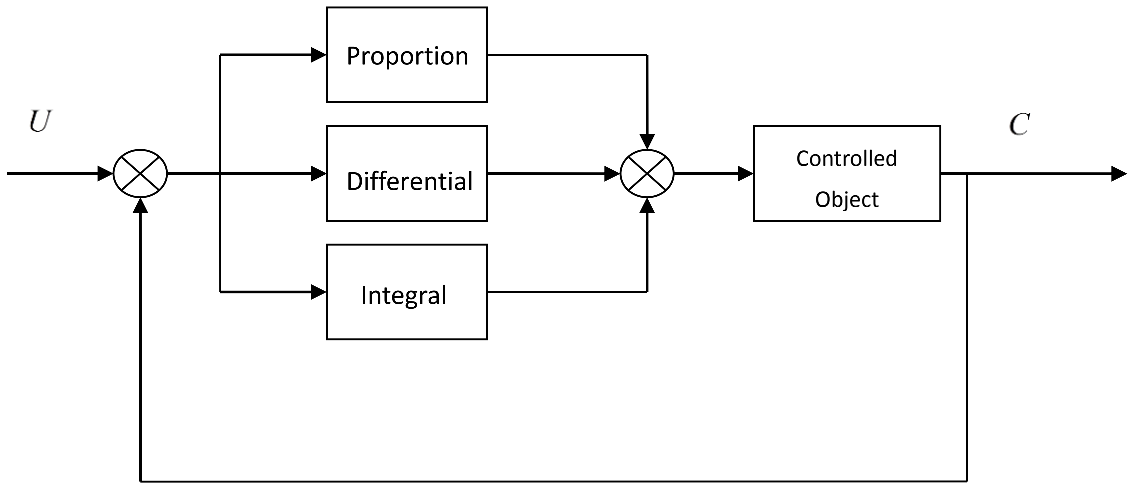


Figure. 10.1 Principle Block Diagram of Simulated

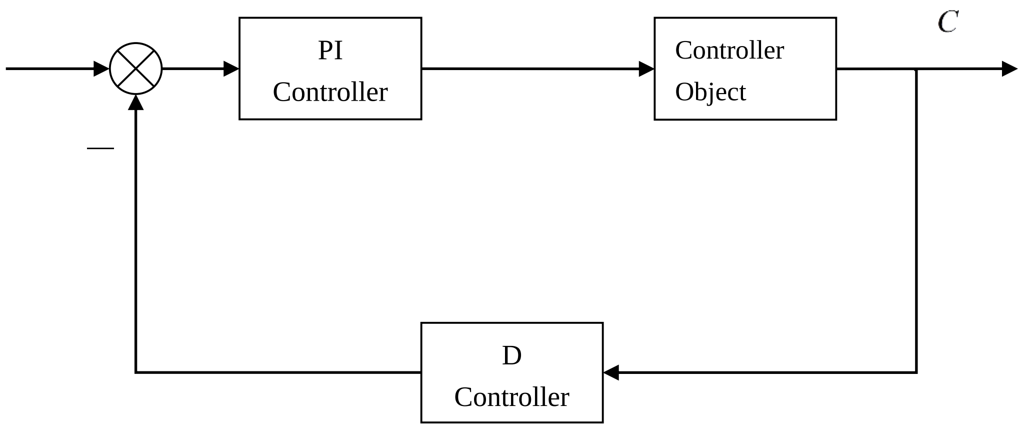


Fig. 10.2 Differential Antecedent PID Control Schematic Diagram

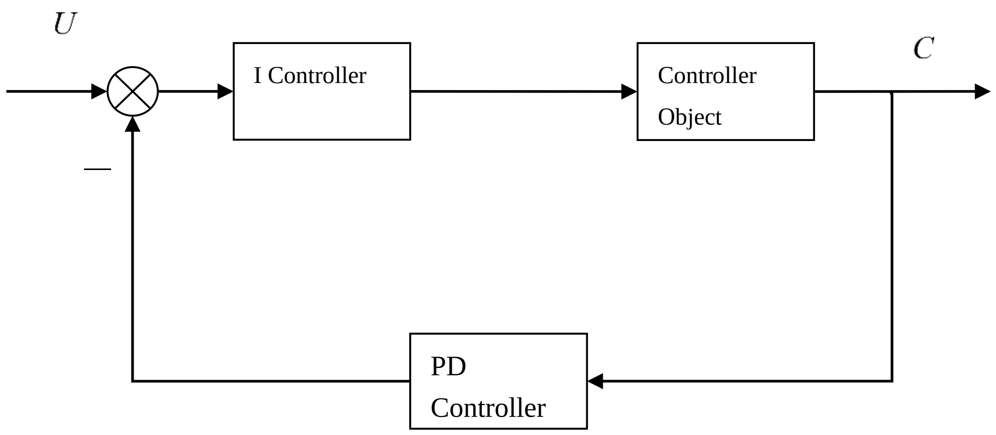


Fig. 10.3 Puppet PID Control Schematic Diagram

As a kind of linear controller, the *PID* controller shall constitute control deviation $e(t)$ between the given value

$r(t)$ and actual output value $y(t)$

$$e(t) = r(t) - y(t) \tag{10.1}$$

The PID controller is so called because it combines Proportion (P), Integral (I) with Differential (D) of the deviation via linear means, and constitutes the control variable to have the object controlled. The law is:

$$u(t) = K_P\left[e(t) + \frac{1}{T_I}\int_0^t e(t)dt + T_D\frac{de(t)}{dt}\right] \tag{10.2}$$

Or in the form of transfer function:

$$G(s) = \frac{U(s)}{E(s)} = K_P\left(1 + \frac{1}{T_I s} + T_D s\right) \tag{10.3}$$

Where, K_P means proportion coefficient, T_I integral time constant, and T_D differential time constant.

In control system design and simulation, the transfer function is also written in the form of:

$$G(s) = \frac{U(s)}{E(s)} = K_P + \frac{K_I}{s} + K_D s = \frac{K_D s^2 + K_p s + K_I}{s} \tag{10.4}$$

Where, K_P means proportion coefficient, T_I integral coefficient, and T_D differential coefficient. Seen from the standpoint of root locus, formula (2.10) is equivalent to adding a pole located at base point and two null points with variable location to the system.

To be simple, each calibration sector of the *PID* controller works as below:

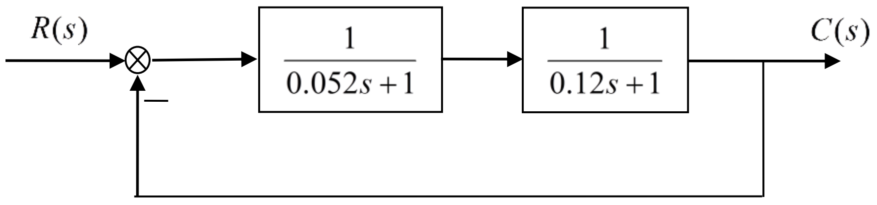
A. Proportion sector: it reflects the deviation signal $e(t)$ of the control system in proportion, the controller shall work as soon as the occurrence of the deviation to reduce it.

B. Integral sector: It is mainly used to eliminate the static error and improve system type level. The role of integral is positively correlated to the integral time constant T_I.

C. **Differential sector:** It reflects the change trend (change rate) of the deviation signal, introduces an effective early correction signal into the system before the deviation signal enlarges, so as to speed up the process speed and shorten tuning time.

2. Tuning of PID parameters with trial-and-error method

Set the structural diagram of the DC servo motor's control system as the figure below:

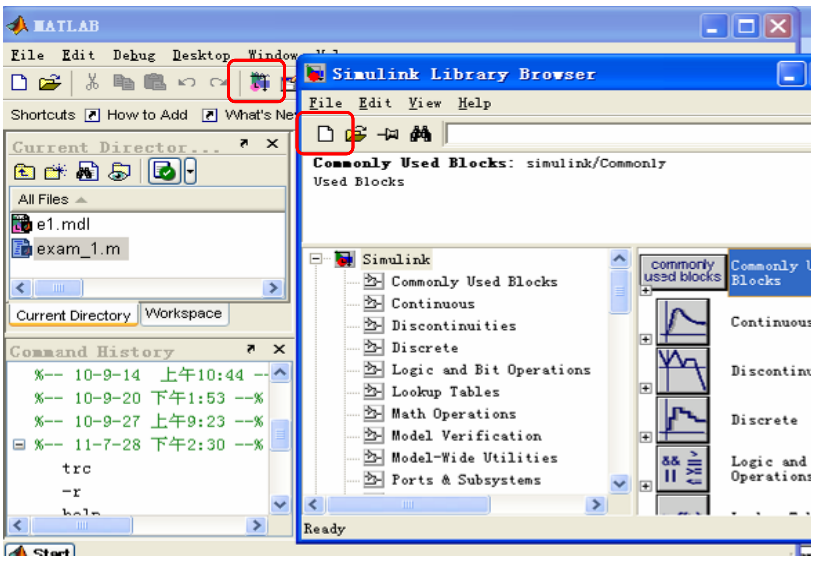


At the input speed of 2000rpm, the performance index of the uncorrected system is: static value 1000rmp, static error 50%, overshoot $\sigma = 2.8\%$, tuning time $t_s = 0.3121$s. The PID calibration link is required to design with the trial-and-error method to tune the parameters, then $t_s \le 1.5$ s, $\delta_p \le 4.3\%$, and static error 0.

V. Experiment Procedures

1. Simulink simulation of uncalibrated system

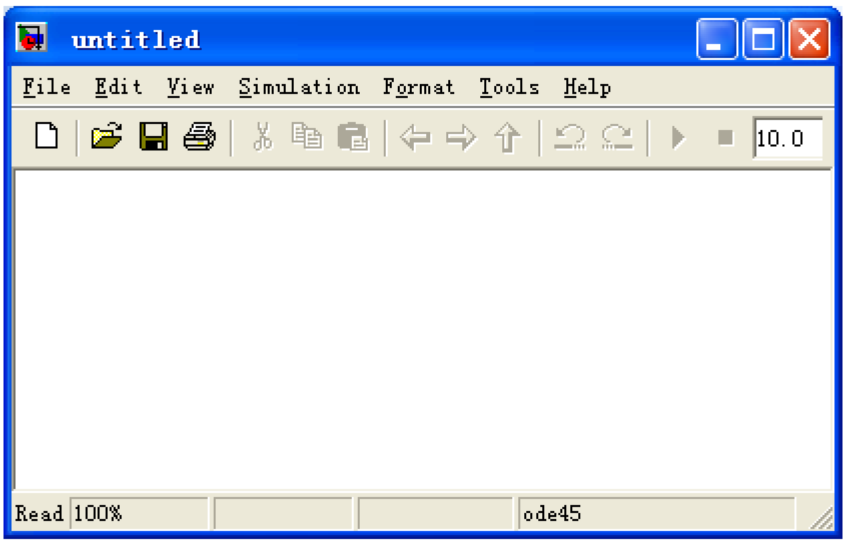
1) Open simulation program in Simulink:



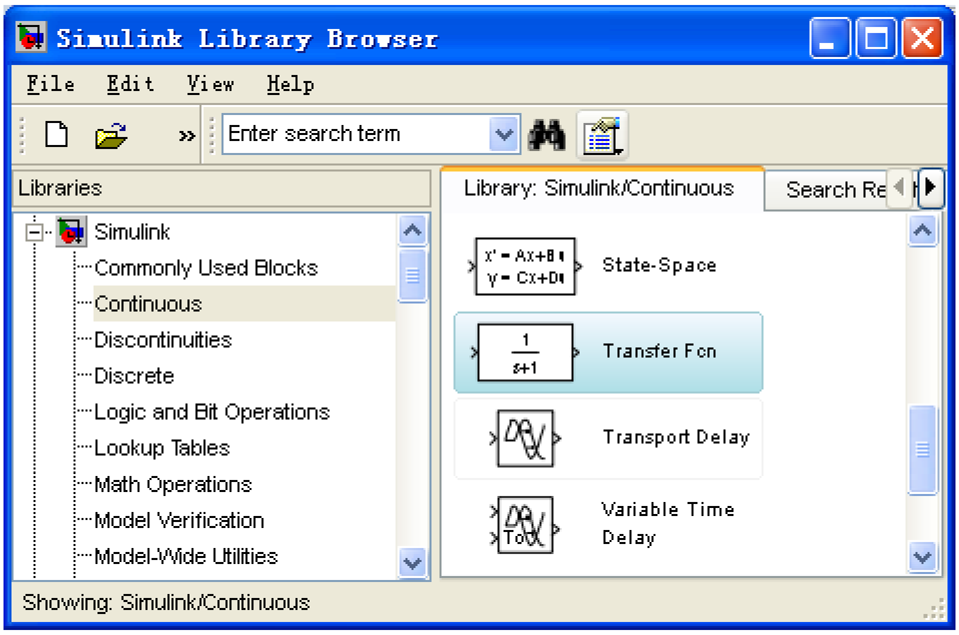
Commonly Used Blocks: simulink/Commonly Used Blocks

```
%-- 10-9-14  上午10:44 --%
%-- 10-9-20  下午1:53 --%
%-- 10-9-27  上午9:23 --%
%-- 11-7-28  下午2:30 --%
    trc
    -r
```

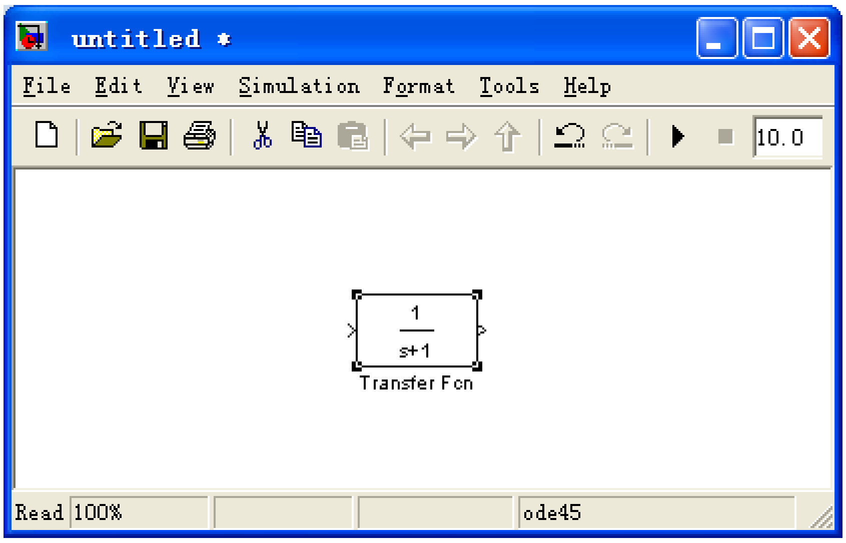
2) Click “🗋”in the upper left corner of the window to create a new “Model” in window:



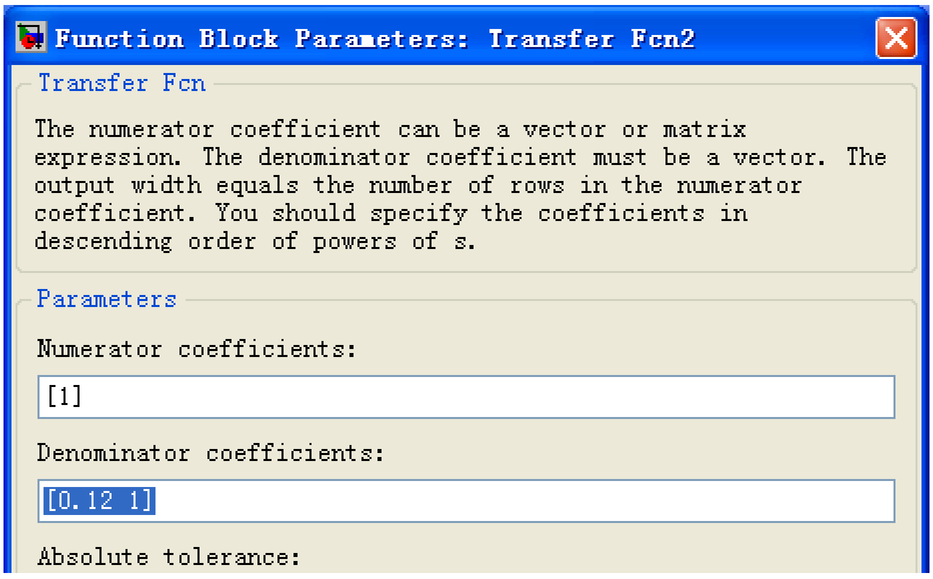
3) In the "Simulink Library Browse" window, open the "Simulink\ Continuous" window, as shown below:



4) Drag the "Transfer Fcn" module into the newly created "untitled" window:



5) Double-click the "Transfer Fcn" module and open the following window. The parameters are set as follows:


Function Block Parameters: Transfer Fcn2

Transfer Fcn

The numerator coefficient can be a vector or matrix expression. The denominator coefficient must be a vector. The output width equals the number of rows in the numerator coefficient. You should specify the coefficients in descending order of powers of s.

Parameters

Numerator coefficients:

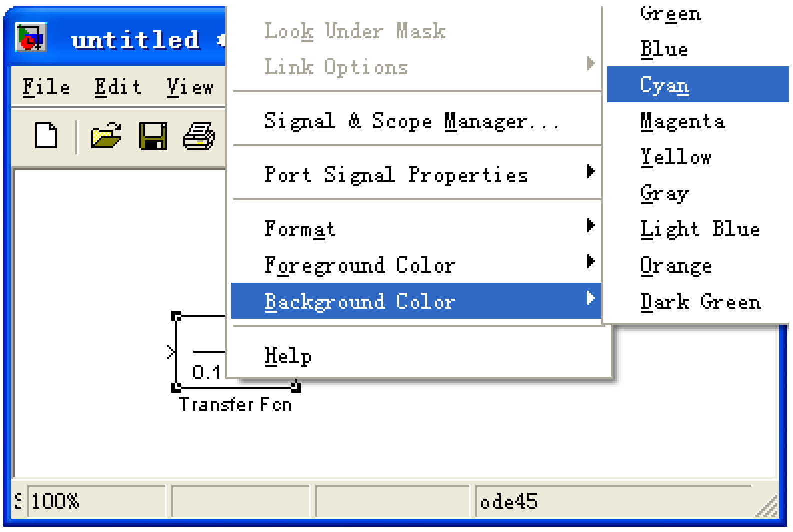
[1]

Denominator coefficients:

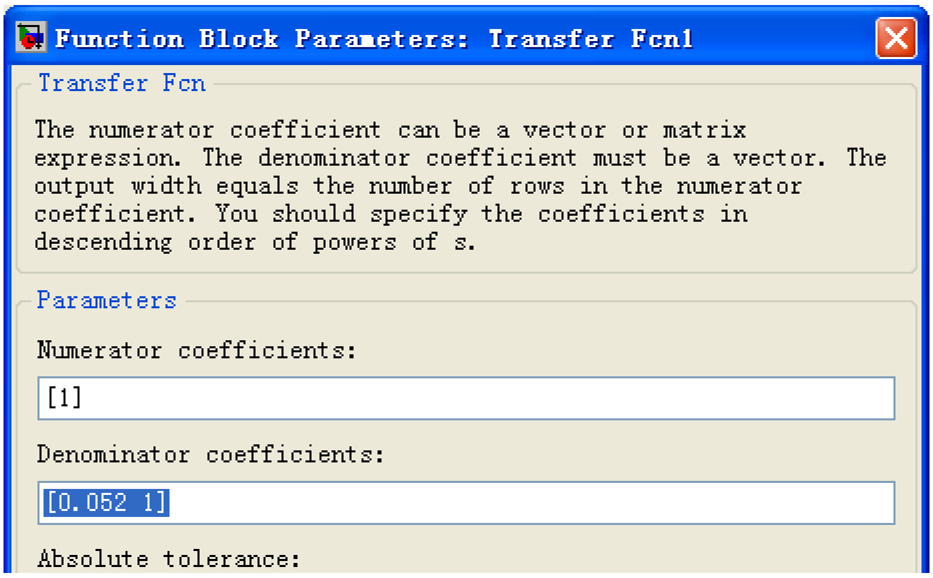
[0.12 1]

Absolute tolerance:

6) Right-click on the "Transfer Fcn" module and set "Background Color" to Cyan.



7) Another copy of the "Transfer Fcn" module, double-click to set the parameters as shown in the figure below, and set its background color as Green



Transfer Fcn

The numerator coefficient can be a vector or matrix expression. The denominator coefficient must be a vector. The output width equals the number of rows in the numerator coefficient. You should specify the coefficients in descending order of powers of s.

Parameters

Numerator coefficients:

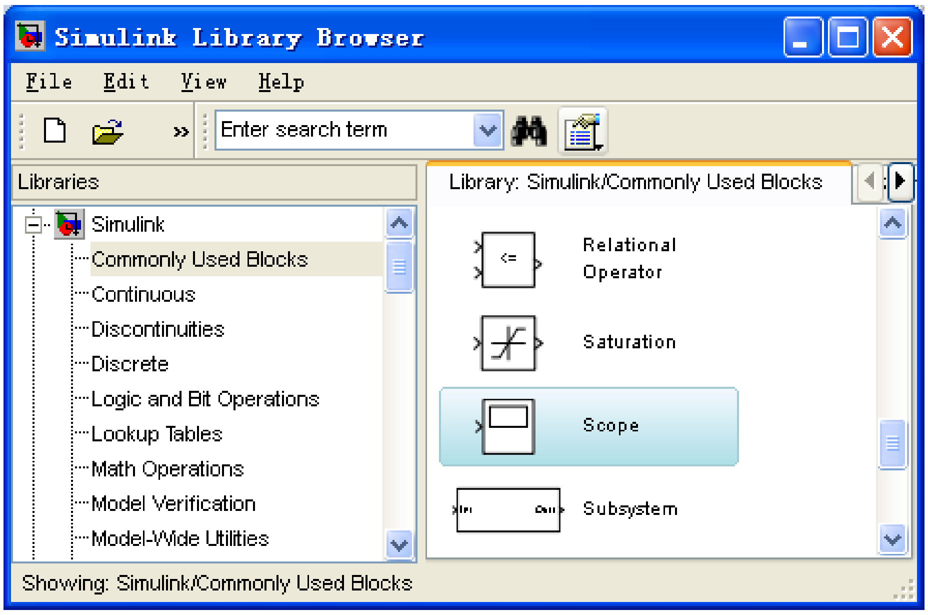
[1]

Denominator coefficients:

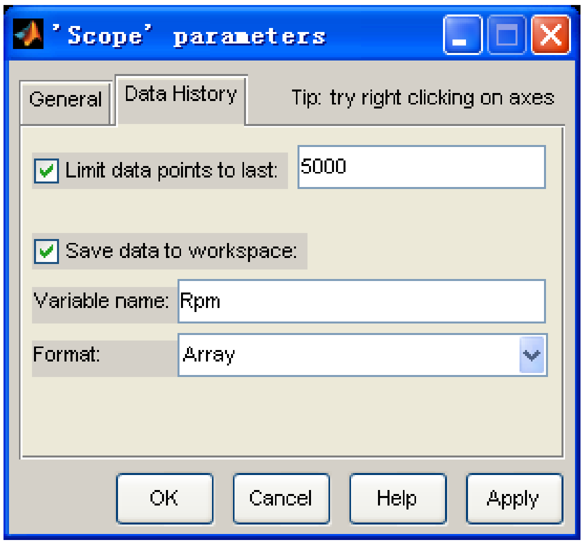
[0.052 1]

Absolute tolerance:

8) Drag from the "Simulink \ Sinks" a "Scope" to "untitled" window:



9) Open" Scope", click 📋, and check "Save data to workspace" . Set "Variable name" to user-defined, and "Format" to Array.


'Scope' parameters

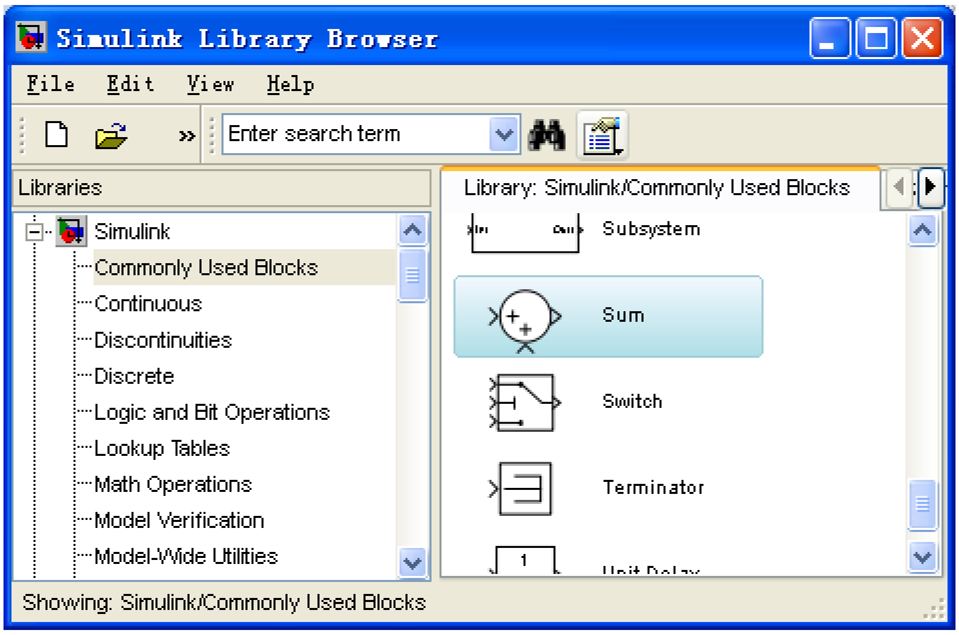
☑ Limit data points to last: 5000

☑ Save data to workspace:

Variable name: Rpm

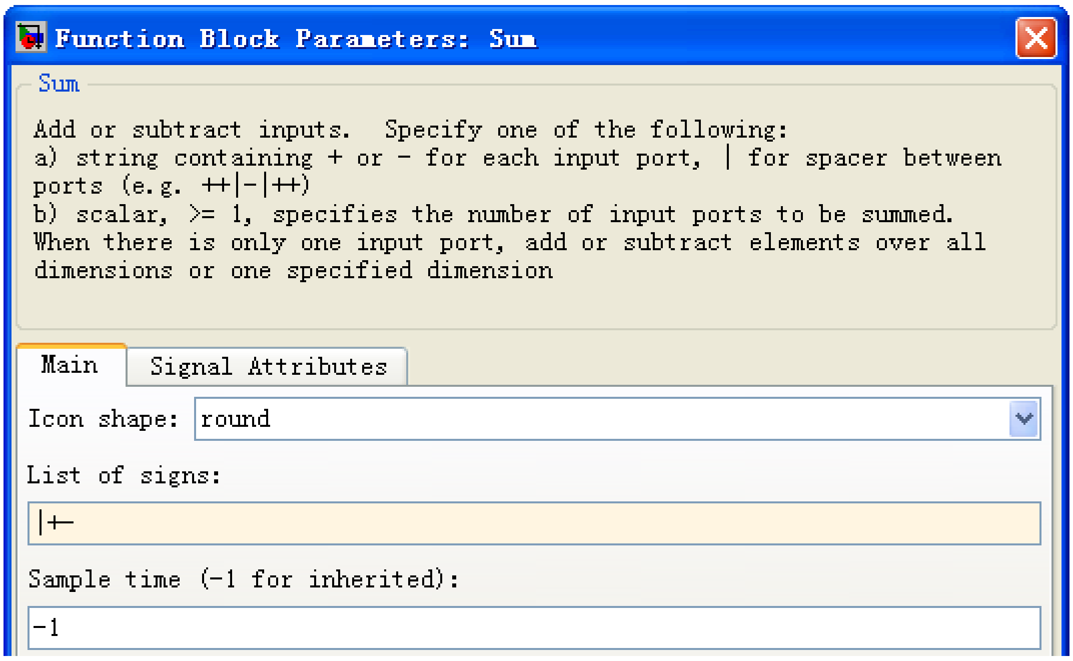
Format: Array

10) Drag from the " Simulink\ Commonly Used Blocks " a "Sum" to "untitled" window:


Simulink Library Browser

Showing: Simulink/Commonly Used Blocks

11) Double-click the "Sum" module and open the following window. The feedback setting is as follows:



Add or subtract inputs. Specify one of the following:
a) string containing + or - for each input port, | for spacer between ports (e.g. ++|-|++)
b) scalar, >= 1, specifies the number of input ports to be summed.
When there is only one input port, add or subtract elements over all dimensions or one specified dimension

Icon shape: round

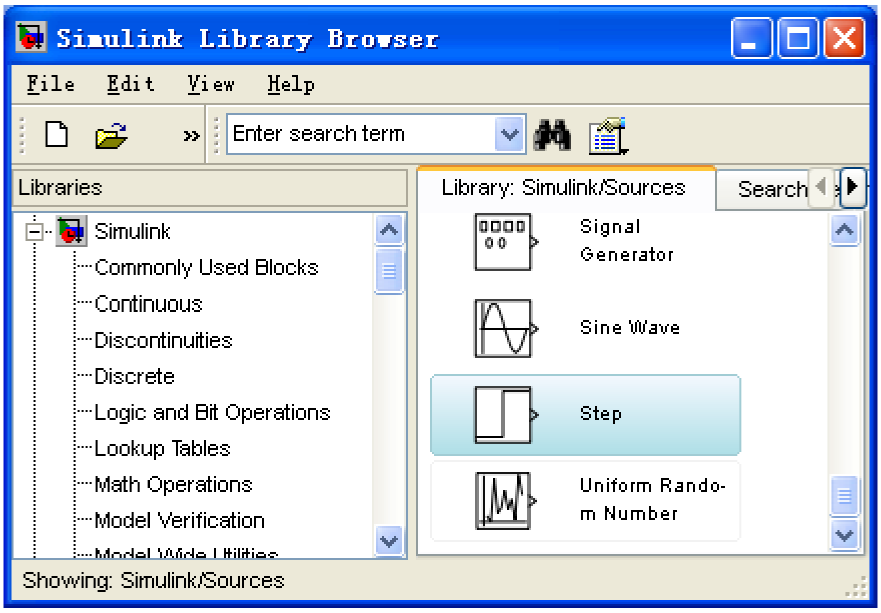
List of signs:

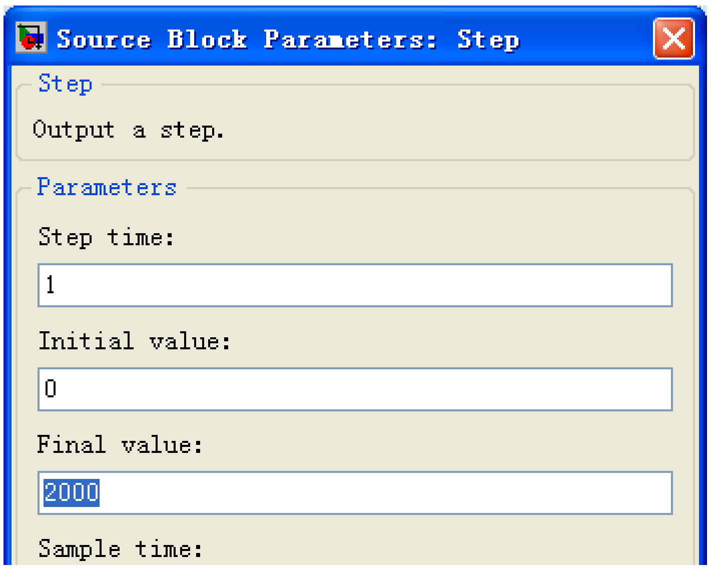
|+-

Sample time (-1 for inherited):

-1

12) Drag from the " Simulink\ Sources " a "Step" to "untitled" window, and set the Final Value is 2000





Source Block Parameters: Step

Step

Output a step.

Parameters

Step time:

1

Initial value:

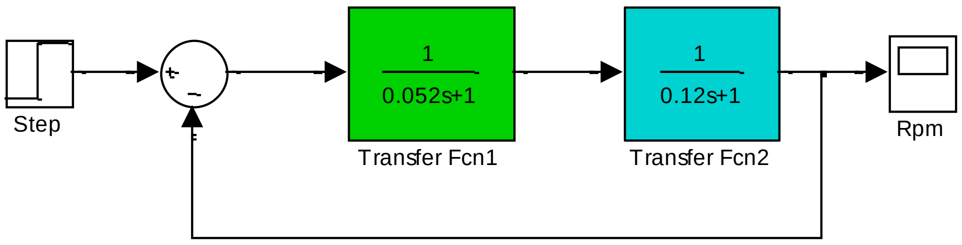
0

Final value:

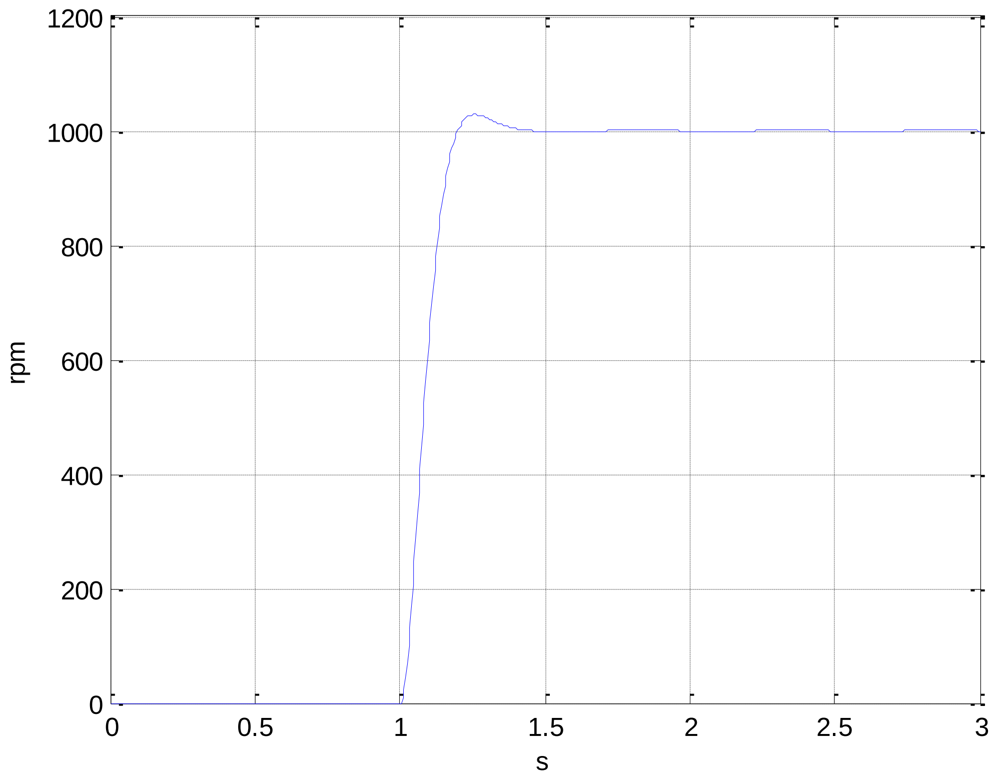
2000

Sample time:

13) Connect the five modules as shown below and save the file as " UncorSys_Sim " with the default format of MDL.



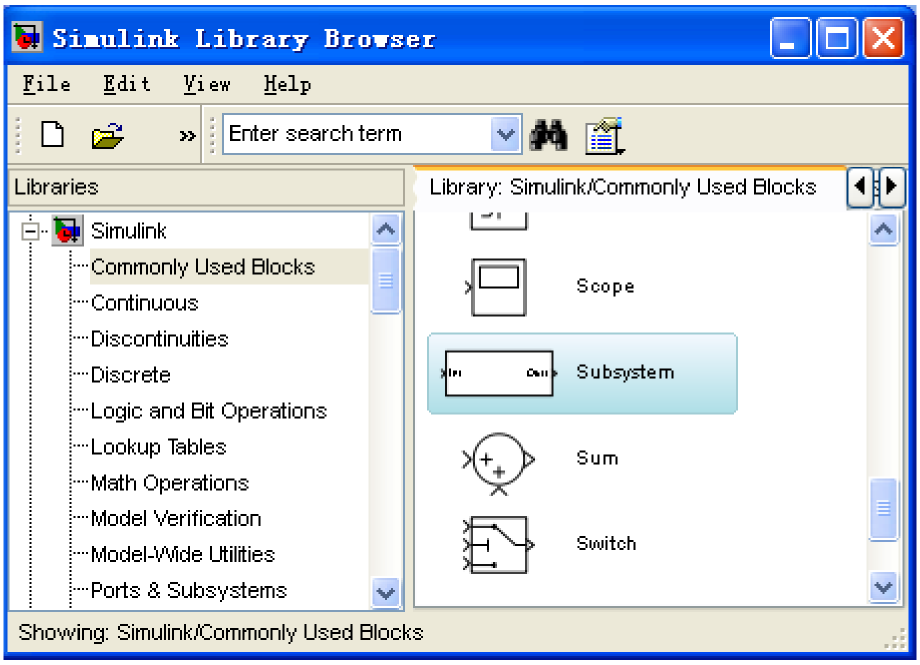
14) Click the button "▶" and double-click the Scope module to get the system simulation curve.

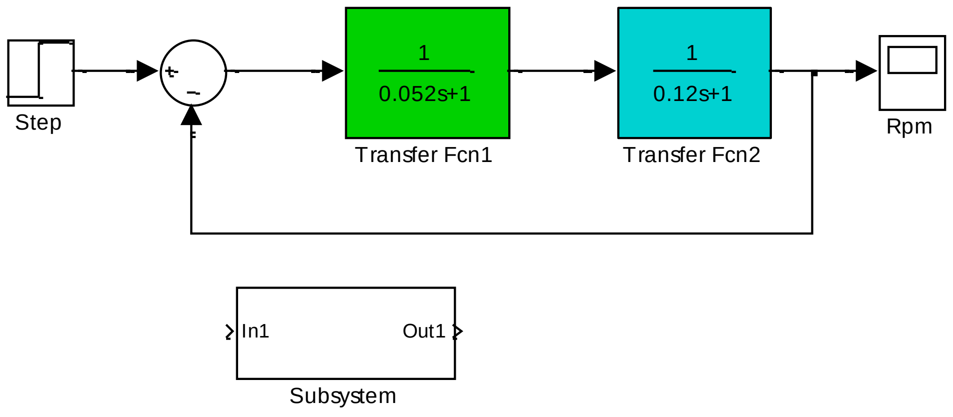


15) The closed loop system is stable with a steady-state value of 1000rpm, a steady-state error of 50%, overshoot amount of $\sigma = 2.8\%$, and adjustment time of $t_s = 0.312$ seconds.

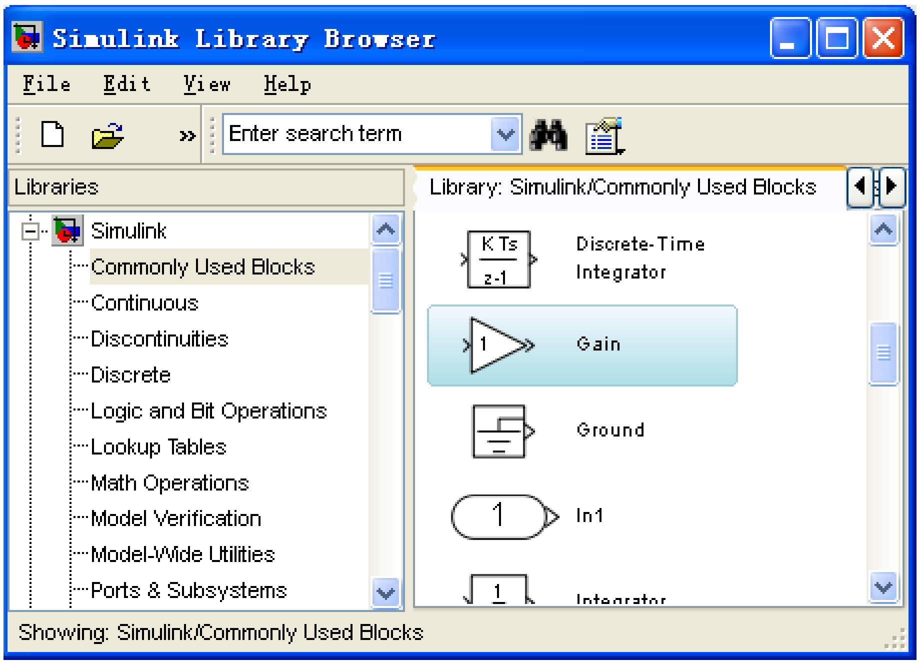
2. Build the system simulation program with calibration link in the Simulink as shown below:

1) Drag a "Subsystem" from Simulink Library module to UncorSys_Sim. MDL.

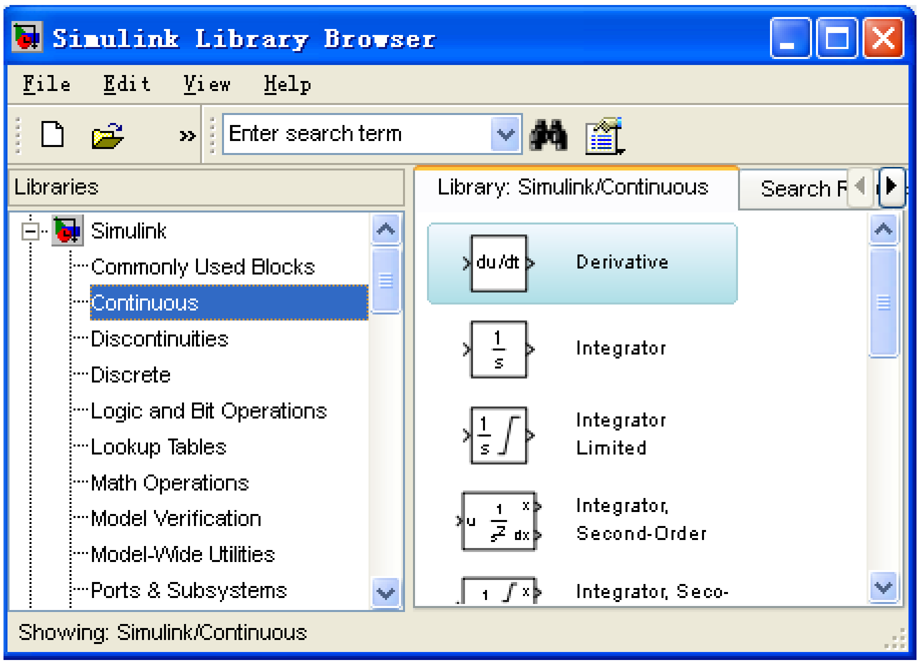


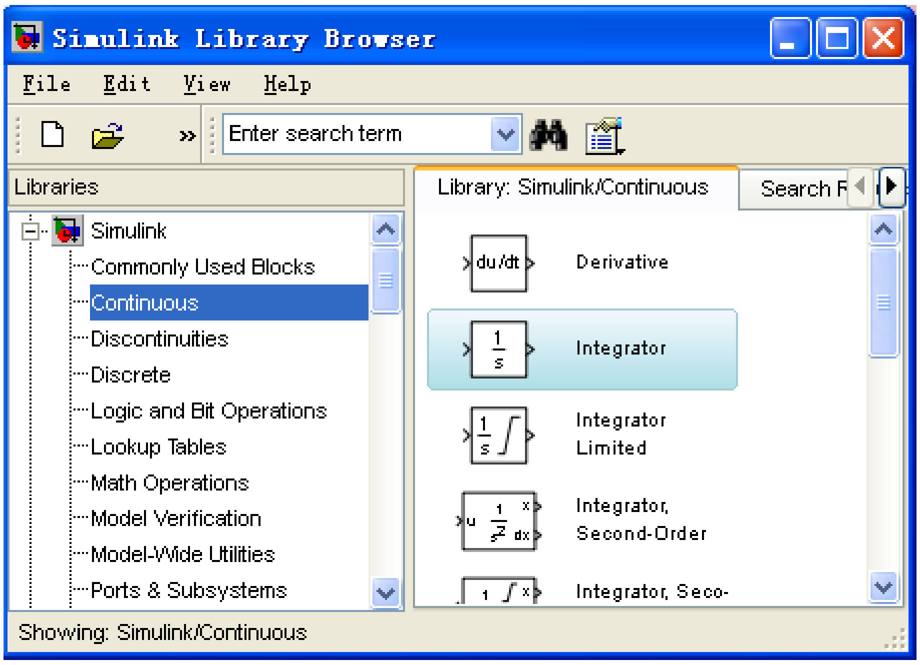


2) Open the Subsystem module, build PID module. First, three Gain modules were dragged into the page from "Simulink Library Commonly Used Blocks" and were named Kp, Ki and Kd respectively.

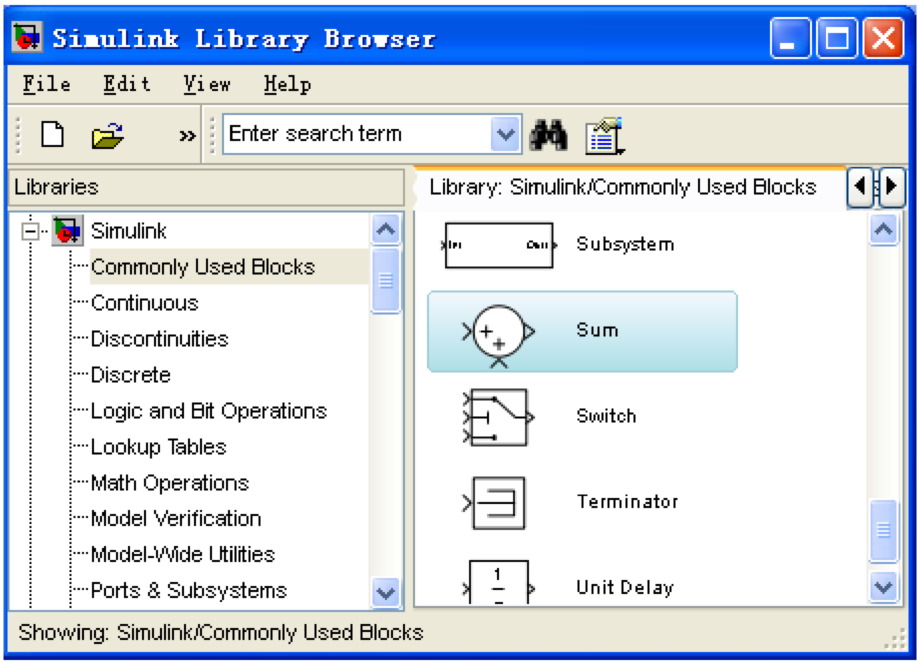


3) Drag "Derivative" and "Integrator" from "Simulink Library\Continuous" to the page.

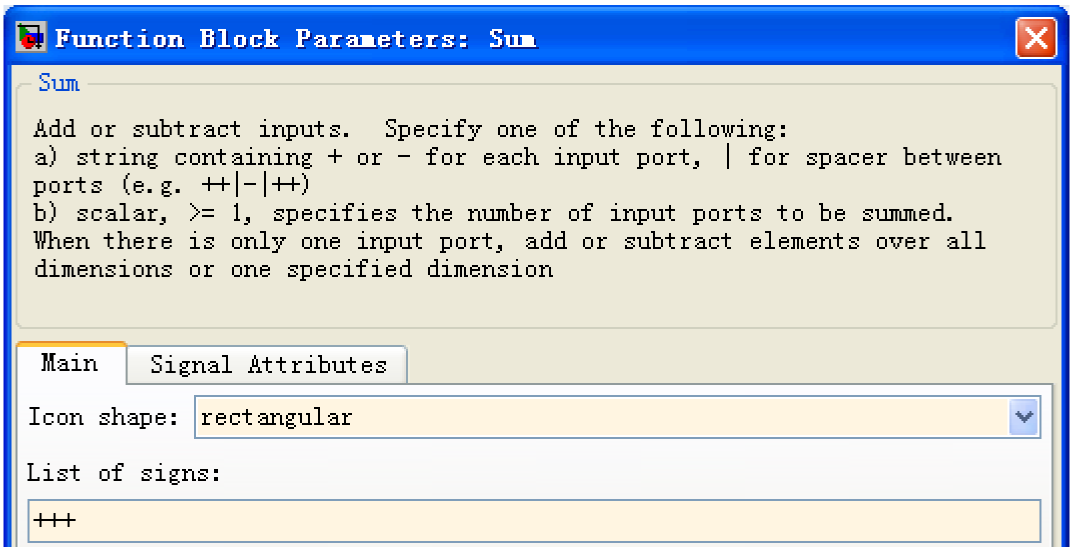




4) Drag a "Sum" from "Simulink Library Commonly Used Blocks" to the page.



5) Double-click the "Sum" module to open the following window, and the feedback Settings are shown below:



Function Block Parameters: Sum

Sum

Add or subtract inputs. Specify one of the following:
a) string containing + or - for each input port, | for spacer between ports (e.g. ++|-|++)
b) scalar, >= 1, specifies the number of input ports to be summed.
When there is only one input port, add or subtract elements over all dimensions or one specified dimension

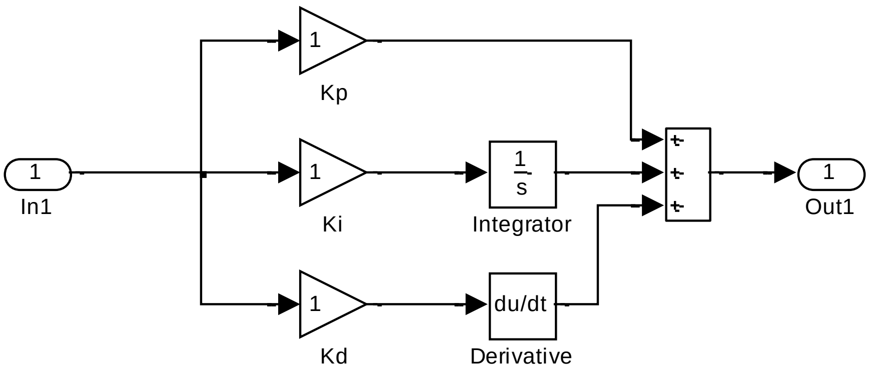
Main | Signal Attributes

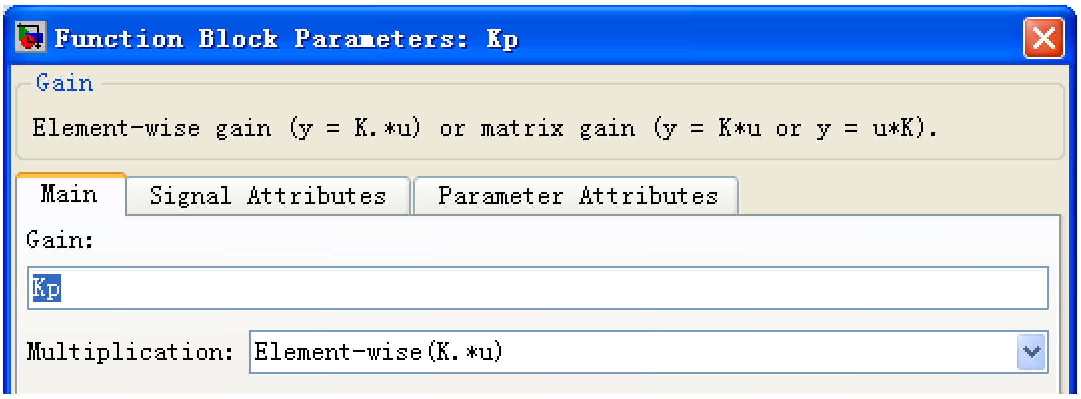
Icon shape: rectangular

List of signs:

+++

6) Connect each module according to the following figure, open three modules of Kp, Ki and Kd, and set their Gain value to Kp, Ki and Kd.





Function Block Parameters: Kp

Gain

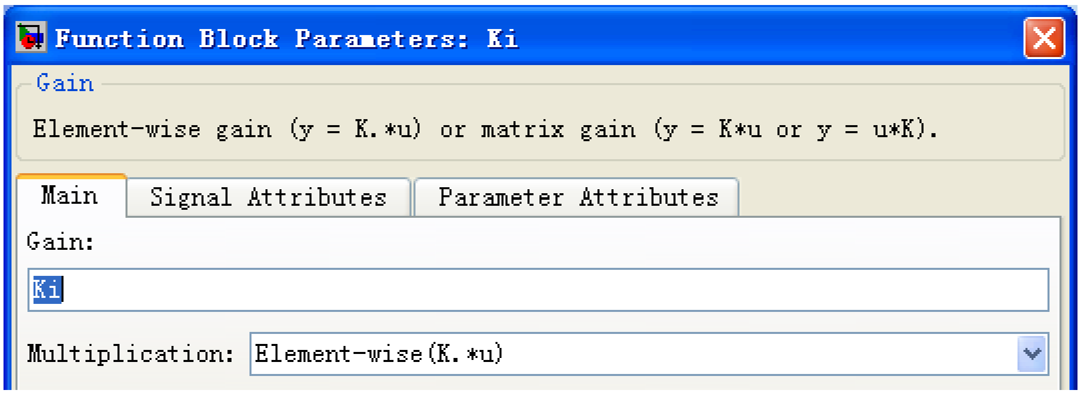
Element-wise gain (y = K.*u) or matrix gain (y = K*u or y = u*K).

Main | Signal Attributes | Parameter Attributes

Gain:

Kp

Multiplication: Element-wise(K.*u)



Function Block Parameters: Ki

Gain

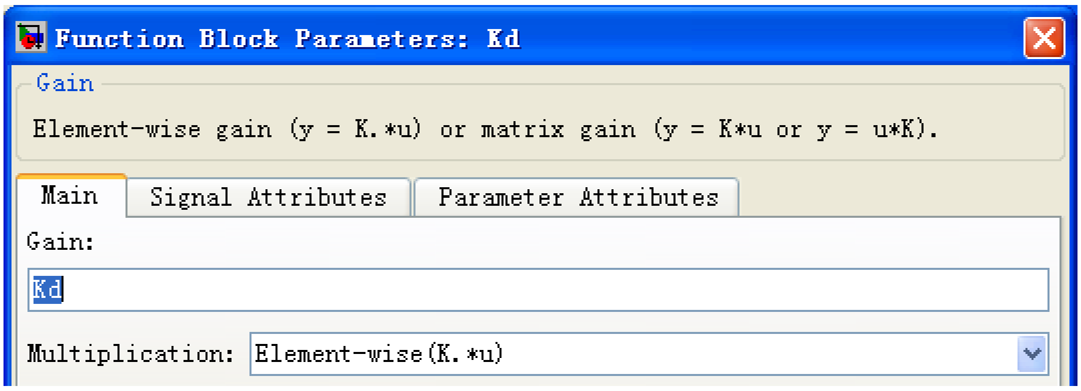
Element-wise gain (y = K.*u) or matrix gain (y = K*u or y = u*K).

Main | Signal Attributes | Parameter Attributes

Gain:

Ki

Multiplication: Element-wise(K.*u)


Function Block Parameters: Kd

Gain

Element-wise gain (y = K.*u) or matrix gain (y = K*u or y = u*K).

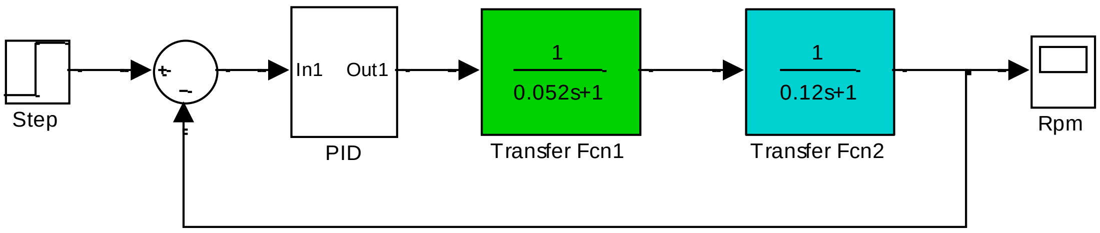
Main | Signal Attributes | Parameter Attributes

Gain:

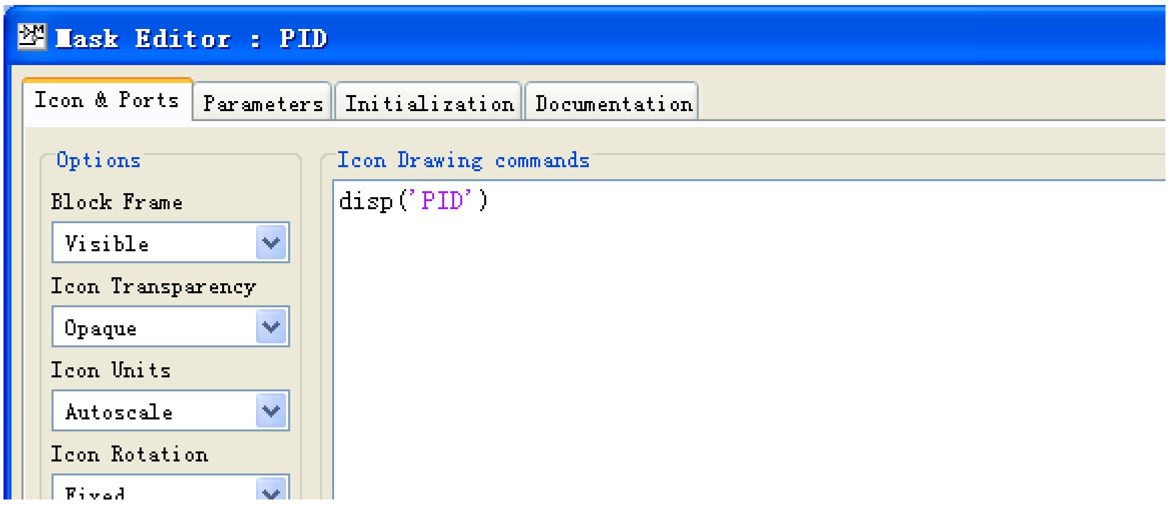
Kd

Multiplication: Element-wise(K.*u)

7) Close the page, return to the upper program, and add the submodule connection to the system under test.



8) Right-click on the PID module and select the "Edit Mask" panel. Set the encapsulated module properties as shown in the figure below.


Mask Editor : PID

Icon & Ports | Parameters | Initialization | Documentation

Options

Block Frame: Visible

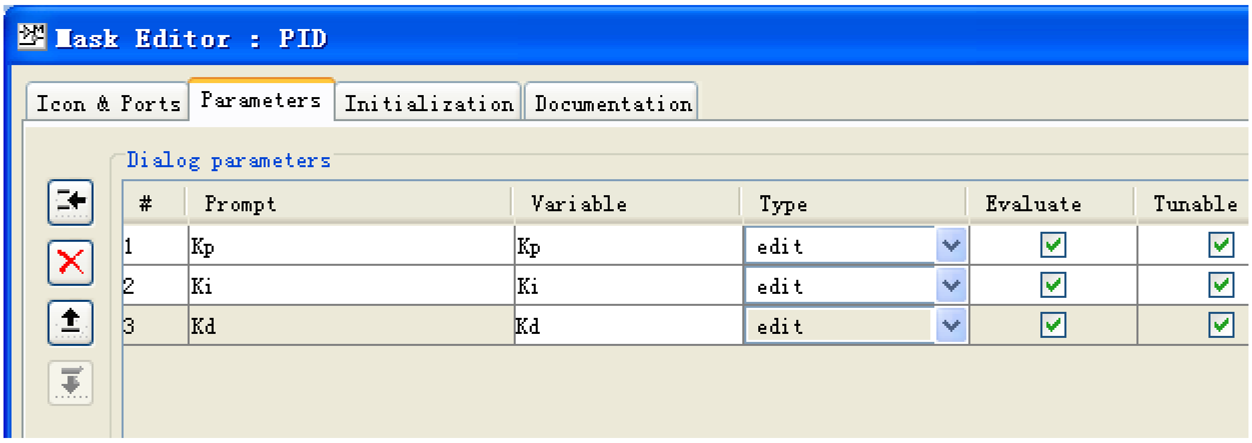
Icon Transparency: Opaque

Icon Units: Autoscale

Icon Rotation: Fixed

Icon Drawing commands

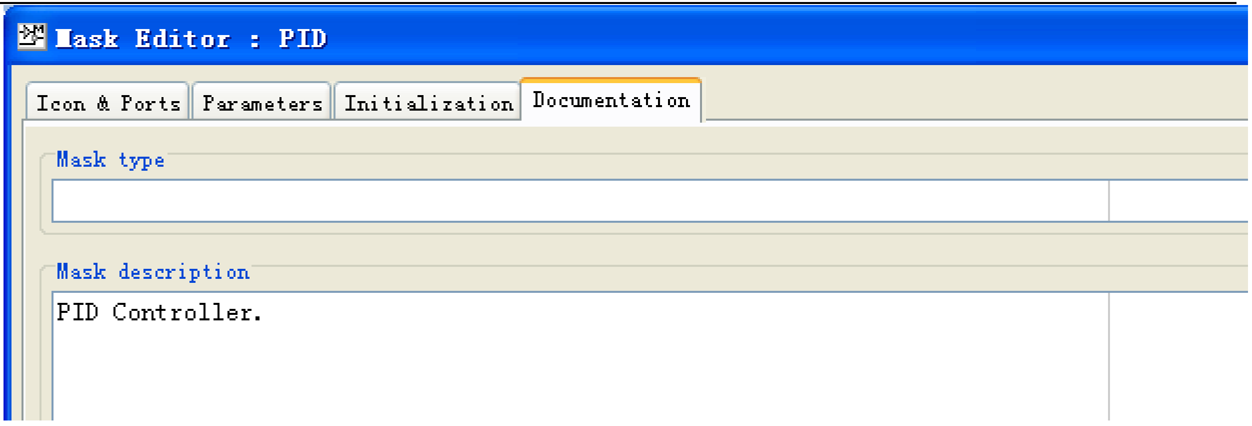
```
disp('PID')
```


Mask Editor : PID

Icon & Ports | Parameters | Initialization | Documentation

Dialog parameters

#	Prompt	Variable	Type	Evaluate	Tunable
1	Kp	Kp	edit	✓	✓
2	Ki	Ki	edit	✓	✓
3	Kd	Kd	edit	✓	✓

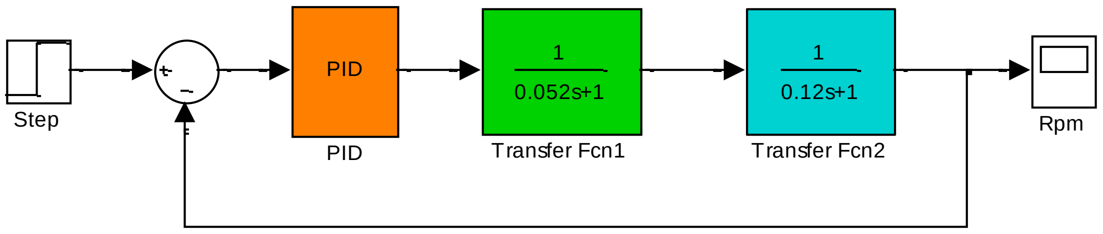

Mask Editor : PID

Mask type

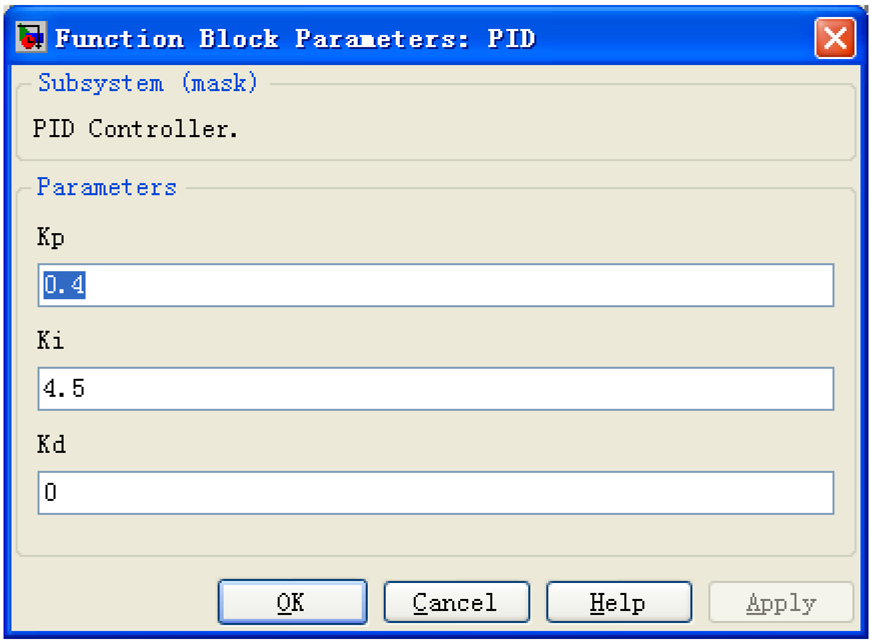
Mask description

PID Controller.

9) Right-click on the PID module and set the module "Background Color" to "Orange". The final control program is shown in the figure below.



10) Save the file as PID_Sim.mdl.Double-click to open the PID module, set Kp=0.4, Ki=4.5, Kd=0.


Function Block Parameters: PID

Subsystem (mask)

PID Controller.

Parameters

Kp

0.4

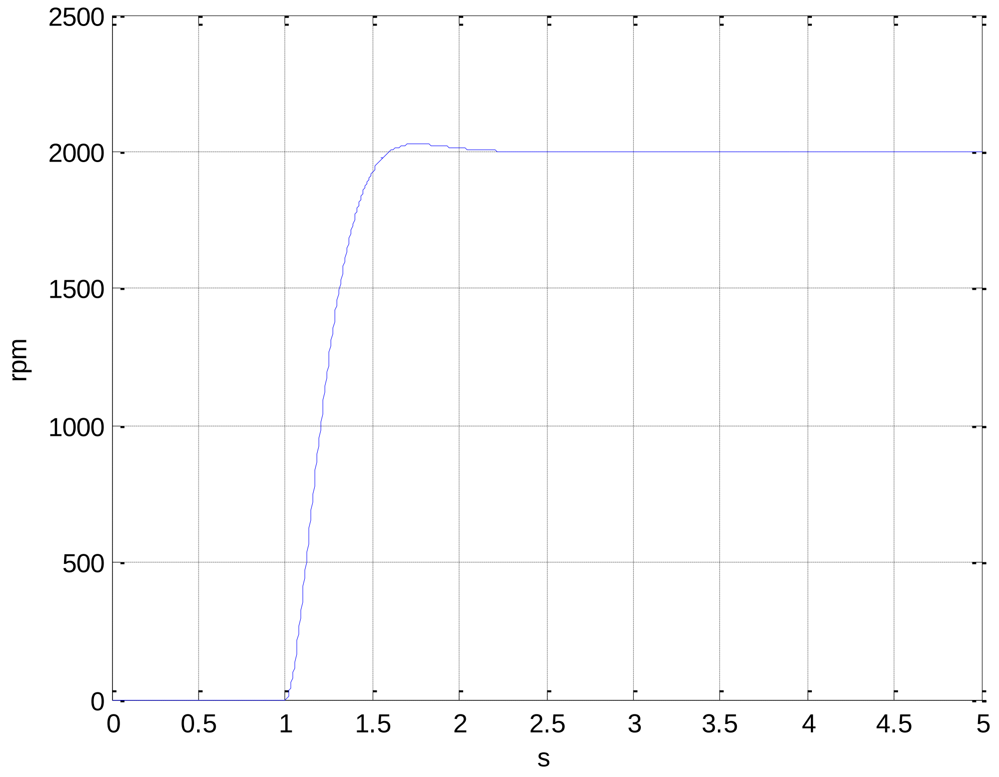
Ki

4.5

Kd

0

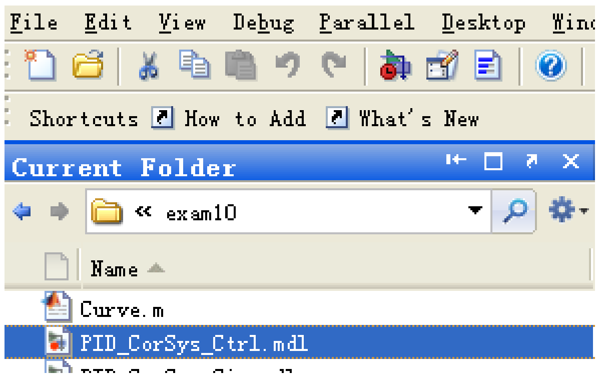
11) Click the button “▶”, and double click Scope module to get system simulation curve.

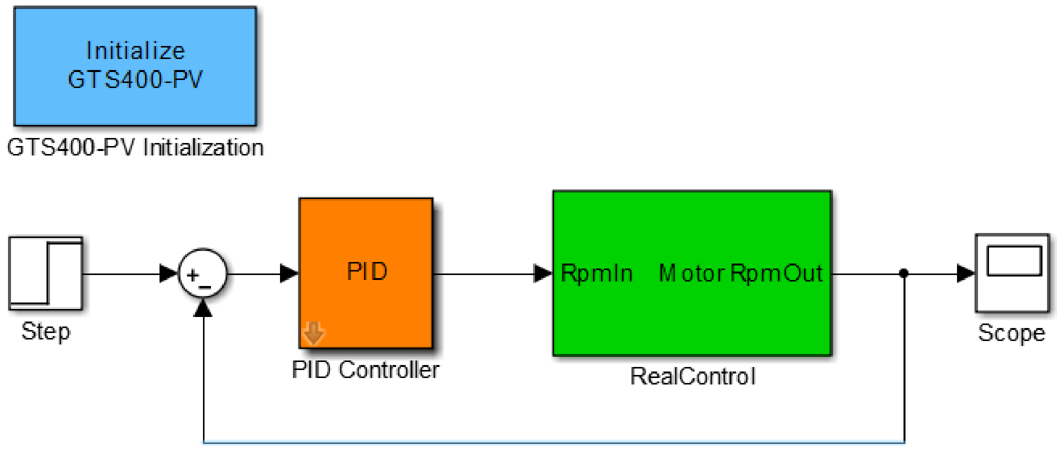


12) The overshoot rate was 1.4%, the adjustment time was 0.536 seconds, and the steady-state error was 0. Meet system design requirements.

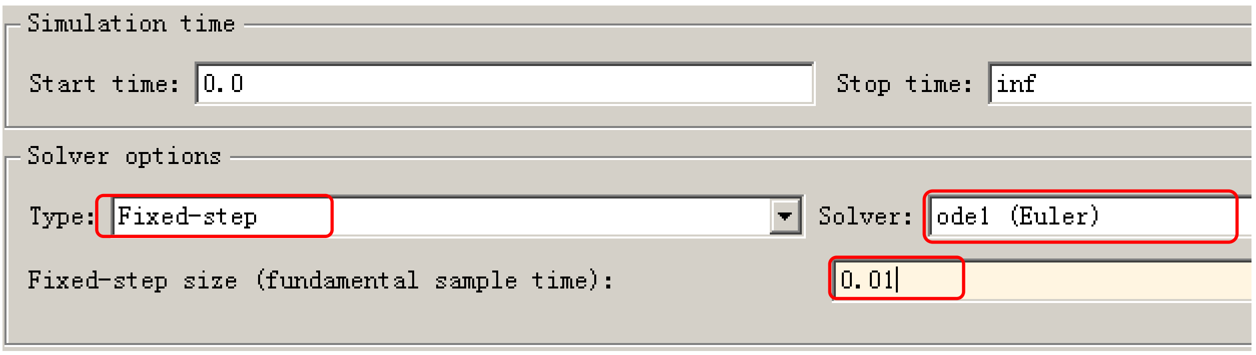
3. Real-time control of the system after adding PID correction

1) Turn on the power button on the electric cabinet of dc servo control platform.
2) Open the file "PID_CorSys_Ctrl.mdl" in MATLAB/Current Folder and the real-time control interface will pop up as shown in the figure.


File Edit View Debug Parallel Desktop Winc
Shortcuts How to Add What's New
Current Folder
« exam10
Name
Curve.m
PID_CorSys_Ctrl.mdl



3) Set PID parameters as: Kp=0.4, Ki=4.5, Kd=0.
4) As the maximum motor speed is 3000rpm, the step signal cannot be too large or too small, so it is appropriate to take 1000-2000rpm. This experiment took 2000 rpm.
5) Select "Simulation/Configuration Parameters", then the following window will pop up. Click "Solver" in the left property tree, set "Type" to fixed-step, and set size to 0.01. Also, set "Solver" to "ode1 (Euler)".


Simulation time

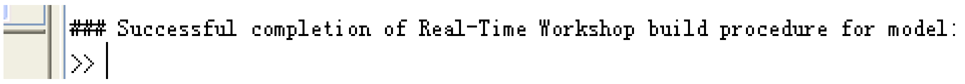
Start time: 0.0   Stop time: inf

Solver options

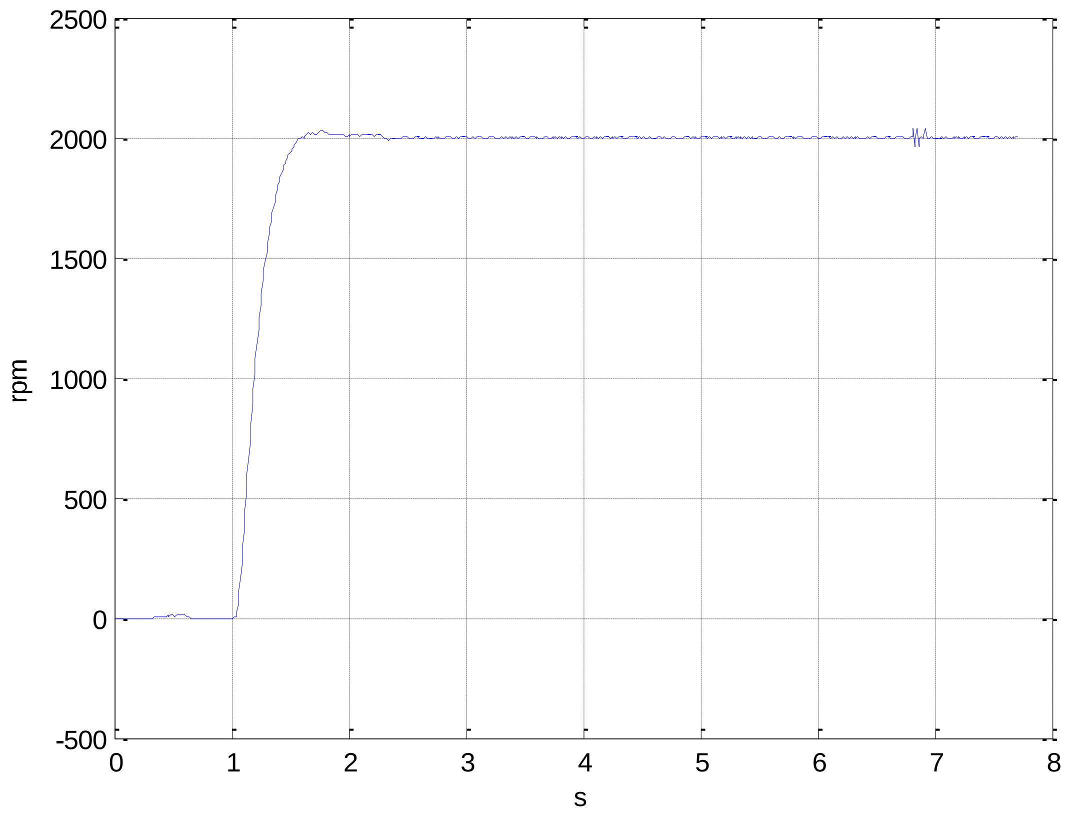
Type: Fixed-step   Solver: ode1 (Euler)

Fixed-step size (fundamental sample time): 0.01

6) Click "" to compile the program. After successful compilation, there is a prompt message in the MATLAB command window (if the control interface structure is not modified, it is not necessary to carry out this step after compiling again):


```
### Successful completion of Real-Time Workshop build procedure for model:
>>
```

7) Click "" connection procedure, at this time, user can hear a light sound when the relay in the electric control box is connected.
8) Click "" to run the procedure, and motor start. Let it run for about 10 seconds, and then click "" stop.
9) Double-click "Scope" to open the oscilloscope and observe the dc servo motor speed response curve when the step signal of 2000rpm is added. Measure and calculate the system overshoot σ , peak time t_p , adjust time t_s ,and fill in the test record form.



VI. Experiment Records

	Controller Parameters	Performance Index
Uncorrected System	None	Static value is 1000rpm, and static error is 50% $\sigma = 2.8\%, \quad t_s = 0.312\text{s}$
Correction System Simulation	Kp=0.4, Ki=4.5, Kd=0	$\sigma = 1.4\%, \ t_s = 0.536\text{s}$ The static error is 0
Correction System Measurement	Kp=0.4, Ki=4.5, Kd=0	$\sigma = 1\%, \ t_s = 0.51\text{s}$ The static error is 0

VII. Experiment Analysis

1. How to determine the PID controller's parameters?
2. Why the PID control is widely used in industrial application?

This experiment adopts PD control only, the students may try to design PI and PID calibration link by themselves. In addition, the PID link, from the standpoint of root locus, adds two null points with variable location to the system and a pole located at base point to the open-loop system. The parameters used in calibration link may be determined based on the expected performance index via the method of root locus analysis.