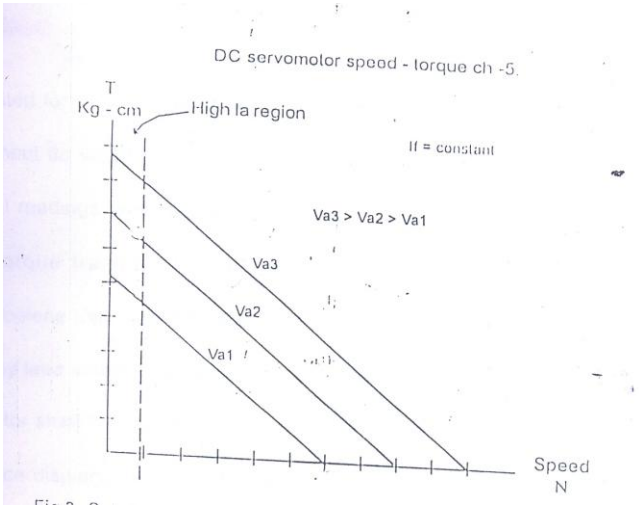
# Control Module of ME354 (Vibration & Control)

## Experiment 1(a)

**Speed torque characteristics of dc servomotor**

The set up consists a small dc servomotor rated for 50 volt (Max.) operation for armature at rated 0.5 amp field current respectively, have nominal moment of inertia 0.6 $KgCm^2$ rigidly attached load, holding torque both windings energized for rated value $4Kg/Cm$. Rated speed 2000 ~ 2500 rpm. An opt interrupter electronic tachogenerator produce a speed proportional output voltage which drives a calibrated analog RPM meter. The field and armature windings are excited with two separate dc regulated power supplies 0- 50 volt for field and 0- 50 volt for armature. Two analog meters one is calibrated for 0-1.5 amp dc and another 0-50$V_{dc}$, fitted upon panel to read input dc supplies and respective currents.

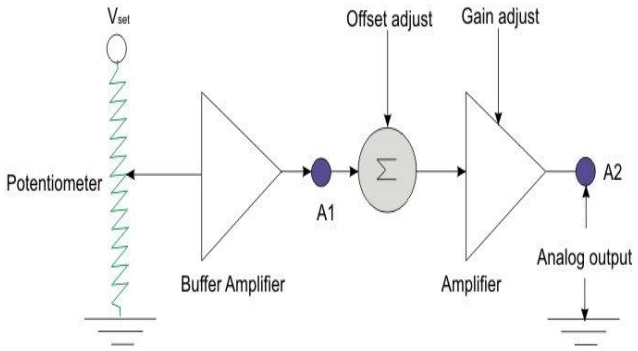
The field or armature volt-current readings are read by the meters selected for the required function. The torque transducer is a transmission belt type dynamometer. A polypropylene belt is suspended between two spring balances made adjustable by lead screw motion. The belt is made round around a pulley attached to motor shaft having radius of 2.5cm. The torque is computed from spring balance displacements by effective belt tension as Torque T= (S1-S2) r Where S1 and S2 is the reading of spring balance in Kg and r is the radius of the pulley in cm (2.5cm) or meter (.0025).

# Experiment 1(b)

**Measurement of Liner Displacement by Potentiometer**

The mechanical displacement of transducer is made by a slider system in which slider is moved along a linear scale. The scale is divided in 10 subparts & 1 cm each, which is further divided into 10 parts & 1mm each with the help of a Vernier Caliper attached to the scale displacement can be noted down. A linear motion potentiometer converts displaced parameter into an electrical signal. The mechanical travel of it is 100 mm, and its electrical resistance is 1 K Ohms. A dc excitation is used as reference $V_R$. The transfer function of the transducer may be written as $K_e = \frac{\Delta V}{\Delta d}$ in volt / mm. The linearity (printed) is 2% in full scale. A 3.0 volt dc, band gap voltage refrence is connected across the transducer fixed ends as excitation source $V_{ref}$.

# Experiment 2

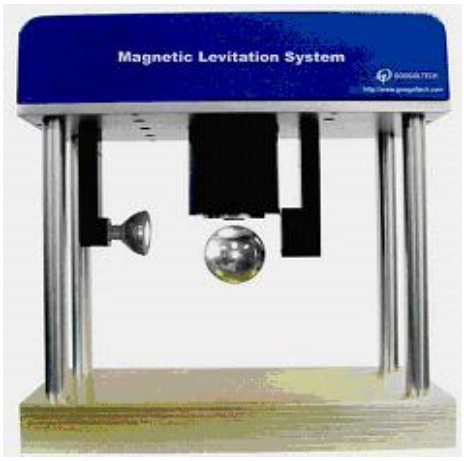
**Speed Measurement of DC Motor by Various Sensors**

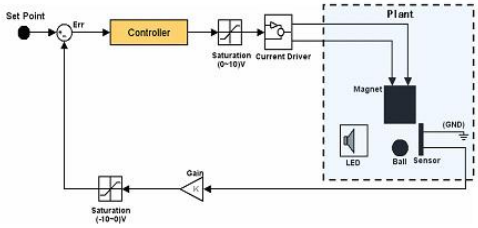
Speed Measurement of DC Motor by Magnetic pickup, Inductance and Hall effect switch uses magnetism whereas Photo reflection and Photo interruption uses optical properties. Stroboscope is based on optical and persistence of vision property of human eye. All the sensors are placed in the vicinity to a slotted wheel fitted to the motor shaft. This slotted wheel causes interruption in magnetism of magnetic pickup, Inductance and hall effect switch. It causes interruption in optical path of photo reflective and photo interruptive sensors. Thus causes some periodic generation of voltage. The periodic signals are amplified, converted into pulses and these pulses are further converted into DC voltage by frequency to voltage converter. Converter output having passed through zero span amplifier can be measured as speed indication.

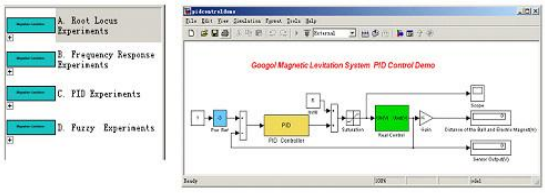
# Experiment 3

**Magnetic Levitation System**

Magnetic Levitation System is a platform for the study of magnetic levitation technology. It's mainly composed of solenoid, position sensor, amplifier/compensation device (driver), digital controller (PCI1711) and control object steel ball. It is a typical magnetized levitation system. The system can be divided into two parts, MLS main body, control platform with data acquisition card and PC.

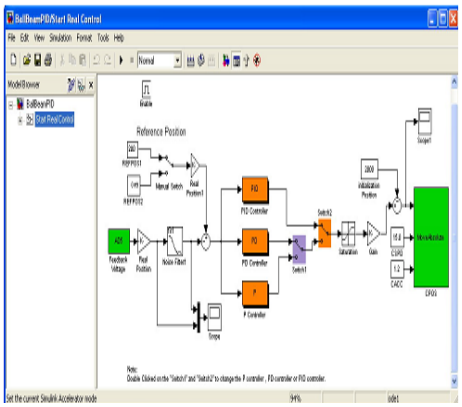


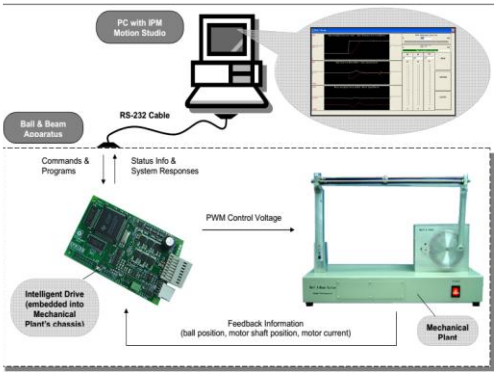




**Ball & Beam apparatus**

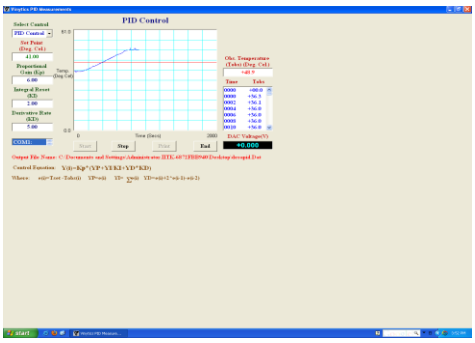
Ball & Beam apparatus includes mechanical plant with built-in DC servo motor and DC voltage supply, IPM100 intelligent servo drive system, PC with the control program. The mechanical plant consists of a base, a beam, a ball, a lever arm, a gear, a support block, a motor and an embedded electrical power supply. The ball can roll freely along the whole length of the beam. The beam is connected to the fixed support block at one side and to the movable lever arm at another one. In turn the motion of the lever arm is controlled by the DC brush motor through gear. The motor is equipped with built-in rotary optical incremental encoder that provides feedback information about current actual rotary position of the motor shaft. In the slot along the beam there is a linear potentiometer sensor that senses current linear actual position of the ball on the beam. Both measured positions are fed back to the control system to organize a closed loop control. As the servo gear turns by an angle theta, the lever arm changes the angle of the beam by alpha. When the beam is moved away from horizontal position, gravity causes the ball to roll along the beam. The purpose is to design and implement a control algorithm that moves the shaft of the motor in such a way that the desired position of the ball is stabilized on the beam.



**Thermal Heating Using Vinaytics PID Controller**

The tank contains PT-100, a thermocouple whose resistance varies with the temperature and there by changes the o/p voltage this analog variation is sensed by the thermocouple sensor and gives a corresponding, amplified o/p to the ADC. The tank also contains a stirrer to keep the temperature of water uniform. Analog to Digital Converter converts to the analog signal from thermocouple and display the temperature digitally on the display. Microcontroller circuit fetches the digital signal from the ADC, when the capture code "#01M" is provided by the software (XTALK). The i/p format for acquiring the data from ADC to the microcontroller is provided by the software from computer through serial cable "RS232".DAC accepts the i/p data format as"#02XBBB". This "BBB" is the 12 bit data for controlling the DAC o/p voltage provided by the computer through microcontroller. The DAC o/p voltage (DC) controls the o/p voltage (AC) of the TRIAC control circuit. The AC o/p voltage of the TRIAC controls the heating elements. This heating is sensed by the PT -100 and gives the analog signal to the thermocouple sensor and thus forms a close loop circuit.



# MEASUREMENT OF LINEAR DISPLACEMENT BY POTENTIOMETER

The measurement of displacement, position or say location is an important parameter in industries. The simplest and fairly accurate tranducer used to measure displacement is potentiometer. It converts linear (or angular) displacement into electrical signals, by changing its resistance. A potentiometeric transducer is an electromechanical device with a resistive element fixed at two ends and a wiper as a movable slider. Motion of the wiper results in a change in resistance with refrence to its ends. The change in electrical parameter (resistance) versus its mechanical parameter (displacement) is fairly linear in linear potentiometer. The device is passive in nature thus it is excited with dc voltage normally to obtain change in voltage or current as displacement signal. The set up is designed to study linear potentiometer as a linear displacement transducer. The set up has following systems:

**a.** ***Slider system for displacement***. The mechanical displacement of transducer is made by a slider system in which slider is moved along a linear scale. The scale is divided in 10 subparts & 1 cm each, which is further divided into 10 parts & 1mm each with the help of a Vernier Caliper attached to the scale displacement can be noted down.

***b.Transducer*****:** A linear motion potentiometer converts displaced parameter into an electrical signal. The mechanical travel of it is 100 mm, and its electrical resistance is 1 K Ohms. A dc excitation is used as reference $V_R$: The transfer function of the transducer may be written as $Ke = \Delta Vo / \Delta d$ in volt / mm. The linearity (printed) is 2% in full scale.

***c. Tim voltage refrence*****:** A 3.0 volt dc, band - gap voltage refrence is connected across the transducer fixed ends as excitation source Vref.

***d. The buffer amplifier* A1:** It is a high impedance op - amp configurated as non - inverting buffer. It prevents loading of measuring stage upon transducer resistance. Its transfer function is K = 1.

**e. *The adder*:** This block has two inputs and one output. An op - amp serve as non - inverting summing amplifier, which adds two electrical signals one from the buffer and another from offset voltage source as Vos. The purpose of adding offset is for calibratic of initial reading. The output of this block may be written as, Ka = Vos + K.

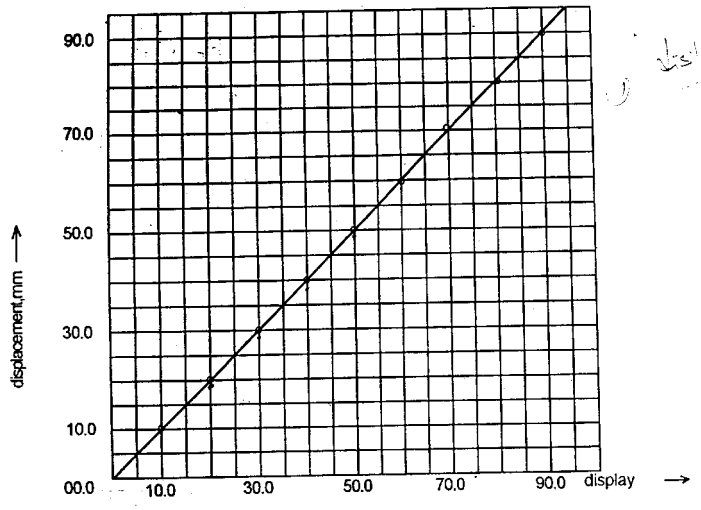
***f. Analog amplifier* A2:** It is another op - amp non - inverting amplifier serve to amplify the analog voltage to sufficient level. As in case of given set up the A2 serve to amplifiy the input votage twice to obtain 0 - 5V analog output with the displacement.

**g. Digtal pane/ *meter:*** A 200 full scale deflection 3 1/2 digit digital meter is provided to take readings in volts (19.99V fs) or in of displacement in mm.

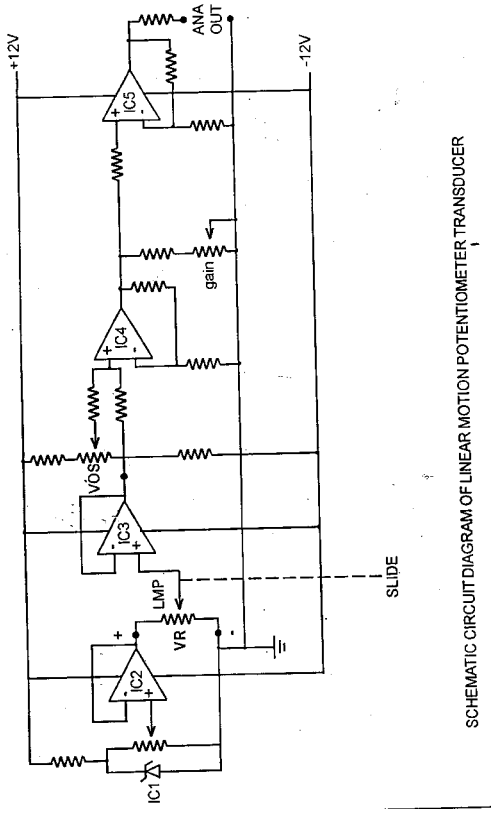
***h. Power supplies:*** Twp, built in IC regulated symmetrical supplies are pro-vided for signal condition circuit and digital display. The reference voltages are generated from *12* volt op - amp supply using band gap reference LM 35. The ground is made common.

Prepaation of Table

| Sr no. | d mm (slide) | V (A1) | d mm (DVM) |
|---|---|---|---|
| 1 | ... | ... | ... |
| 2 | ... | ... | ... |
| n | ... | ... | ... |




Typical polt of displacement v/s displayed readings. form plot $S_r$ = .03 V / mm.
The $P_r$ = 10Ω/mm




SCHEMATIC CIRCUIT DIAGRAM OF LINEAR MOTION POTENTIOMETER TRANSDUCER

# Experimental Procedure

**Title** –Measurement of displacement by linear potentiometer.

**Objective**- To measure linear displacement by potentiometer and find out the hysteresis, linearity and systemic error of the Uncalibrated and Calibrated system.

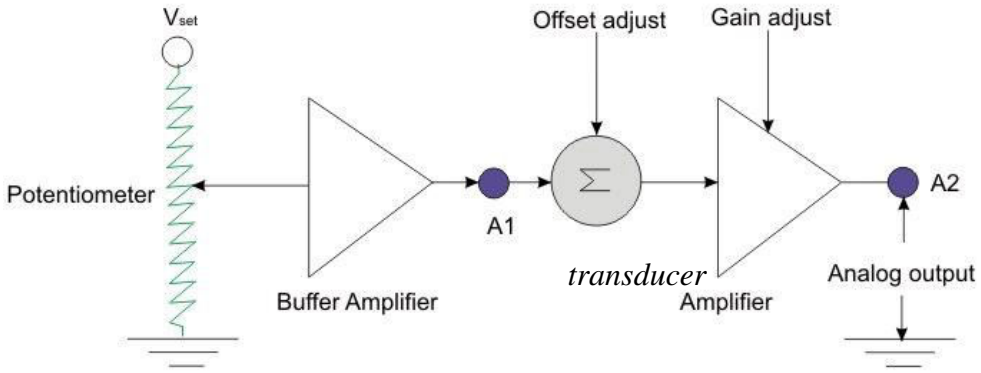



Fig: 1. Block Diagram of Potentiometer Displacement

**Calibration of the system at $A_2$**:

1. Adjust the slider to 10mm (1cm) line. Adjust offset control to read 10.0mm at the Digital voltage meter (DVM).
2. Now bring the slider to 90mm line. Adjust gain control to read 90.0mm in digital voltage meter (DVM).

Repeat step 1 and 2, two to three times, such that both readings coincide.

**Uncalibrated Reading from $A_1$:**

Start from zero of the scale; take reading on the DVM for displacement and the corresponding buffer output ($A_1$) for each increment of 10 mm.

Repeat the experiments in the decreasing order of the displacement.

**Calibrated Reading from $A_2$:**

Again, start from zero of the scale take readings of displacement, the DVM reading corresponding to ($A_2$) amplifier adjust for each increment of 10 mm.

Repeat the experiments in the decreasing order of the displacement.

**Observation and calculations**:

1. Plot a graph between the displacement on DVM and output volts for Calibrated and Uncalibrated reading and D Vs d plot, as shown at last of this manual.
2. From the plot find out the slope of the curve as Sensitivity of the transducer given by:

$$S_R = \Delta V / \Delta d, \text{ mV/mm.}$$

$\Delta$ **V =** Voltage difference between two assigned point

$\Delta$ **d =** distance between two assigned point, in the V-d graph

3. Find the maximum hysteresis error $\mathbf{E_H}$ as the difference of the two readings taken in upscale and downscale reading for calibrated and Uncalibrated curves

$$\mathbf{E_H = (Vu - V_d)}$$

Where **Vu** is voltage reading in increasing direction of scale. And $\mathbf{V_d}$ is voltage reading in decreasing direction of scale.

4. Find out the maximum deviation in two sets of readings (upscale and downscale).

5. Check the non-linearity of the system from the V-d graph.

6. Find out uncalibrated and calibrated pot sensitivity **Sr** as the difference between the two resistances divided by the displacement

$$\mathbf{Sr_{uc\_pot} = (R_{2dA1} - R_{1dA1})/(D_2 - D_1)}$$

$$\mathbf{Sr_{c\_pot} = (R_{2dA2} - R_{1dA2})/(D_2 - D_1)}$$

Where $\mathbf{R_{ndA1} = d_{A1} * (R/100)}$

and $\mathbf{R_{ndA2} = d_{A2} * (R/100)}$

here R=1000 Ω and n= 1,2,3,4…10.

Now to calculate the system error $\mathbf{E_R}$,

$$\mathbf{E_{Ruc} = (d_{2A1} - d_{1A1})/(D_2 - D_1)}$$

$$\mathbf{E_{Rc} = (d_{2A2} - d_{1A2})/(D_2 - D_1)}$$

**Observation from the graph**:

7. Comment on the non-linearity in the system.

8. Comment on the calibrated and uncalibrated response of the system.

9. Comment on the calibrated and uncalibrated error of the system.

10.Comment on the change on the gain at $A_2$ and its effect on the readings.

**Note: Don't allow force to slider after maximum and minimum travel.**

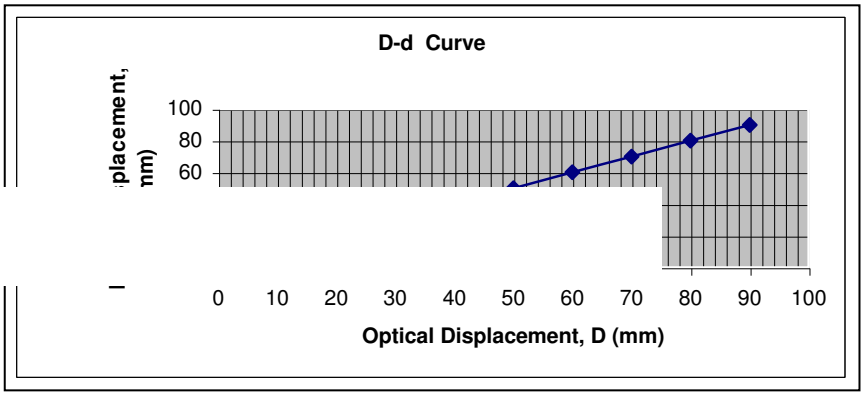
**Observation table:**

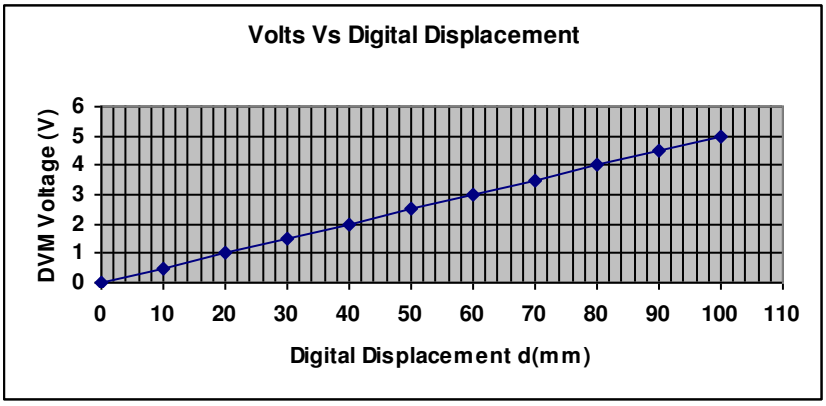
| Sno. | D | $V_{A1}$ | | $V_{A2}$ | | $d_{A1}$ | | $d_{A2}$ | |
|---|---|---|---|---|---|---|---|---|---|
| | mm | UP | DN | UP | DN | UP | DN | UP | DN |
| **1** | 10 | | | | | | | | |
| **2** | 20 | | | | | | | | |
| **3** | 30 | | | | | | | | |
| **4** | 40 | | | | | | | | |
| **5** | 50 | | | | | | | | |
| **6** | 60 | | | | | | | | |
| **7** | 70 | | | | | | | | |
| **8** | 80 | | | | | | | | |
| **9** | 90 | | | | | | | | |

## Calculations from observation table:

| $R_{dA1}$ Resistance Uncal. Up direction | $R_{dA2}$ resistance Cal. Up direction | $S_{RUC}$ | $S_{RC}$ | $E_{UC}$ | $E_C$ | $E_H$ |
|---|---|---|---|---|---|---|
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |

**Plots:**



# SPEED TORQUE CHARACTERISTICS OF DC SERVOMOTOR

**Introduction:**
DC servomotor has been the prime mover element most widely used in control applications. In position control the DC servomotor is of the highest choice due to simple control circuitry. These motors suffer from wear and tear because of brush and commutator. Generally the DC motor is controlled by armature control via a dc servoamplifier. In certain operations field control is used but variable flux motors suffer from costlier current source and poor control of field flux due to magnetization saturation for low speed application. In general way armature control is preferred.

**DC Servomotor:**
DC Servomotor is basically a separately excited dc motor with only difference is made that its armature resistance is normally kept high with low mass to obtain fast response and steeper torque – speed slope. The field is energized by a constant voltage source and armature is controlled by an amplifier. Now days rare earth permanent magnet field dc motors are available which gives excellent steeper slopes because of it does not suffer from the field saturation and having low mass high resistance armature facilitate to operate it from wide range input voltages.

**Basic operation principle of DC servomotor:**
In either way of controlling the DC servomotor is basically a torque transducer which convert electric energy into proportional mechanical energy. The torque developed on the motor shaft is directly in proportion of field flux and the armature current (In either case the motor torque increase with increase in field flux or increase in armature current).
The relationship between motor torque Tm, field flux and armature current is given as.

$$T_m = K_m \phi I_a$$

Where $K_m$ is the motor torque constant, $\phi$ is the magnetic field strength in air gap and $I_a$ is the armature current. In addition with it torque developed, a voltage eb, developed across the armature winding carrying current which is proportional to the angular shaft velocity $\omega_m$. The relationship between these is given as

eb = $K_m \phi \omega_m$

In fig1, armature controlled DC servomotor is shown. The field current is made constant and armature is connected with the variable supply $V_a$. The speed of the motor is controlled by controlling armature input voltage $V_a$. The speed is proportional to the applied voltage $V_a$ since $I_f$ is constant. The armature has an

inductance $L_a$ and an equivalent dc resistance $R_a$. The field current is constant, thus flux $\phi$ is constant and the armature (Current) voltage control the motor shaft (rotational) output such.

$$N = [(V_a - I_a R_a) / \phi]$$

The moment of inertia being $J_m$ and coefficient of viscous fraction is $f_m$ respectively. The angular shift in the motor shaft being $0_m$ and the corresponding angular velocity being $\omega_m$.

As the $\phi$ is kept constant the relationship between developed torque $T_m$ and the armature current $I_a$ is given as

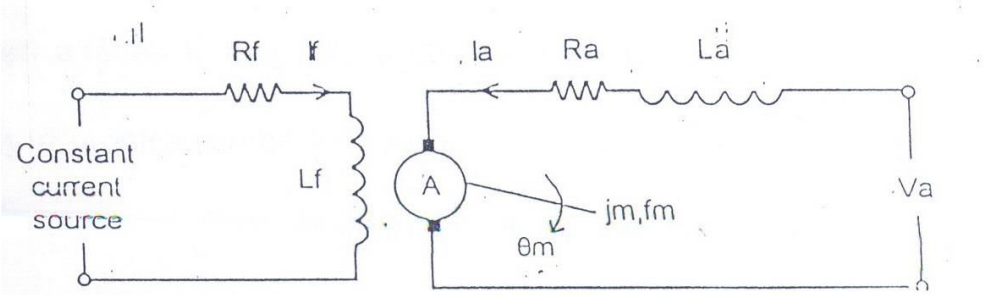



Fig 1 : Armature controlled DC servomotor.

---

$$T_m = K_T I_a$$

Where $K_T$ is the motor torque constant expressed in $N_m / A$

The applied input voltage $V_a$ is being opposed by the back $emf$ produced by angular velocity. The relationship between $eb$ and the motor speed $\omega_m$ is given as

$$eb = (K_b) d\theta m / dt$$

**Torque speed curves of DC servomotor:**

The torque speed curves of a DC servomotor describe the static torque producing capability of the motor with respect to the applied voltage and the motor speed. With refernce to curves shown in fig. 2, in steady state the effect of inductance $L_a$ is negligible and the torque equation of the motor is given

$$T_m = K_T I_a = K_T (V_a - K_b \omega m) / R_a$$

For a given applied voltage $V_a$ a straight line is approximated for speed torque but in practice some limitations occurs particularly at high armature currents such as saturation of iron (Core) and heat dissipation in commutator. In servo applications the motor is stand still in these cases rises to small angular error which is made to remove by increasing the controller amplifier gain.

* For further references see, Automation control systems by B.C. Kuo and Linear control system by B.S. Manke.

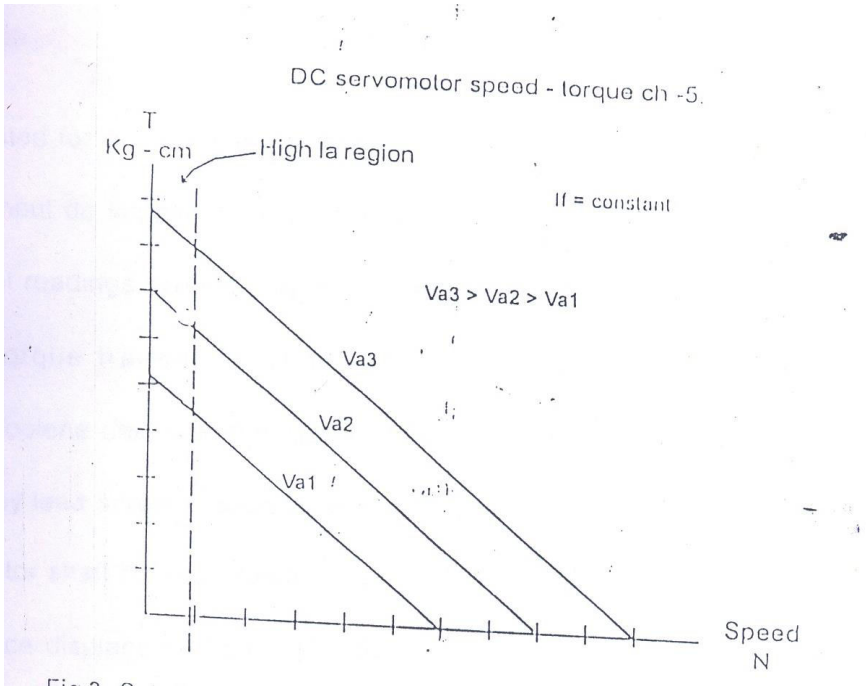



**Fig 2**: Set of speed – torque curves (theoretical) for different values of $V_a$ .

**About the set up:**

The set up consists a small dc servomotor rated for 5o volt (Max.) operation for armature at rated 0.5 amp field current respectively, have nominal moment of inertia 0.6 $KgCm^2$ rigidly attached load, holding torque both windings energized for rated value $4Kg/Cm$. Rated speed 2000 ~ 2500 rpm. An opt interrupter electronic tachogenerator produce a speed proportional output voltage which drives a calibrated analog RPM meter. The field and armature windings are excited with two separate dc regulated power supplies 0- 50 volt for field and 0-

50 volt for armature. Two analog meters one is calibrated for 0-1.5 amp dc and another 0-50$V_{dc}$, fitted upon panel to read input dc supplies and respective currents. The field or armature volt-current readings are read by the meters selected for the required function. The torque transducer is a transmission belt type dynamometer. A polypropylene belt is suspended between two spring balances made adjustable by lead screw motion. The belt is made round around a pulley attached to motor shaft having radius of 2.5cm. The torque is computed from spring balance displacements by effective belt tension as Torque T= (S1-S2) r

Where S1 and S2 is the reading of spring balance in Kg and r is the radius of the pulley in cm (2.5cm) or meter (.0025).

**Experiment procedure:**

**Objective:**
To draw armature controlled speed- torque characteristics curve for given DC servomotor.

**First time operation:**
The set up has two units, one is the power supply for said motor and other is the motor unit with tachogenerator and torque transducer. Both units should be attached by mean of given 8 pin (PM8) plug with motor unit.
Observe the front panel of the set – up. There are three analog meters and three miniature toggle switches. One switch is designated for power on /off and two switches for take readings of volt and current either for field or for armature depends upon switch positions. Two continuously variable controls are provided to control the field and armature supplies. The third analog meter reads number of revolutions RPM. In the motor unit two spring balances are suspended with a polypropylene belt rounded upon the pulley attached with motor shaft. Both spring balance tension is adjustable by mean of lead –screw motion.

1. After familiarization of controls, adjust lead –screw know to adjust spring balance to read 0's in both. Keep both supply control to minimum (fully counter clockwise position). Switch on the power. A led will glow to commence the power is applied to the set - up.

2. Select current read switch to 'field' side. 'field supply' control to read 0.5 amps at current meter.

3. Select current read switch to 'armature' side. 'Armature supply' control to 50 volt dc.

4. Note the no load speed N, from the RPM meter and no – load $I_a$ from the current meter. Observe at the spring balances that these are at 0 readings.

5. Increase the load by mean of lead – screw adjuster mounted upon spring balance to read the other balance some reading say 100gm (0.1Kg).

6. Note the spring balance readings as S1 (LHS) and S2 (RHS), armature current $I_a$, speed N, rpm at steady state condition. Increase tension of spring balances* S2 slowly in 100 gm steps and note successive readings as before. Tabulate the observations.

The table for observations..... ($I_a$ =….volts, If =….amp.)

| Sr. no. | S1 Kg | S2 Kg | $I_a$ Amp | T Kg cm |
|---|---|---|---|---|
| 1. | 0 | 0 | …. | 0 |
| 2. | …. | …. | …. | . … |
| 3. | …. | …. | …. | . … |
| 4. | …. | …. | …. | . … |
| 5. | …. | …. | …. | . … |
| n. | …. | …. | …. | . … |

In unstable (vibrating) condition take mean readings between two points.

7. At some stage the motor goes very slow or stops, decrease the field current to minimum and note the $I_a$ . If current meter shoots (out of scale) than bring armature supply gradually back to bring $I_a$ in readable range. Note the values of voltage and current for armature.

8. Calculate the armature resistance $R_a$ ($\Omega$) = $V_a$ / $I_a$. Calculate the torque T= S1 - S2 (r), where S1 and S2 is corresponding readings of spring balances; r is the radius of pulley (2.5).

9. Draw the speed –torque curves from the table. At high armature currents the error is predictable, increase the curve by straight line approximation to meet y-axis.

10. From the point T stall (torque at zero speed), find its value from the graph.

11. Convert the torque T into Newton meter Nm. Convert the speed N into angular velocity.

$\omega_m = 2\pi N / 60$ rad/sec.

Draw another graph between torque T in $N_m$, and angular velocity $\omega_m$. Plot another graph between $\omega_m$ and $I_a$ (see fig.4 for example). Fix an operating point 'p' in the linear region of the curve to avoid magnetization saturation.

From the curve in fig. 4

Now the $T_m = K\phi I_a$, since $\phi$ is constant

$T_m = K_T I_a$, Where $K_T$ is motor torque constant and $I_a$ current at intersect m

so $K_T = T_m / I_a$ Nm/A

Now motor back being proportional to speed so

$eb = K_b (d_o / d_t)$, where $K_b$ is back $emf$ constant

$eb = K_b \omega_m$

Taking $I_a$ at corresponding angular speed at point 'p' is m amp and $R_a$ is the armature resistance calculated before.

$eb = I_a R_a$ at p

So $K_b = eb / \omega_m$ V/ rad/ sec.

If the armature inductance is neglected, prepare the block diagram of modelling the dc motor.

From the model the transfer function is

$$\frac{0(s)}{V(s)} = \frac{K_T}{s(s R_a Jm + K_T K_b)}$$

$Jm$ = given 6 x $10^4$ $Kgm^2$

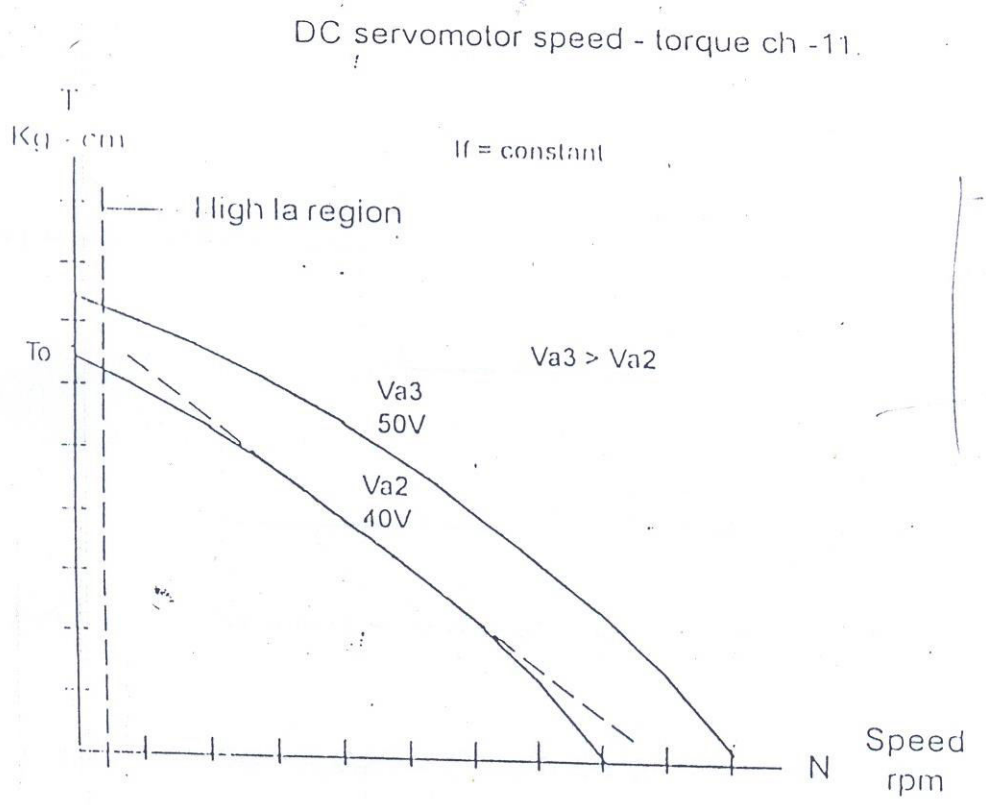



Fig 3. Typical Set of speed - torque curves for different values of Va.

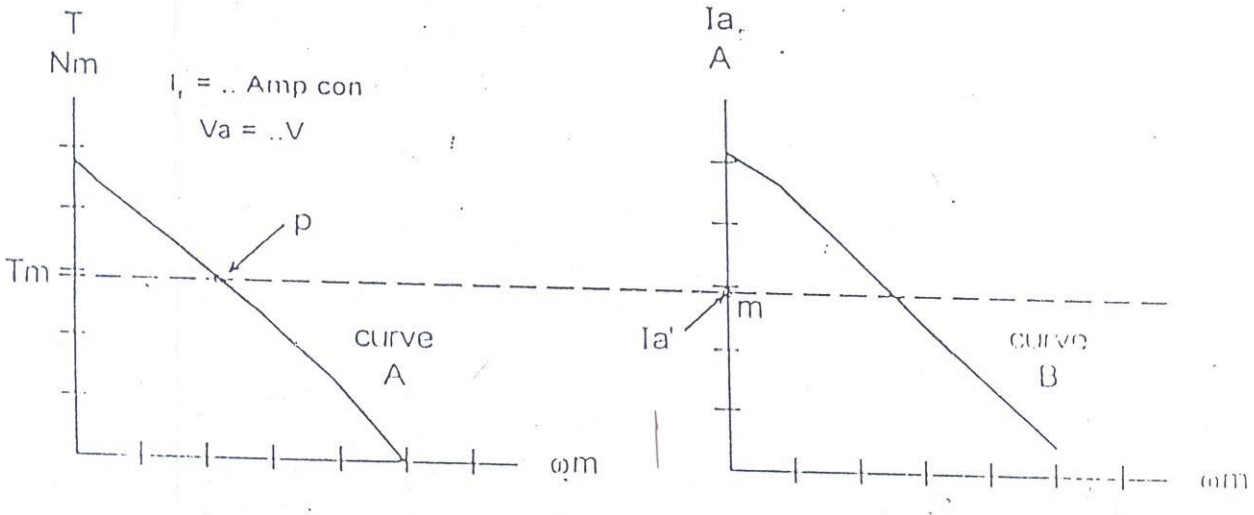



Fig 4. Typical Set of speed - torque curves to evaluate transfer function.

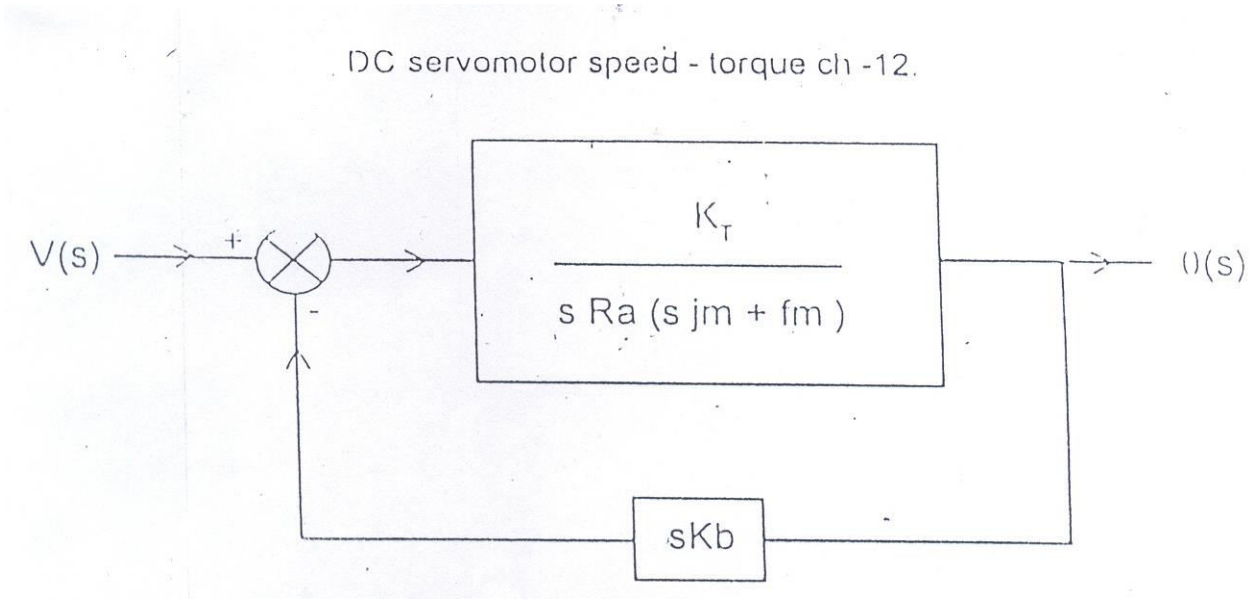



**Fig.5** Block diagram representation for (armature controlled) dc servomotor.

12. The few more set of curve should be drawn taking step 1 to 6 for different field current (0.3 ~ 0.9 amp.) and armature voltages (30~ 50 volt.).

# Experimental Procedure

**Title**- Speed torque characteristics of dc servomotor

**Objective**: To draw armature controlled speed-torque characteristics curve for given DC servomotor.

**Procedure**: 1) Connect motor unit to power supply unit by means of given 8 pin PM8 plug.
2) Adjust both the spring balance to zero. Keep both supply control to minimum. Switch on the power.
3) Select current read switch to field side. Adjust field supply to0.5 amp at current meter.
4) Select current read switch to armature side. Adjust the supply to 50 volts.
5) Note the no load speed N and no load current Ia (armature current). Observe at the spring balance that these are at zero reading.
6) Increase the load by means of lead screw adjuster and adjust one of the spring (say S1) to100 gm.
7) Note the spring balance readings as S1 (LHS spring) and S2 (RHS spring). Also note down armature current (Ia), speed (N) at steady state condition. Increase tension of spring balance S1 slowly in 100 gm steps and note successive readings as before.
8) At some stage the motor goes very slow or stops. Decrease the field current to minimum and note the Ia.
9) Calculate the torque and draw the speed-torque curve.
10) Find the point Tstall (torque at zero speed) from graph.
11) Convert the torque into Nm and speed into rad/sec. Then draw the graphs T in Nm vs angular velocity in rad/sec and Ia in amp vs angular speed in rad/sec.
12) Fix an operating point 'p' in the linear region of the curve. Now note the Ia', $T_m$ and $\omega_m$ corresponding to the point 'p'.

**Observation:**

Armature voltage Va =
Field current If =

| Sr. no. | S1 (kg) | S2 (kg) | Ia (amp) | N (rpm) | T (kg cm) |
|---|---|---|---|---|---|
| 1 | 0 | 0 | | | 0 |
| 2 | | | | | |
| 3 | | | | | |
| 4 | | | | | |
| . | | | | | |

**Calculation**:

Armature resistance Ra = Va/Ia
Torque T = (S1 –S2) * r (radius of pulley r = 2.5cm)
Motor torque constant $K_T = T_m / Ia'$
Back emf constant $K_b = eb / \omega_m$ where eb = Ia' Ra

Transfer function = $\dfrac{K_T}{S\,(s\,Ra\,j_m + K_T\,K_b)}$ where $j_m = 6*10^{-4}$ kg $m^2$

# Laboratory Experiment Number-5(ME354A): Study of PID Control Systems*

Vibration and Dynamics Laboratory, IIT Kanpur

## Experiment-5(a): Study of temperature control system

***Study of PID Control System*** consists of total three experiments. At first we will study *Temperature control of water using PID controller*. In this experiment we will control the temperature of the water using a PID controller.

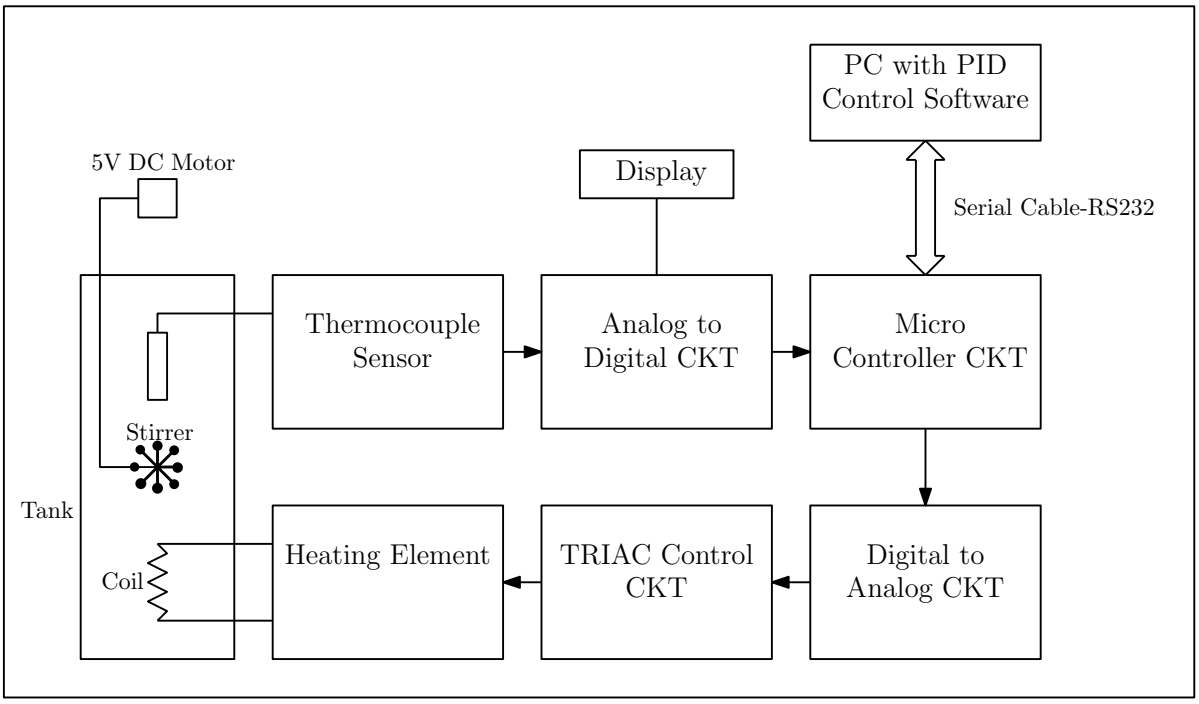



Figure 1: Schematic of the total system.

The schematic of the whole system is shown in Fig.(1). In this experiment heating of the water

*This particular document is prepared by Santanu Das (TA@ME354A, mail id: santanu@iitk.ac.in). If you find any discrepancy, please contact him.

is controlled by the PID controller but cooling is not controlled by PID controller. The cooling of water takes place through natural convection (between the tank water and ambient). So, for the cooling of water there is no control.

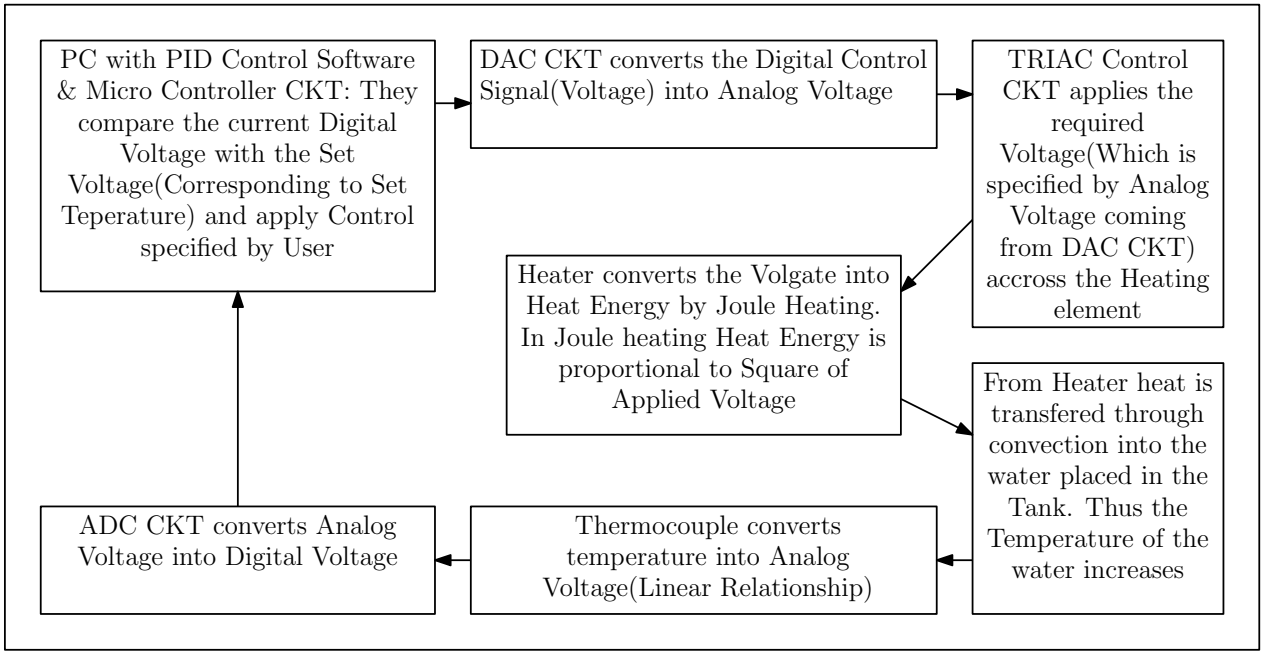



Figure 2: Schematic of control process during heating of water.

Schematic of feedback control process during heating of water is shown in Fig.(2). The corresponding block diagram is shown below[1].

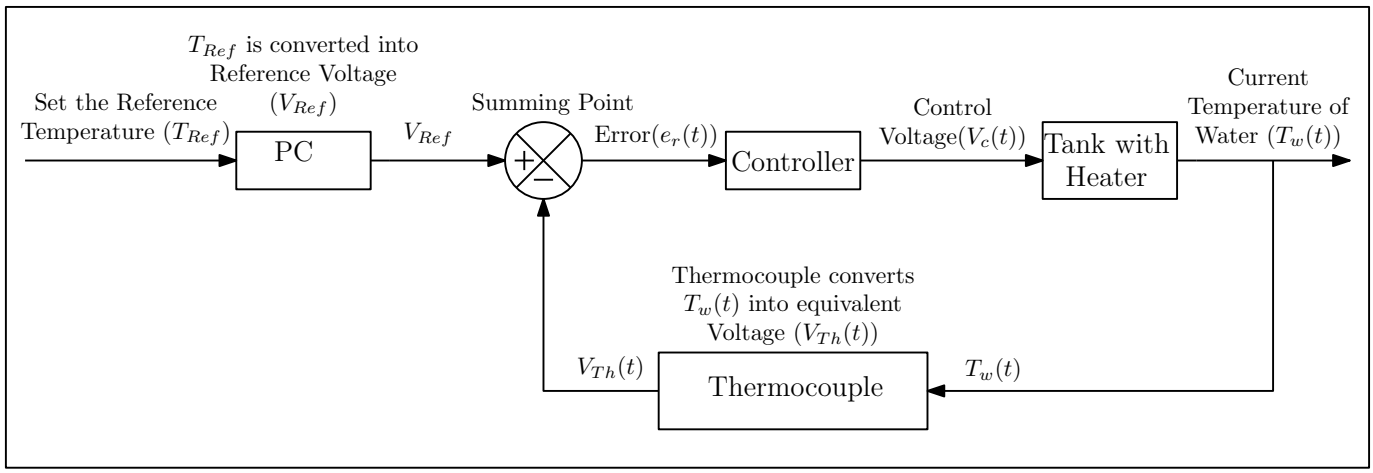



Figure 3: Block diagram of control process during heating of water.

When the error signal ($e_r(t) = V_{ref} - V_{Th}(t)$) is positive, then the controller becomes active and

[1]The maximum control voltage ($(V_c)_{max}$) is set as 10 volts.

control the heating process. And when $(e_r(t) = V_{ref} - V_{Th}(t))$ is negative, the PC does not send any signal to the controller. So the controller does not control the cooling of water.

### Governing equation of the whole system during heating and cooling of water

**Water Tank:** Let us consider the water tank as a system and try to obtain the governing equation of water heating. Now the governing differential equation will be obtained by applying heat energy

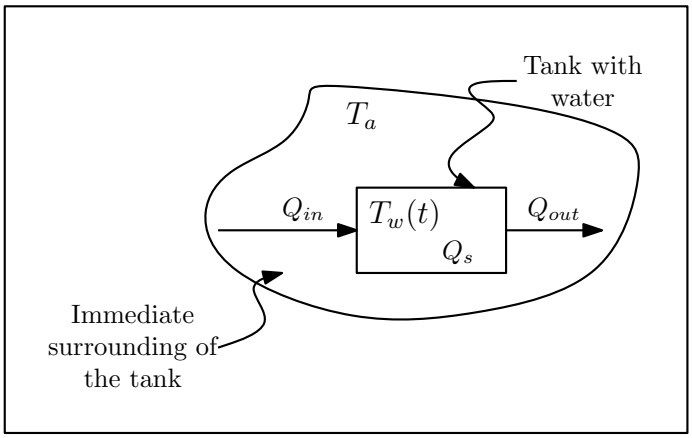



Figure 4: Water tank and immediate surrounding.

balance. Energy supplied by the heater per second $(Q_{in})$ - Energy lost to the immediate surrounding by natural convection per second $(Q_{out})$= Change in the internal energy of the system per second $(Q_s)$.

As heat energy supplied by Joule heating, $Q_{in} \propto (V_c(t))^2$. So $Q_{in} = \alpha(V_c(t))^2$, where $\alpha$ is proportionality constant. Let $U$ be the overall heat transfer coefficient for this natural convection and $T_a$ be the surrounding temperature, so $Q_{out} = U(T_w(t) - T_a)$. Let $m$ be the mass of water and $C$ be the specific heat of water, so $Q_s = mC\frac{dT_w(t)}{dt}$. Now the energy balace equation will look like,

$$\begin{aligned} Q_{in} - Q_{out} &= Q_s, \\ \implies \alpha(V_c(t))^2 - U(T_w(t) - T_a) &= mC\frac{dT_w(t)}{dt}, \\ \implies \frac{dT_w(t)}{dt} &= A(T_a - T_w(t)) + B(V_c(t))^2, \end{aligned}$$

where $A = \frac{U}{mC}$ and $B = \frac{\alpha}{mC}$.

**Thermocouple:** Thermocouple senses the current temperature of the water $(T_w(t))$ and linearly converts into voltage $(V_{th}(t))$. So, $V_{Th}(t) = \beta T_w(t)$, where $\beta$ is proportionality constant.

**PC:** PC converts linearly the reference temperature $(T_{Ref})$ into reference voltage $(V_{Ref})$. Here the proportionality constant will have to be same as of thermocouple, only then comparing two different

voltage signals will be possible at summing point. So, $V_{Ref} = \beta T_{Ref}$.

**Summing Point:** At summing point, two voltage signals from thermocouple and PC are compared and the error signal/voltage ($e_r(t)$) is obtained. So, $e_r(t) = V_{Ref} - V_{Th}(t) = \beta(T_{Ref} - T_w(t))$. If $e_r(t)$ is positive, then it goes to controller else the controller does not get any signal and becomes inactive and heating process stops.

**PID Controller:** The schematic of PID controller is shown below.

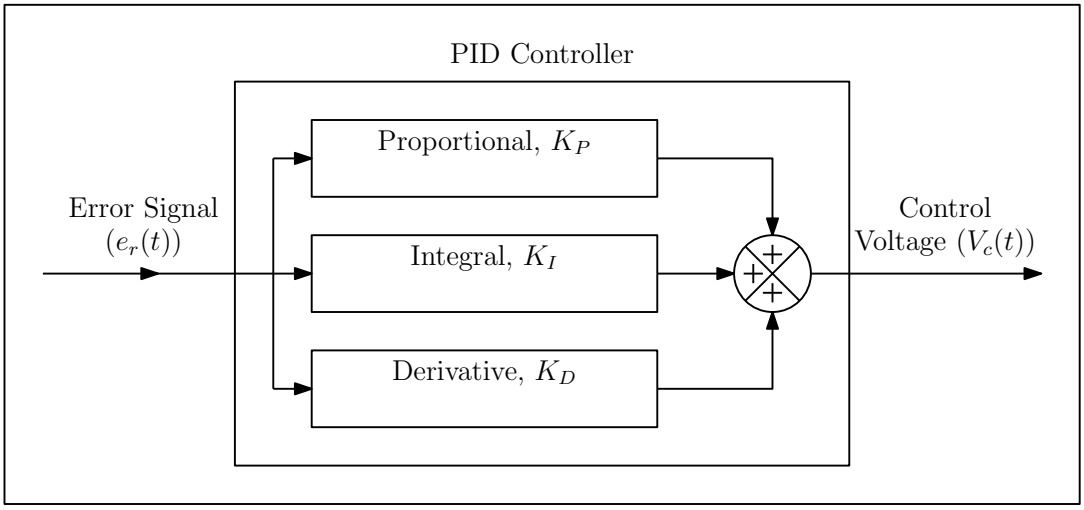



Figure 5: Schematic of PID Controller .

The controller applies the required control voltage (based on error signal) across the heater when the error signal is fed to the PID controller. Now, the control voltage($V_c(t)$) is the function of error signal($e_r(t)$). With Proportional, Integral and Derivative controllers ON, the control voltage expression will look like,

$$V_c(t) = K_P e_r(t) + K_I \int_0^t e_r(t)\,dt + K_D \frac{de_r(t)}{dt},$$

$$\implies V_c(t) = K_P(V_{Ref} - V_{Th}(t)) + K_I \int_0^t (V_{Ref} - V_{Th}(t))\,dt + K_D \frac{d(V_{Ref} - V_{Th}(t))}{dt},$$

$$\implies V_c(t) = \beta \left( K_P(T_{Ref} - T_w(t)) + K_I \int_0^t (T_{Ref} - T_w(t))\,dt + K_D \frac{d(T_{Ref} - T_w(t))}{dt} \right).$$

So the governing equation of the whole system during heating of water is given by,

$$\frac{dT_w(t)}{dt} = A(T_a - T_w(t)) + \beta^2 B \left( K_P(T_{Ref} - T_w(t)) + K_I \int_0^t (T_{Ref} - T_w(t))\,dt + K_D \frac{d(T_{Ref} - T_w(t))}{dt} \right)^2.$$

Similarly, the governing equation of the whole system during cooling through natural convection is given by,

$$\frac{dT_w(t)}{dt} = A(T_a - T_w(t)).$$

So this system is nonlinear.

### The assumed linear system

**Objective** of this experiment is to get the system parameters with the help of some readings obtained from the experiment and some formulas obtained from the theory of the experiment. Now we do not have any theory regarding the nonlinear system. We only have theory regarding linear system and we also know the response of linear system with respect to step input.

**Assumptions** regarding the linear system are,

1. We assume that the heat energy supplied to the water is proportional to applied voltage. So, $Q_{in} = \alpha V_c(t)$. The same proportionality constant ($\alpha$) is used for simplicity.

2. We assume that heating and cooling both are controlled by PID controller.

**Transfer function** is very helpful in studying the linear system.

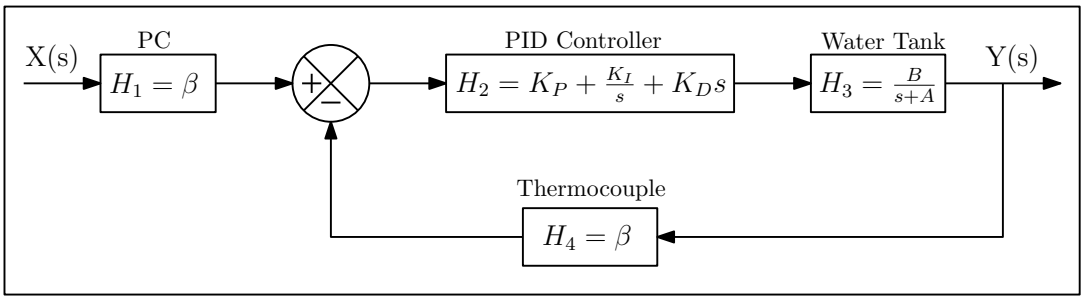



Figure 6: Transfer function of all the blocks.

The transfer function of the whole system is given by,

$$
\begin{aligned}
H(s) &= \frac{Y(s)}{X(s)}, \\
\implies H(s) &= \frac{H_1 H_2 H_3}{1 + H_2 H_3 H_4}, \\
\implies H(s) &= \frac{\beta B(K_D s^2 + K_P s + K_I)}{(\beta B K_D + 1)s^2 + (\beta B K_P + A)s + \beta B K_I},
\end{aligned}
$$

where $X(s) = \mathcal{L}[x(t)]$ and $Y(s) = \mathcal{L}[y(t)]$ [2].

In this system $x(t)$ and $y(t)$ is defined as, $x(t) = T_{Ref} - T_a$ and $y(t) = T_w(t) - T_a$. Here, $x(t)$ is a step input.

Clearly, the system is 2nd order as the denominator of $H(s)$ has a quadratic polynomial in $s$. The response of any 2nd order system is defined by the two parameters. Those are natural frequency ($\omega_n$) and damping ratio ($\zeta$) of the system.

[2] $\mathcal{L}$ = Symbol for Laplace transform.

### Response of a 2nd order system with respect to a step input

The standard form of closed-loop transfer function of a 2nd order system is given by,

$$H(s) = \frac{\text{Output}}{\text{Input}} = \frac{C(s)}{R(s)} = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n + \omega_n^2}.$$

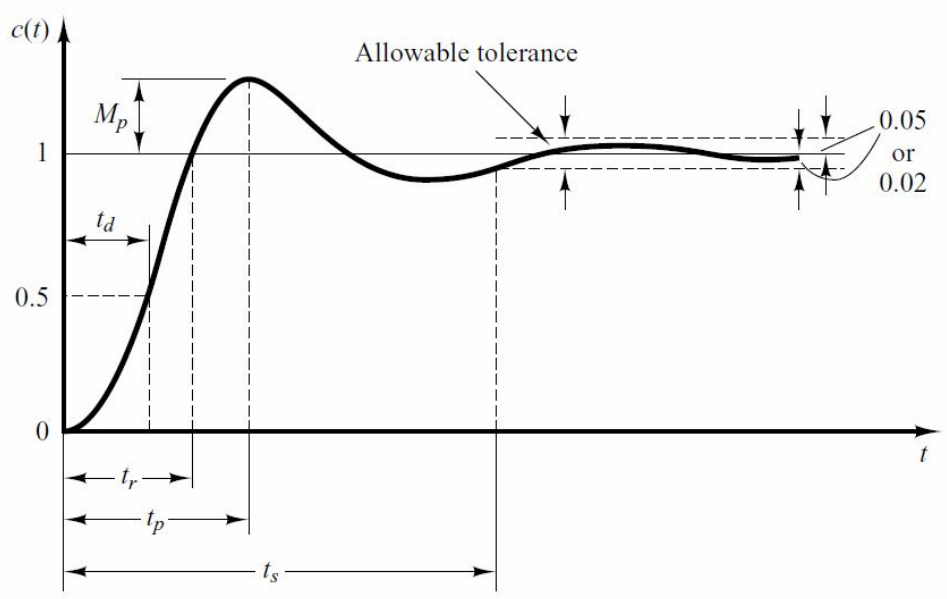



Figure 7: Unit-step response of 2nd order system.

The unit-step ($R(s) = \frac{1}{s}$) response of a 2nd order system (with standard form of transfer function) is given by,

$$c(t) = \mathcal{L}^{-1}[C(s)] = 1 - \frac{\exp(-\zeta\omega_n t)}{\sqrt{1-\zeta^2}} \sin(\omega_d t + \phi),$$

where $\omega_d = \omega_n\sqrt{1-\zeta^2}$ and $\phi = \cos^{-1}\zeta$.

To calculate the natural frequency ($\omega_n$) and damping ratio ($\zeta$) of the system, we need to know two quantities. Those are maximum overshoot ($M_p$) and peak time ($T_p$).

**Peak time** ($T_p$) is the time at which maximum response ($c_{max}$) occurs. The expression corresponding to peak time in terms of $\omega_n$ and $\zeta$ is given by, $T_p = \frac{\pi}{\omega_d}$.

**Maximum overshoot** ($M_p$)**:** The expression corresponding to maximum overshoot in terms of $\omega_n$ and $\zeta$ is given by, $M_p = \frac{c_{max}-c(\infty)}{c(\infty)} = \exp\left(-\frac{\pi\zeta}{\sqrt{1-\zeta^2}}\right)$. In this experiment maximum overshoot ($M_p$) is defined as, $M_p = \frac{(T_w(t)-T_a)_{max}-(T_{Ref}-T_a)}{(T_{Ref}-T_a)} = \frac{T_w(T_p)-T_{Ref}}{(T_{Ref}-T_a)}$.

***Note:*** For two consecutive experiments, let us assume to apply PI control first and then PID control. Ideally $T_a$ must be equal to $T_{Initial}$. But this is only true for the first experiment (i.e., for PI). Now, water has already heated up and we should not perform next experiment immediately. We should wait until the temperature of the water comes down to $T_a$. But due to shortage of time, we perform

next experiment immediately. So, for the second experiment $T_a \neq T_{Initial}$. For the 2nd experiment onwards, we use $T_{Initial}$ as $T_a$ for $M_p$ calculation. Here, $T_{Initial}$ is the initial temperature of water at the starting of each new control experiment.

**Data to be collected from the experiment:** To calculate natural frequency ($\omega_n$) and damping ratio($\zeta$), we need to collect the values of peak time ($T_p$), maximum water temperature ($(T_w(t))_{max} = T_w(T_p)$), reference temperature ($T_{Ref}$), and ambient temperature ($T_a$) or Initial temperature ($T_{Initial}$).

### Lab Report

The lab report should consist of the followings, [3]

1. Plots generated in lab using VIM-PID software. 20 marks
2. Calculations of natural frequency ($\omega_n$) and damping ratio ($\zeta$) for each experiments. 15 marks
3. MATLAB simulation of unit-step response of the second order system. (Use standard form of transfer function and previously calculated values of natural frequency ($\omega_n$) and damping ratio ($\zeta$).) Print out of both codes and plots for each cases. 15 marks
4. Discussions. The following points should be included in the discussions. 20 marks
   (a) How close is our heating system to actual PID control?
   (b) Does MATLAB plot matches with the experimental plot and Why?
   (c) How did we lower the temperature of the system?
   (d) What did you learn from this experiment?

## Experiment-5(b): Magnetic levitation system and Experiment-5(c): Ball and beam system

These two different experiments are only for demonstration. For the regarding theories, read the given manuals.

**Lab report** should consist of the following,

- Draw the schematic of physical set ups for both the experiments and show the flow directions of the signals with arrows. 15+15=30 marks

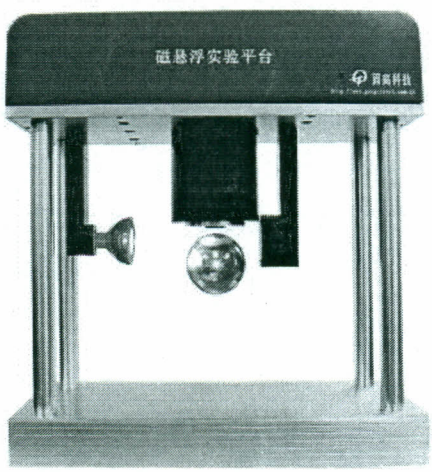
[3] We shall perform PI and PID control. If time permits we shall also perform P control.

GOOGOLTECH
固 高 科 技

# *Magnetic Levitation System*

# INSTRUCTION MANUAL

Suitable for GML Series




*The 4th Edition, April, 2007*

**Headquarter**

Room 3639, Annex Building, HKUST,
Clear Water Bay, Kowloon, Hong Kong
Tel.: (852) 2358 1033, (852) 2719 8310
Fax: (852) 2719 8399

**Googol Shenzhen**

2/F, West Wing, IER Building, High-Tech Park
Nanshan, Shenzhen, China
Tel: (86)755 2697 0817; (86)755 2697 0838
Fax: (86)755 2697 0846
http://www.googoltech.com.cn

## Copyright Statement



## Disclaimer

Googol Technology reserves the right of modifying the products and product specifications described in this manual without notification in advance.

Googol Technology is not responsible for any direct, indirect, special, incidental or consequential loss or liability caused by using this manual or product incorrectly.

## Trademarks

Windows and Microsoft are registered trademarks of Microsoft Corporation.

MATLAB is a registered trademark of MathWorks Inc.

| | |
|---|---|
|  **Warning** | ***Machinery in motion can be dangerous! It is the responsibility of the user to design effective error handling and safety protection as part of the machine. Googol Technology shall not be liable or responsible for any incidental or consequential damages.*** |

# TABLE OF CONTENTS


Copyright Statement ..... 1

Disclaimer ..... 1

Trademarks ..... 1

**TABLE OF CONTENTS ..... 2**

**FOREWORD ..... 4**

Welcome to the World of Googol Technology Educational Products ..... 4

For Additional Information about Googol Technology Products ..... 4

For Technical or Customer Support ..... 4

**Chapter 1 Cautions ..... 5**

1.1 Cautions on Safety ..... 5

1.2 Cautions on unpacking ..... 5

1.3 Cautions on using ..... 5

**Chapter 2 Overview ..... 6**

2.1 Configuration of Magnetic Levitation System ..... 6

2.1.1 MLS Main Body ..... 7

2.1.2 Control Platform ..... 7

**Chapter 3 System Installation ..... 8**

3.1 Installing Data Acquisition Card ..... 8

3.2 Wiring ..... 8

3.3 Software Installation ..... 9

3.3.1 Installing the Data Acquisition Card driver ..... 9

3.3.2 Installing System Demo and Experimental Software ..... 9

3.3.3 Real-Time Kernel Installation and Compiler Selection ..... 10

**Chapter 4 Real Time Control in MATLAB Simulink ..... 13**

4.1 Operation Procedures (Take PID control demo for example) ..... 13

**Chapter 5 MAINTENANCE AND TROUBLESHOOTING ..... 17**

5.1 Maintenance ..... 17

5.2 Troubleshooting ..... 17

# FOREWORD

## Welcome to the World of Googol Technology Educational Products

Thank you for purchasing GML1001, Magnetic Levitation Educational Control System. This is a complete turnkey solution that includes a mechanical plant unit, a high-performance digital control system, a user-friendly software interface and comprehensive reference materials. It is suitable for teaching basic practical motion and automatic control courses to students as well as for doing scientific research in related areas.

To repay you, our customer, for your trust in our products, we will help you to build your own control systems by providing our first-class motion control equipment, complete after-sales services, and effective technical support.

## For Additional Information about Googol Technology Products

Googol Technology is online on the Internet at http://www.googoltech.com. Our Web pages provide information on the company and its products, including access to technical information and documentation, products overviews, and product announcements.

You may also obtain additional information about Googol Technology and its products in request for information by phone (+852) 2358-1033.

## For Technical or Customer Support

You can reach our Customer Support group in the following ways:

- E-mail questions to Support@googoltech.com
- Fax questions to (+86) 755-26970846
- Call us at (+86) 755-26970825
- Contact your local Googol sales office or authorized Googol distributor
- Send questions by mail to:
  Googol Technology (sz) Ltd
  Room W211 IER Building, South Area ShenZhen Hi-tech Industrial Park.
  Post code: 518057

# Chapter 1 Cautions

## 1.1 Cautions on Safety

Please read this manual carefully before installing, using and maintaining.

| | |
|---|---|
| **Danger** | Incorrect operation may result in such a dangerous situation as death or serious injury. |
| **Caution** | Incorrect operation may result in such a dangerous situation as medium or slight injury. |
| **Prohibition** | Prohibition<br>What should not be done by all means.•• |
| **Compulsion** | Compulsion<br>What should be done by all means.•• |

## 1.2 Cautions on unpacking

- After unpacking, please check if the product type marked on the package is consistent with your purchase.
- Check if there is any mechanical damage on the product during transportation.
- Check carefully if the accessories are complete (According to the packing list).
- If there is mechanical damage on the product, or any item is missing in the package, please do not use it and contact Googol Technology or our agent immediately.

## 1.3 Cautions on using

- Make sure that the magnetic levitation control main body should be placed on a level surface; it should never be tilted at an angle.

- Make sure that the light source supply is stable when it is in use.

# Chapter 2 Overview

## 2.1 Configuration of Magnetic Levitation System

As an automatic control experiment system, Magnetic Levitation System (MLS) is able to satisfy the needs of control elements, automatic control theory, modern control theory and control system courses designed for engineering colleges. It is the educational experimental platform that Googol Technology developed to satisfy the various teaching needs of automatic control courses.

MLS is a platform for the study of magnetic levitation technology. It's mainly composed of solenoid, position sensor, amplifier/compensation device (driver), digital controller (PCI1711) and control object ***steel ball***. It is a typical magnetized levitation system. The system can be divided into two parts, MLS main body, control platform with data acquisition card and PC. System block diagram is shown in Figure 2-1.

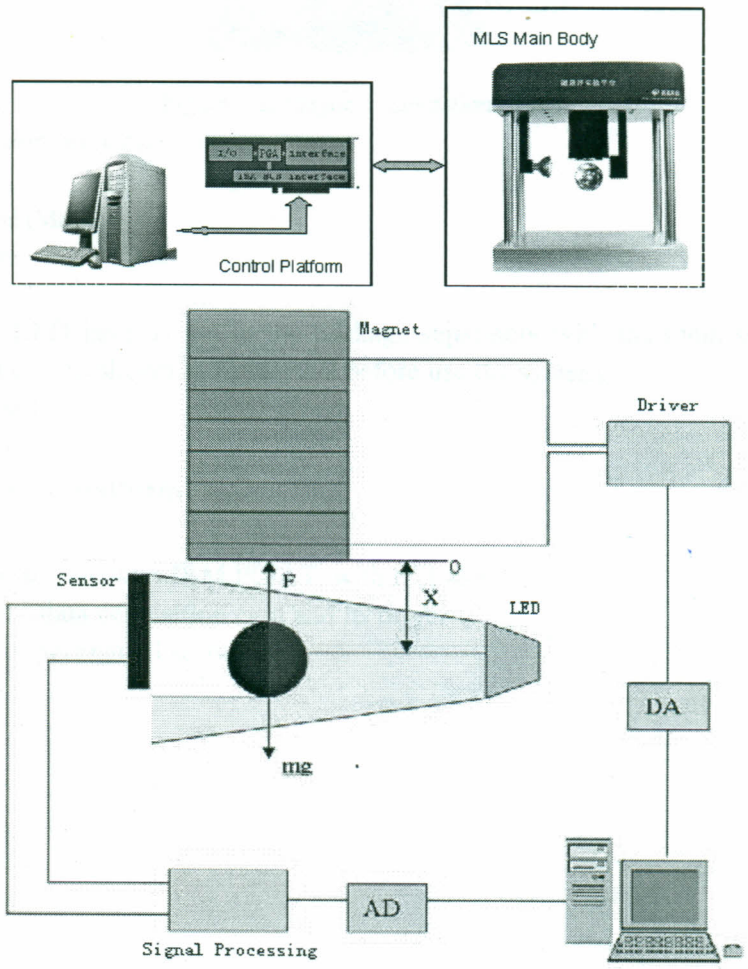



Figure 2-1 MLS block diagram

### 2.1.1 MLS Main Body

By passing a certain amount of current through the solenoid, an electromagnetic force will be generated. By controlling the current that passing through the solenoid such that the electromagnetic force equals to the weight of the steel ball, the steel ball can levitate in the air while it is in equilibrium state. But this state is an unstable equilibrium. The system is an open-loop unstable system.

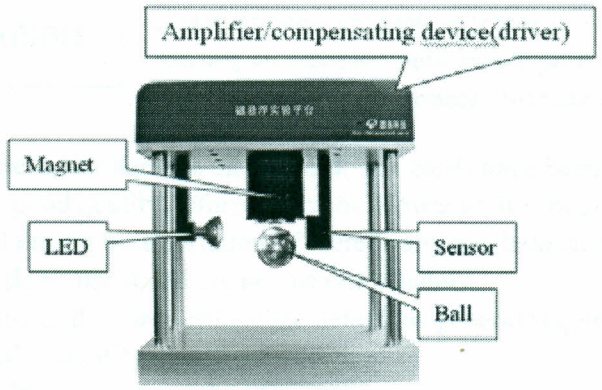



Figure 2-2. Magnetic Levitation System

GML1001 main body components:

- Driver
- Solenoid (Magnet)
- Sensor
- LED

**Note:** The LED lamp is put in the package separately with the main body; user should install it as shown in Figure 2-2 before use the system.

- Steel ball

### 2.1.2 Control Platform

- PC compatible with IBM PC/AT, with PCI slot.
- PCI1711 data acquisition card and its driver
- Demo experimental software.

# Chapter 3 System Installation

## 3.1 Installing Data Acquisition Card

**Caution**

*Before unpacking the data acquisition card from antistatic box, please discharge static electricity from your body. You can do so by touching an unpainted metal surface on the computer chassis. To avoid damaging the card, please don't touch the chips by hand.*

- Check if there is any damage on the card. The cards have been tested repeatedly to guarantee good quality before shipment. However, it is possible that the card get damaged during transportation. If there is any mechanical damage on the card, please don't use it and contact us immediately.
- Before shipment, the card settings are default as general requirement. Users are not suggested to modify them.
- Turn off the PC and remove the PC cover.
- Insert data acquisition card into a free PCI slot of the computer.
- Secure the card bracket with screw.
- Replace the PC cover.

## 3.2 Wiring

- Check if the power switch of MLS main body is "OFF" or not. If not, switch to "OFF".
- Before use the system, user should install the LED lamp which is put in the package separately with the main body, as shown in Figure 2-2.
- Connecting the data acquisition card in PC and MLS experiment main body with a 68pin to 15Pin cable.

**Danger**

*The cable used in next step is connected to high voltage power supply; any damage on it may result in physical injury or death.*

- A power cable should be connected between MLS main body and 220V AC power supply.
- There is an Ana/Dig switch at the interface of the main body. Digital mode should be selected when data acquisition card is used to control the magnetic levitation system, while the Analog mode is reserved for future use by users who can develop their own analog controller.

## 3.3 Software Installation

The demo and experimental software are operated in Windows system. Visual C++6.0 ,MATLAB 6.5 and related toolbox are required.

### 3.3.1 Installing the Data Acquisition Card driver

After installing card and wiring, turn on the PC. The system will automatically detect new hardware and prompts user to insert the product CD to install the driver. Insert the product CD from Data Acquisition Card box which will automatically run,**Please install "Advantech Device Manager"before installing any other items.** Follow the wizard to complete the installation. Then go to the Individual Driver and select PCI series to install the PCI-1711driver. Follow the wizard to complete the installation.

After completing the installation, open device manager of your PC and check if there is a hardware Advantech DA&C I/O cards with Advantech PCI1711S Device attached. If there is, the data acquisition card has been installed successfully; otherwise, it has to be reinstalled. Please refer to the manuals of data acquisition card in CD for more details.

### 3.3.2 Installing System Demo and Experimental Software

In Windows, open **Explore**, search the "**MATLAB Simulink and Real-time Control Programs**" folder in CD, then open the "**setupx.xx**" folder and run **setup.exe**, Follow the wizard to complete the installation. After completing the installation, you can run demo in the MATLAB toolbox, as in Figure3-1.



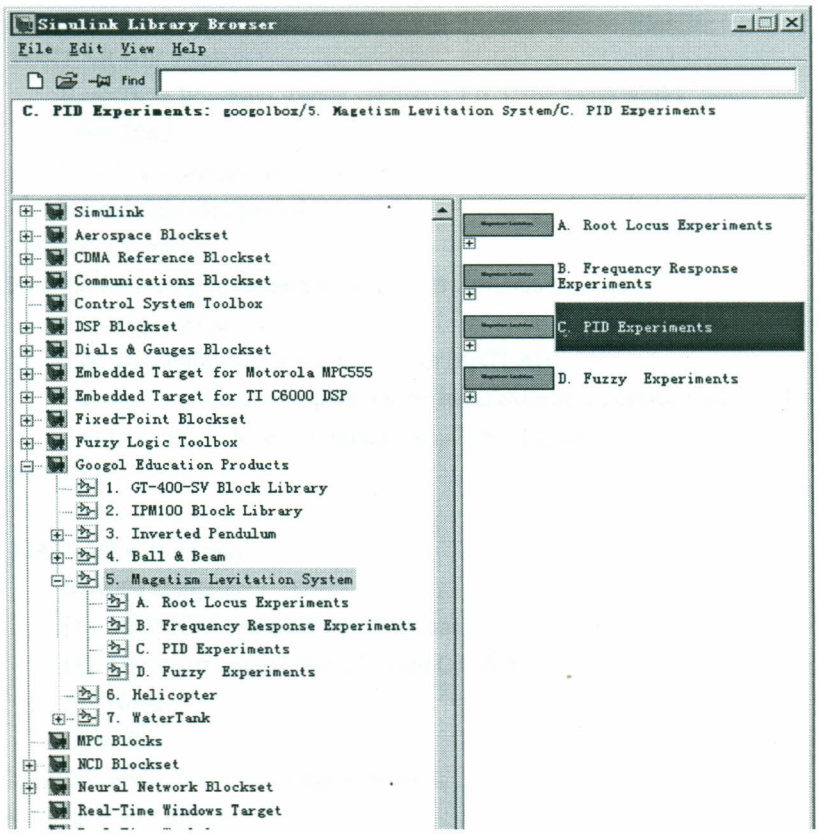



Figure 3-1 MATLAB Interface

Please read carefully the next chapter before system running.

### 3.3.3 Real-Time Kernel Installation and Compiler Selection

After installing MATLAB/Real-Time Windows Target and Visual C/C++, user should install Real-Time Windows Target kernel and select C compiler in MATLAB before using the real control software (Details please refer to MATLAB help). The steps are as follow:

- **Real-Time Kernel Installation.**

1• Type the following command in MATLAB command window

rtwintgt -install

The following information can be seen in MATLAB:

You are going to install the Real-Time Windows Target kernel.

Do you want to proceed? [y] :

2• Continue installation, enter

y

The following information will be displayed on the screen:

The Real-Time Windows Target kernel has been successfully installed.

If there is a notice to restart the computer, you must restart the computer to ensure the software function correctly.

3• Check if the kernel is correctly installed. Enter:

rtwho

MATLAB will display:

Real-Time Windows Target version 2.2 (C) The MathWorks, Inc.

1994-2002

MATLAB performance = 100.0%

Kernel timeslice period = 1 ms

➢ **Select C Compiler.**

1• Type the following command in MATLAB command window:

mex -setup

The following information can be seen in MATLAB:

Please choose your compiler for building external interface (MEX) files. Would you like mex to locate installed compilers? ([y]/n):

2• Enter

y

MATLAB will display:

Select a compiler:

[1]: WATCOM compiler in c:\watcom

[2]: Microsoft compiler in c:\visual C++6.0

[0]: None

Compiler:

3• Select Microsoft compiler, enter number 2.

MATLAB will display:

Please verify your choices:

Compiler: Microsoft 5.0

Location: c:\visualAre these correct? ([y]/n)

4• Enter

y

MATLAB will display:

The default options file:

"C:\WINNT\Profiles\username\Application

Data\MathWorks\MATLAB\mexopts.bat" is being updated.

After installing Real-Time Windows Target kernel and selecting C compiler, uses may start using the real control software.

**Compulsion**

- Before wiring, dissembling or installing, do please cut off all the power of system.
- Before using, please check carefully the wiring of each component. Make sure there is no slack or disconnection.
- Check if there is any obstruction in sensor and the LED.

**Prohibition**

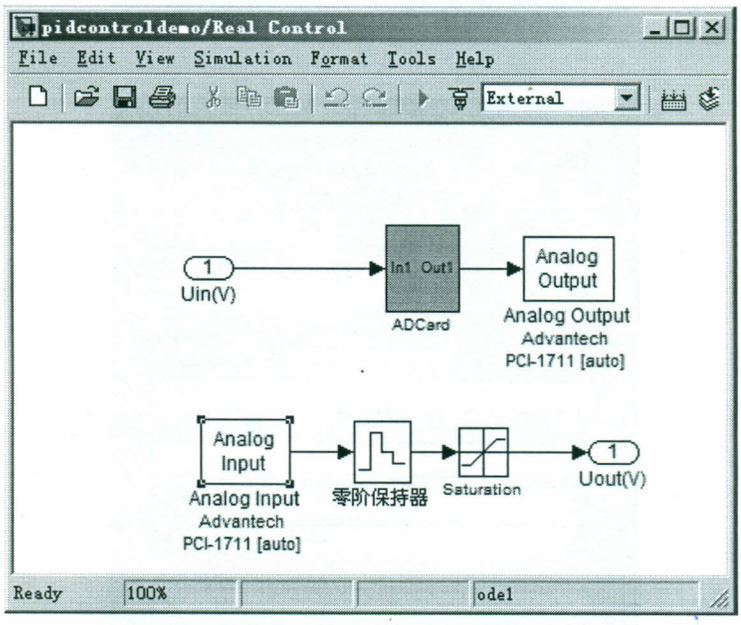
- Frequent change in the background light intensity is prohibited.
- Avoid interference from glare when the system is operating. When photographing, turn off the flash lamp.

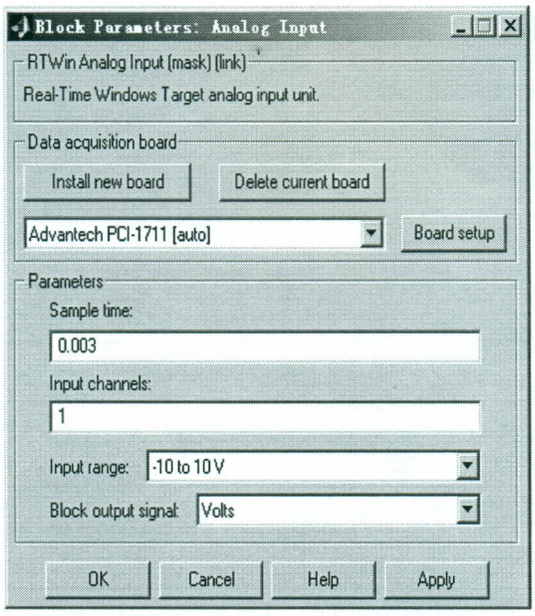
# Chapter 4 Real Time Control in MATLAB Simulink

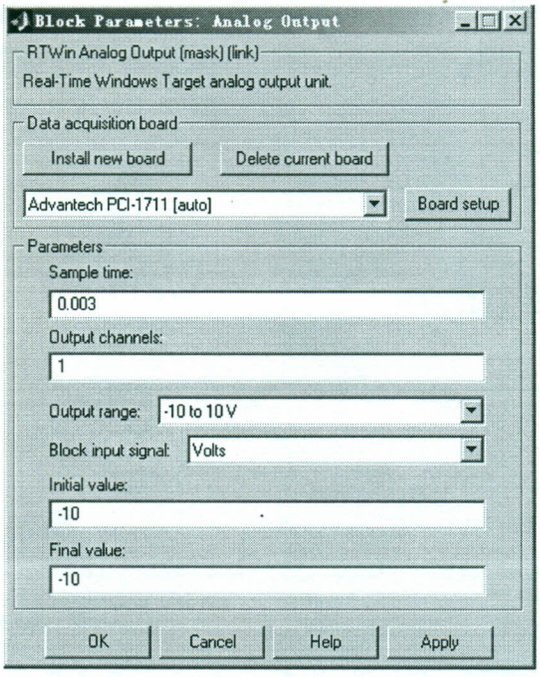
## 4.1 Operation Procedures (Take PID control demo for example)

1) For the first time use, please install the interface of the PCI1711 card for MATLAB. • • MATLAB toolbox directory: Simulink\Googol Education Products\Magnetic Levitation System \ PID Experiments\PID Control Demo• •

After completing the installation of all drivers and also the test of PCI1711 is ok, open "PID Control Demo", get into the"Real Control" module, and double click the "Analog Input" and "Analog Output" modules. Separately "Install new board\Adbantech\PCI1711", as shown in the following figure:





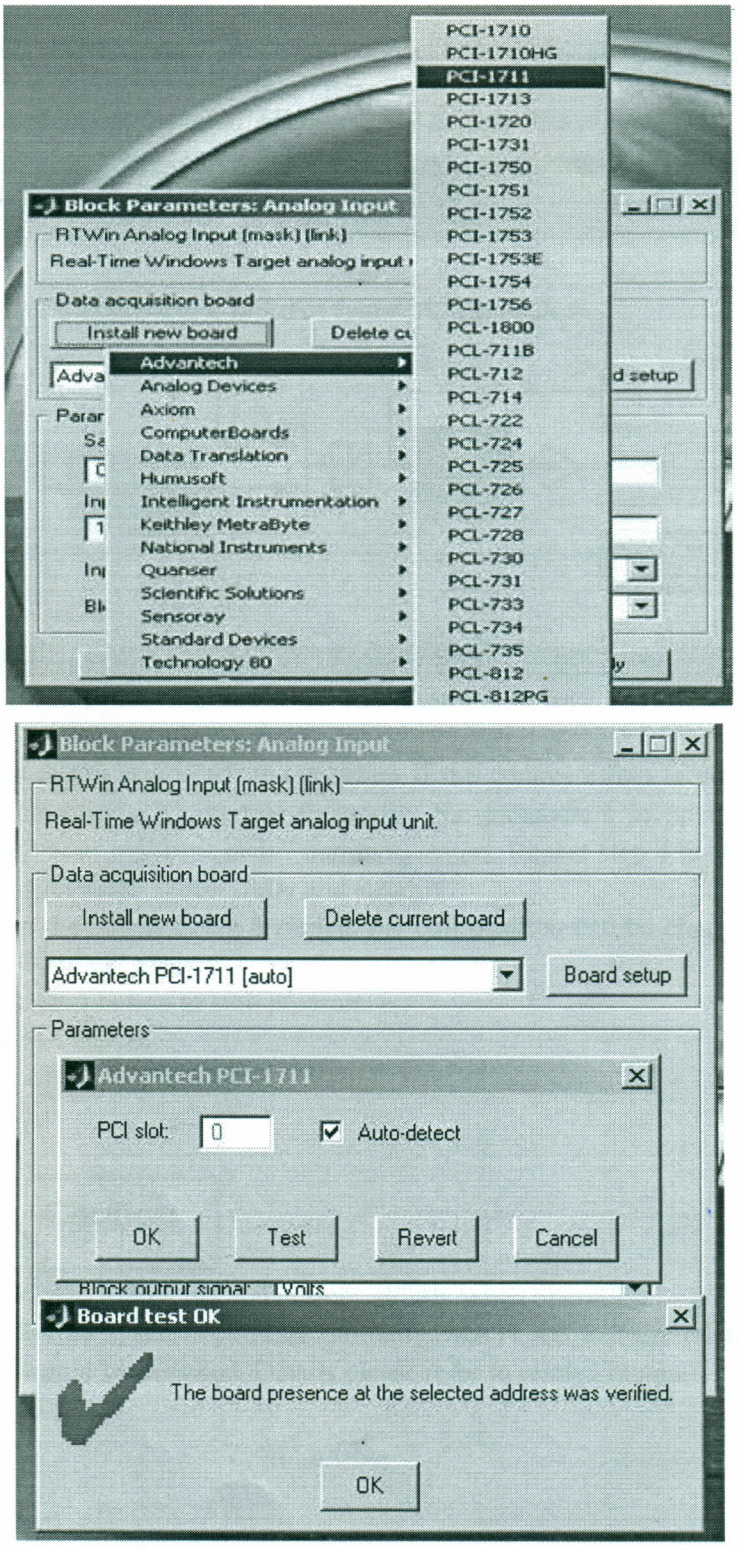



Figure4-1 Setup the PCI1711's interface for MATLAB

2) After it's ok, compile the "PID Control Demo" model.

3) Select the "External" mode and connect to target by clicking 

4) Click "Start real-time code" button  and start real time control.

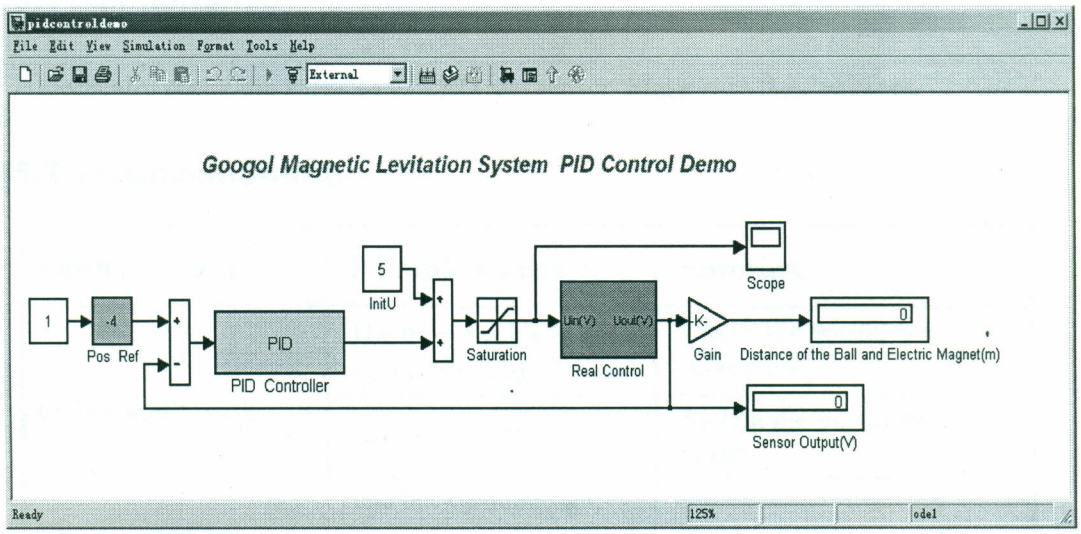



Figure 4-2 Real Time Control Diagram

5) Press STOP button to stop program if the control effect is not satisfactory. Double click PID module to modify the parameters in open window and press "Start real-time code" button to restart. Repeat step 4 to step 5 until the ball is levitated successfully and steadily.
6) When the ball is being levitated, the ball position can be changed by tuning the "Pos Ref" slide.
7) Click Stop button to stop program.
8) Exit MATLAB.
9) Turn off the PC and MLS.

**Caution**

- Please follow the above steps strictly when operating.
- Before system operating normally, MATLAB and the related toolboxes should be installed. Details please refer to related manual.

# Chapter 5 MAINTENANCE AND TROUBLESHOOTING

## 5.1 Maintenance

Periodically check the surfaces of LED and sensor. Make sure they're clean.

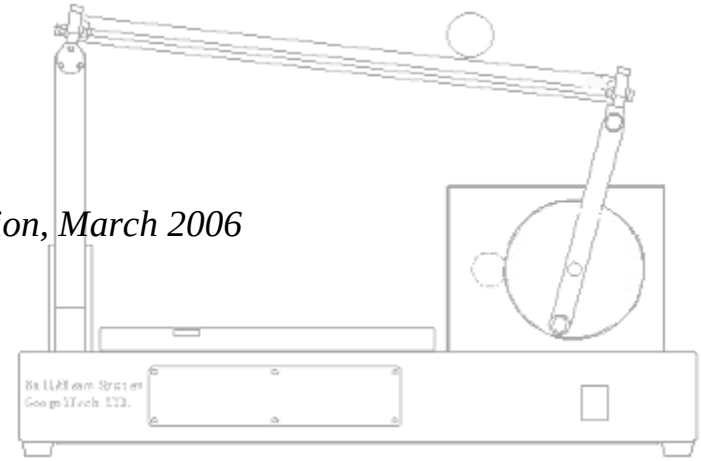
## 5.2 Troubleshooting

| Faulty Action | Possible Cause | Solution |
|---|---|---|
| LED doesn't work | The power cable is not well connected. | Check the wiring and reconnect. |
| | Power is "OFF". | Switch the Power switch to "ON" |
| The ball can't be levitated steadily when the system is running. | The background light source varies dramatically.<br>PID parameters isn't the best | Readjust PID parameters |

GOOGOL TECHNOLOGY

# Ball & Beam GBB1004

## USER'S GUIDE & EXPERIMENT MANUAL



*Second Edition, March 2006*


**Headquarter**

香港九龍清水灣香港科技大學新翼大樓 3639 號
Room 3639, Annex Building, HKUST,
Clear Water Bay, Kowloon, Hong Kong
Tel.: (852) 2358 1033, (852) 2719 8310
Fax: (852) 2719 8399

**Googol Shenzhen**

深圳市南山區高新科技園深港產學研基地西二樓
2/F, West Wing, IER Building, High-Tech Park
Nanshan, Shenzhen, China
Tel: (86)755 2697 0817; (86)755 2697 0838
Fax: (86)755 2697 0846
http://www.googoltech.com..cn

## Copyright Statement



## Disclaimer

Googol Technology reserves the right of modifying the products and product specifications described in this manual without notification in advance.

Googol Technology is not responsible for any direct, indirect, special, incidental or consequential loss or liability caused by using this manual or product incorrectly.

## Trademarks

Windows and Microsoft are registered trademarks of Microsoft Corporation.

IPM and IPM Motion Studio are registered trademarks of Technosoft Ltd.

MATLAB is a registered trademark of MathWorks Inc.

| | |
|---|---|
|  **Warning** | ***Machinery in motion can be dangerous! It is the responsibility of the user to design effective error handling and safety protection as part of the machine. Googol Technology shall not be liable or responsible for any incidental or consequential damages.*** |

# TABLE OF CONTENTS


COPYRIGHT STATEMENT I
DISCLAIMER I
TRADEMARKS I

**TABLE OF CONTENTS II**

**FOREWORD 1**

WELCOME TO THE WORLD OF GOOGOL TECHNOLOGY EDUCATIONAL PRODUCTS 1
FOR ADDITIONAL INFORMATION ABOUT GOOGOL TECHNOLOGY PRODUCTS 1
FOR TECHNICAL OR CUSTOMER SUPPORT 1

**READ THIS FIRST 2**

ABOUT THIS MANUAL 2
INTENDED AUDIENCE 2
PREREQUISITES 2
CONTENTS DESCRIPTION 2
CONVENTIONS USED IN THE DOCUMENT 3

**CHAPTER 1 OVERVIEW 4**

1.1 OUTLINE 4
1.2 BALL & BEAM FEATURES & BENEFITS 4
1.3 BALL & BEAM FUNCTIONAL COMPONENTS 4
1.3.1 MECHANICAL PLANT 6
1.3.2 INTELLIGENT DRIVE 6
1.3.3 PC-BASED CONTROL SOFTWARE 7

**CHAPTER 2 GETTING STARTED 8**

2.1 THE BALL & BEAM PACKAGE 8
2.2 ELEMENTS YOU NEED 8
2.3 GETTING STARTED IN MATLAB SIMILINK 8
2.3.1 STEP 1: CONNECTING BALL&BEAM 8

2.3.2 STEP 2: INSTALLING BALL&BEAM CONTROL TOOLBOX 9
2.3.3 STEP 3: SET PARAMETERS 10

**CHAPTER 3 SYSTEM MODELING 16**

3.1 MECHANICAL MODEL OF BALL&BEAM 16
3.2 MODELING THE BALL AND BEAM EXPERIMENT IN SIMULINK 17
3.3 ELECTRICAL MODEL 22
3.4 CONTROL MODEL 24

**CHAPTER 4 SAMPLE EXPERIMENTS 27**

4.1 DATA COLLECTION AND PROCESSING 27
4.2 OPEN-LOOP MODEL OF BALL&BEAM SYSTEM 29
4.3 BALL AND BEAM CONTROL USING PID 31
4.3.1 PROPORTIONAL CONTROL 31
4.3.2 PROPORTIONAL-DERIVATIVE CONTROL 34
4.3.3 ANALYSIS OF PID CONTROL 36
4.4 ROOT LOCUS METHOD 39
4.5 FREQUENCY RESPONSE METHOD 43
4.6 USER DEFINE CONTROLLERS 49

**CHAPTER 5 TROUBLESHOOTING 50**

**APPENDIX A REAL CONTROL IN IPM MOTION STUDIO 52**

STEP BY STEP INSTALLATION USING IPM MOTION STUDIO 52
STEP 1: INSTALLING CONTROL SOFTWARE 52
STEP 2: ESTABLISHING COMMUNICATION 53
STEP 3: RUNNING THE DEMO PROGRAM 53
STEP 4: ANALYZING RESULTS OF THE DEMO EXPERIMENT 57

**APPENDIX B LISTING OF DEMO 60**

PART ONE: VARIABLES DECLARATION 60
PART TWO: THE MAIN CONTROL LOOP 61

**APPENDIX C ELECTRIC DIAGRAM 66**

# FOREWORD

## Welcome to the World of Googol Technology Educational Products

Thank you for purchasing GBB1004, Ball & Beam Educational Control System. This is a complete turnkey solution that includes a mechanical plant unit, a high-performance digital control system together with intelligent drive, a user-friendly software interface and comprehensive reference materials. It is suitable for teaching basic practical motion and automatic control courses to students as well as for doing scientific research in related areas.

To repay you, our customer, for your trust in our products, we will help you to build your own control systems by providing our first-class motion control equipment, complete after-sales services, and effective technical support.

## For Additional Information about Googol Technology Products

Googol Technology is online on the Internet at http://www.googoltech.com. Our Web pages provide information on the company and its products, including access to technical information and documentation, products overviews, and product announcements.

You may also obtain additional information about Googol Technology and its products in request for information by phone (+852) 2358-1033.

## For Technical or Customer Support

You can reach our Customer Support group in the following ways:

- E-mail questions to Support@googoltech.com
- Fax questions to (+86) 755-26970846
- Call us at (+86) 755-26970825
- Contact your local Googol sales office or authorized Googol distributor
- Send questions by mail to:
  Googol Technology (sz) Ltd
  Room W211 IER Building, South Area ShenZhen Hi-tech Industrial Park.
  Post code: 517057

# READ THIS FIRST

## About This Manual

This manual provides basic information on installation and operational principles of GBB1004 Ball & Beam Educational Equipment and guides you through Googol Technology approach to programming motion applications for Ball & Beam using IPM Motion Studio software or MATLAB Simulink.

## Intended Audience

Users who are familiar with basics of motion control are the primary audience for this manual. They include, but not limited to, under- and post-graduate students of technical universities studying automatic control, mechatronics, robotics and other similar disciplines. It is assumed that the audience has a fair understanding of programming and basic knowledge of control hardware operation principles.

## Prerequisites

The manual provides installation procedure and design example based on IPM Motion Studio interface program for Windows. Users should refer to "IPM100 Motion Studio User Manual" provided together with the Ball & Beam package for all questions related to the operation and programming details of this software.

## Contents Description

The manual is organized in the following manner:

- Chapter 1 — Overview — provides an overview of GBB1004 Ball & Beam and its features. It gives brief introduction to the plant's hardware and software that may be useful for further understanding of Ball & Beam operational principles.
- Chapter 2 — Getting Started — leads user through step-by-step GBB1004 installation and turning procedures in MATLAB Simulink including electrical wirings between different parts of the products.
- Chapter 3 — System Modeling — gives overview of operational principles used in Ball & Beam including mathematical modeling of this system and structure of the control program.

- Chapter 4 — Sample Experiment — provides some sample design experiments suitable for Ball&Beam.
- Chapter 5 — Troubleshooting — contains cures for most frequent problems which users may face while using Ball & Beam system.
- Appendix A — contains steps for real time control in IPM Motion Studio.
- Appendix B — contains listing of IPM Motion Studio demo program that realizes a control algorithm for balancing a ball in GBB1004.
- Appendix C — Electric diagram.

## Conventions Used in the Document

- The following terms are used interchangeably together with "GBB1004 Ball & Beam Educational Control Equipment" throughout the manual:

  GBB1004
  Ball&Beam

- Throughout the manual, the following conventions for the given pieces of program code will be used:

  Program code will be displayed like this so the user can easily copy the code into his/her application

# CHAPTER 1 OVERVIEW

## 1.1 Outline

Ball & Beam is a piece of lab equipment designed specially for teaching basic control courses to students and for doing research in motion control field. It can serve as a convenient tool for practical implementation of many classical and modern control system design methods.

## 1.2 Ball & Beam Features & Benefits

The ball-beam system is a frequently encountered example of a nonlinear dynamical system. While the ideal ball-beam system is indeed nonlinear, its practical implementation in the lab has additional non-linearities, including:

- Deadband,
- Backlash introduced by the DC motor and gearbox,
- Discrete position sensing,
- Uneven rolling surface.

These effects are traditionally hard to measure and to model. However they may drastically influence the behavior of the system under control. How to build a robust control for such systems is one of the most important questions of modern theory.

Ball&Beam product offered by Googol Technology is a universal tool to study the control of systems possessed both non-linear dynamics and most common non-linear effects observed in real control systems.

From control point of view the mechanical plant of ball-beam systems poses another important feature – instability. Beside traditional theoretical difficulties, in many cases the study of such systems may involve certain practical risk to the health of researcher because of dangerous and unpredictable behavior of unstable mechanisms. Ball&Beam, being simple, compact and safe, is the best laboratory solution to this problem. Equipped with modern intelligent power drive module and intuitive programming Windows interface it is an ideal tool for control systems modeling.

## 1.3 Ball & Beam Functional Components

Ball & Beam is functionally divided into the following components, as shown in Figure 1-1.

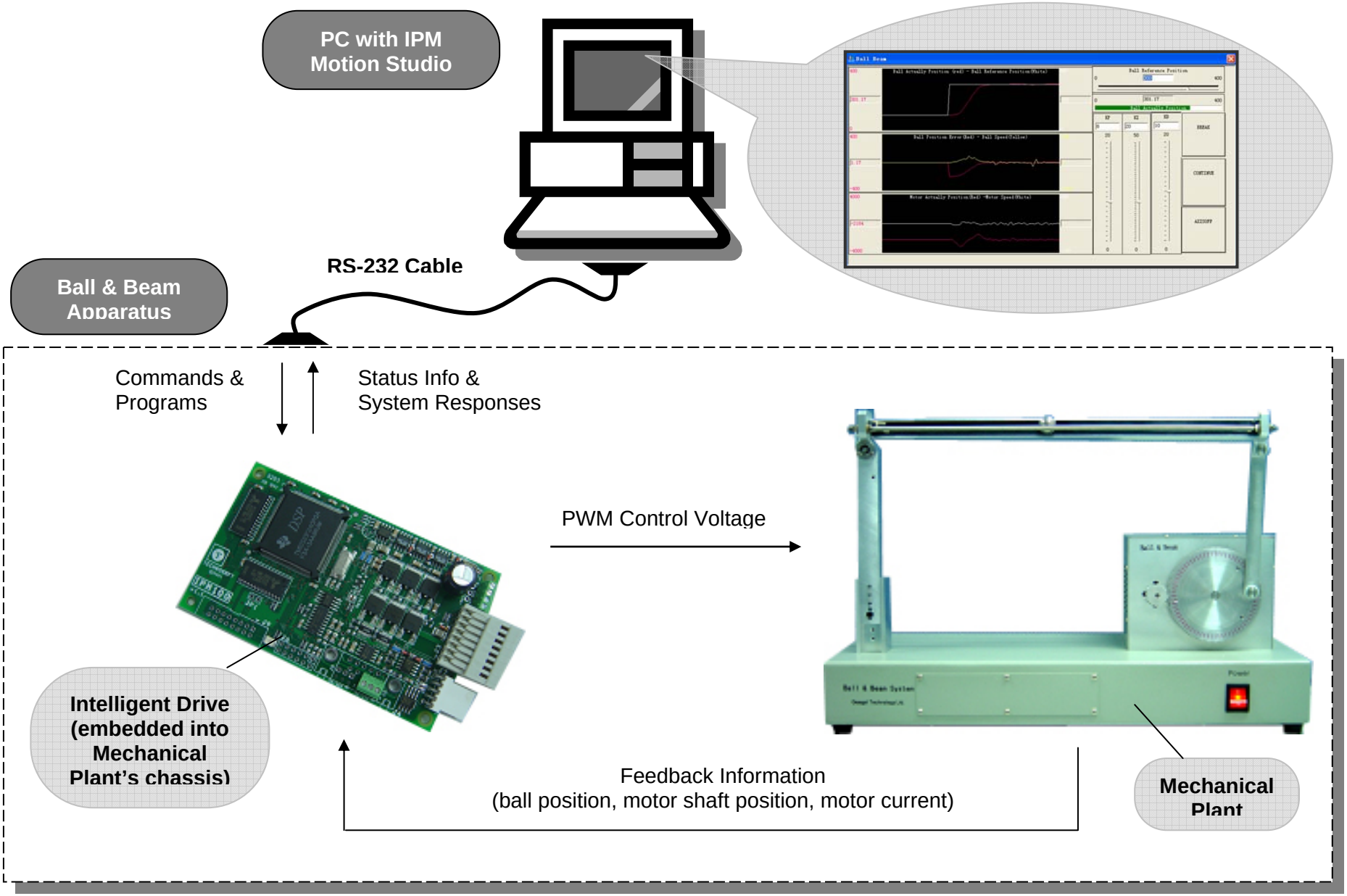



FIGURE 1-1 BASIC COMPONENTS AND THEIR INTERCONNECTION OF BALL&BEAM

- Ball & Beam apparatus that includes
  - ✓ mechanical plant with built-in DC servo motor and DC voltage supply,
  - ✓ IPM100 intelligent servo drive system,
- PC with the control program.

The details of functional blocks are discussed in the following sections.

### 1.3.1 Mechanical Plant

The mechanical plant consists of a base, a beam, a ball, a lever arm, a gear, a support block, a motor and an embedded electrical power supply as shown in Figure 1-2.

The ball can roll freely along the whole length of the beam. The beam is connected to the fixed support block at one side and to the movable lever arm at another one. In turn the motion of the lever arm is controlled by the DC brush motor through gear.

The motor is equipped with built-in rotary optical incremental encoder that provides feedback information about current actual rotary position of the motor shaft. In the slot along the beam there is a linear potentiometer sensor that senses current linear actual position of the ball on the beam. Both measured positions are fed back to the control system to organize a closed loop control.

As the servo gear turns by an angle theta, the lever arm changes the angle of the beam by alpha. When the beam is moved away from horizontal position, gravity causes the ball to roll along the beam.

The purpose is to design and implement a control algorithm that moves the shaft of the motor in such a way that the desired position of the ball is stabilized on the beam.

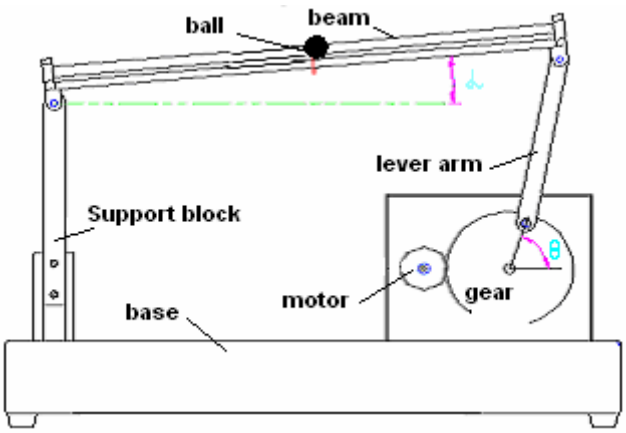



FIGURE 1-2 FUNCTIONAL COMPONENTS OF MECHANICAL PLANT

### 1.3.2 Intelligent Drive

The motion of the motor's shaft is governed by IPM100 intelligent drive. This is a high precision, fully digital servo drive with embedded intelligence and 100W power amplifier suitable for brushless/brush motors. Based on feedback information from sensors it computes and then applies appropriate PWM modulated voltage to the motor windings in such a way that a sufficient torque moves the motor shaft according the user programmed control algorithm.

The drive is called “intelligent” because besides built-in power amplifier for control signal amplification and PWM-modulation it has an on-board Digital Signal Processor (DSP), memory and other digital logic which offer advanced motion control and PLC functionality. Real-time trajectory generation, closed loop servo control, handling commands from host computer and processing I/O signals are executed on-board according to the stored programs and on-line commands from host PC. This embedded intelligence provides a true real-time control performance independent of any delays caused by PC’s non-real time Operating System.

Although the drive can operate both in stand-alone and slave modes, Ball&Beam application examples given in this manual make use of its stand-alone features.

Physically IPM100 is located inside the body of the mechanical plant and communicates with the upper-level PC through RS-232 interface. The DC voltage to the drive is provided by the DC power supply also embedded into mechanical plant chassis.

### 1.3.3 PC-based Control Software

The user programs the drive with the high-level Technosoft Motion Language (TML) in IPM Motion Studio running on host PC. The code and on-line commands are downloaded to the drive for execution through PC’s COM port.

IPM Motion Studio is an advanced and intuitive-based Windows Integrated Development Environment for the set-up and analysis of motion control applications with IPM drives.

With this platform it is easy to:

- ✓ Identify motor, sensor and load parameters
- ✓ Tune and adjust the servo-drive control loops
- ✓ Build flexible motion control algorithms/programs using TML
- ✓ Analyze and evaluate the behavior of the system

The user can define the motor commands to be applied to the motor. The Motion Wizard tool can help to generate all TML instructions in a graphical way without need to write any actual TML code. Specific selection and definition dialogs can be opened, viewed and edited depending on the command type to be generated.

The advanced graphics tools include Data Logger, Control Panel, Watches for TML parameters, registers and memory. They can be used to perform real-time analysis of the behavior of the motion system.

It is advisable for users to read “IPM100 Motion Studio User Manual” for more details on IPM operation before assuming any practical steps with Ball&Beam.

# CHAPTER 2 GETTING STARTED

## 2.1 The Ball & Beam Package

It is important to inspect the GBB1004 package immediately upon receiving it for any mechanical damage, though it has been factory verified. If any damage has been found, please DO NOT use the product and contact us immediately.

The package includes the following items:

- One piece of mechanical plant unit with integrated power supply and intelligent IPM100 servo drive
- Two pieces of steel balls
- One piece of power cable
- One piece of 9-pin cable
- One piece of CD with software and documentation

| | |
|---|---|
|  **Warning** | ***Machinery in motion can be dangerous! It is the responsibility of the user to design effective error handling and safety protection as part of the machine. Googol Technology shall not be liable or responsible for any incidental or consequential damages.*** |

## 2.2 Elements You Need

Before starting your installation, make sure the follow items are ready.

- Ball & Beam package
- A compatible IBM-PC AT with hard disk, CD-ROM drive, VGA or SVGA monitor, running Windows OS (95/98/NT/ME/2K/XP)
- Screw driver

## 2.3 Getting Started in Matlab Similink

Please follow the following steps to set-up Ball&Beam control Toolbox into MATLAB Simulink (MATLAB Version 6.5)

### 2.3.1 Step 1: Connecting Ball&Beam

**i.** Extract from the package and unwrap GBB1004 system and its accessories

**ii.** Connect the Ball&Beam apparatus to the COM1 port of the computer through 9-pin RS-232 serial cable

**iii.** Plug the power cable to the apparatus but don't power up the GBB1004 system

### 2.3.2 Step 2: Installing Ball&Beam Control Toolbox

This section explains the process of installing the Toolbox into MATLAB Simulink.

**i.** Insert the setup CD-ROM into CD-ROM drive. Browse it.

**ii.** Install Ball&Beam toolbox from CD-ROM " … \ControlSoftware\ MATLABToolboxSetup" folder. If you use Windows NT/2K/XP please make sure that you own administrator privileges.

**iii.** Run the Setup.exe program, as the following :

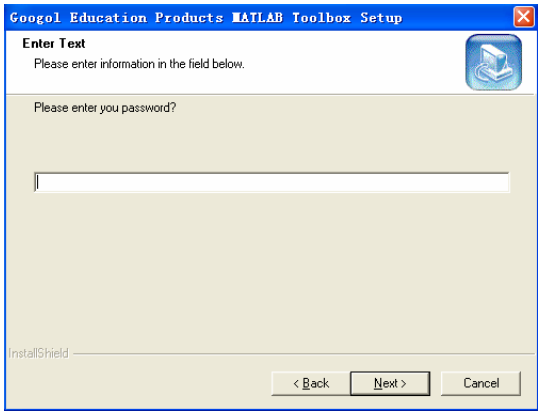



FIGURE 2 -1 SETUP BALL&BEAM CONTROLTOOLBOX INTO MATLAB

Enter the password into the dialog box to continue, you can found the SN number in the SN.txt.

**iv.** Following the steps to complete the installation.

**v.** Run MATLAB and check that the "Googol Educational Products" Toolbox has been appended.

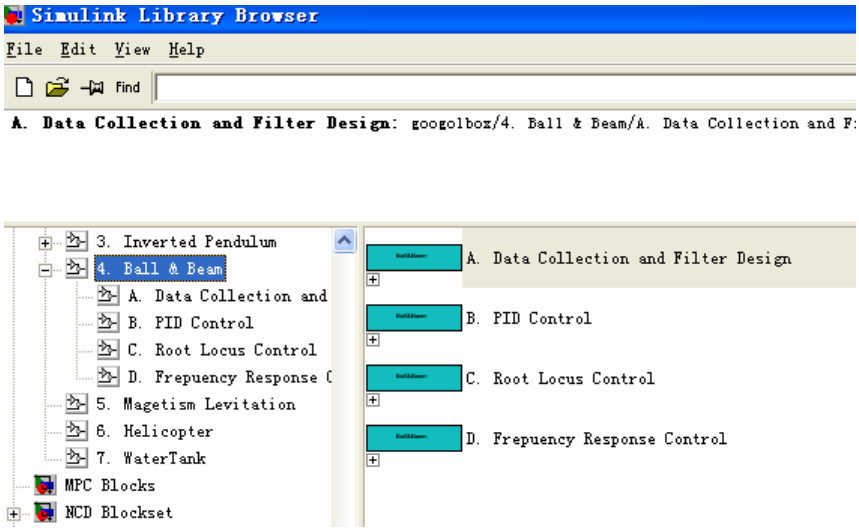



FIGURE 2 -2 GOOGOL EDUCATIONAL PRODUCTS TOOLBOX

**vi.** Double click on the "Ball&Beam Control \ PID Control" module and open the Ball&Beam control Demo:


FIGURE 2-3 INITIPMANDCONTROLCYCLE S-FUNCTION

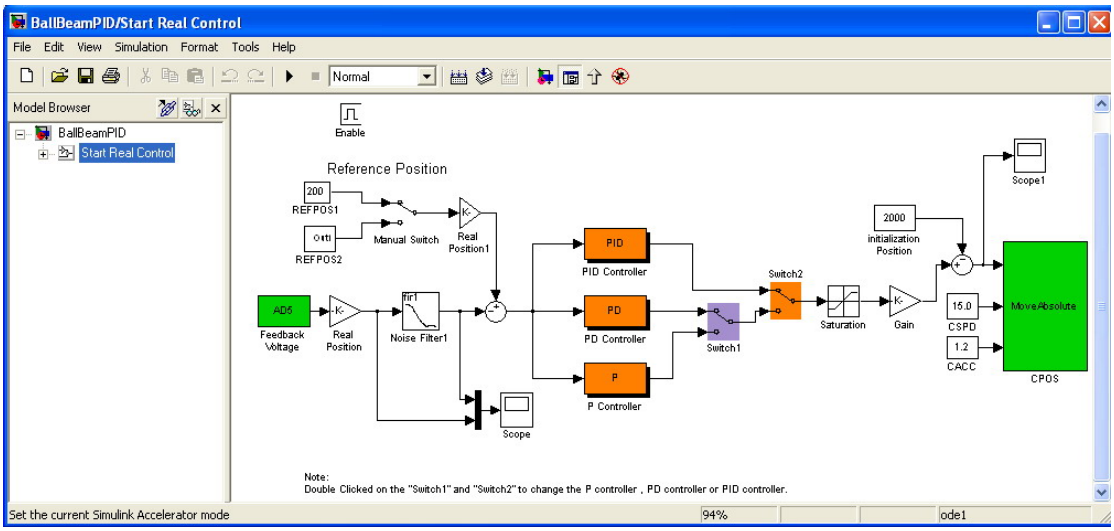



FIGURE 2-4 BALL&BEAM CONTROL DEMO

**vii.** Click on the Run icon "▶"to run the Demo.

### 2.3.3 Step 3: Set Parameters

***AD5*** module is a S-Function，it gets the ball actual position, as show in Figure 2-5.

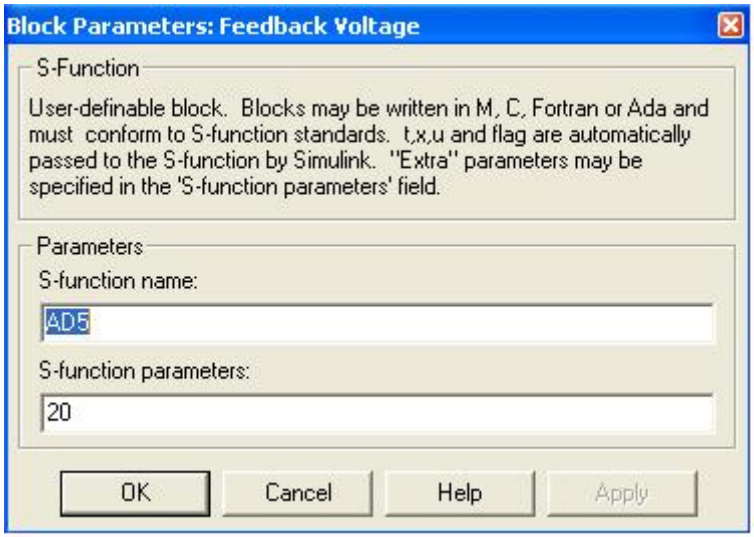



FIGURE 2 -3 AD5 S-FUNCTION

The actually position gained by ***AD5*** s-function may contains some electric disturbance and other noises, and usually there is some high frequency signs, these disturbances will decrease the controllability or the precision of the system, especially for the controllers that contains differentiator, a lowpass filter was designed to decrease the noises, it's convenient to design a filter by using matlab, seen as Figure 2-6 shows(there are hardware filters in the real system).

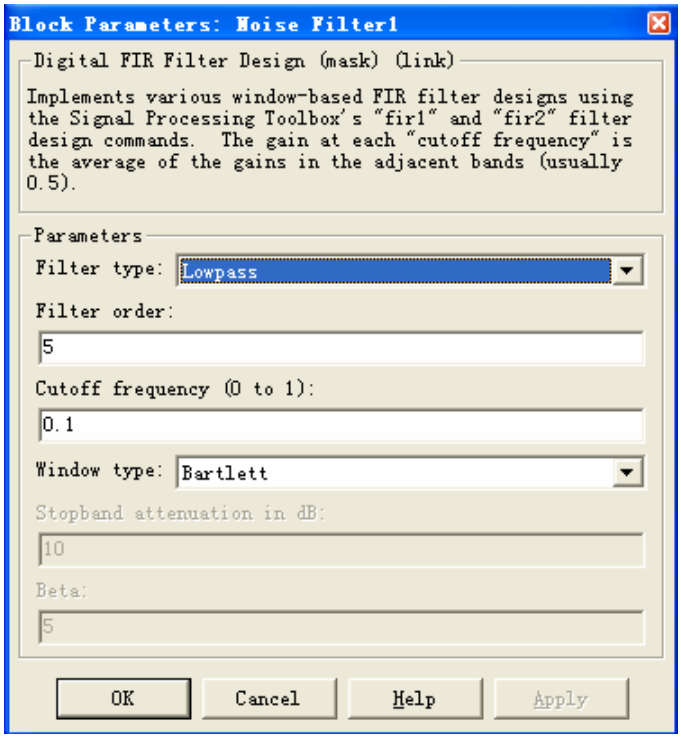



FIGURE 2 -4 FILTER DRSIGNED

The ball position after filtered compared with reference position, gain the ball position error, showed as Figure 2-7:

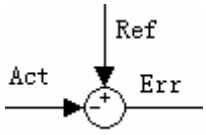



FIGURE 2-5 POSITION ERROR

The position was send to PID controller，PID controller is showed as Figure 2-8：

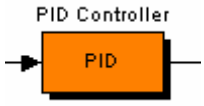



FIGURE 2-6 PID CONTROLLER

Click the mouse on the right, select "***Look under mask***"，open PID Controller structure, as Figure 2-9 shows：

The input of PID controller (In1）is the position error of the ball，the output of the controller（Out1）is the position of the motor, there Kp is proportional gain，Ki is integral gain, Kd is derivative gain，num(z) and den(z) are real control sample time.

Double click on the PID module, set the Kp、Ki and Kd parameters，as showed in Figure 2-10 ：

For there is a limit of the angle of the beam, it need add a upper and lower limit for the motion of the motor, double click on "Saturation" module，open the module as Figure 2-11 shows：

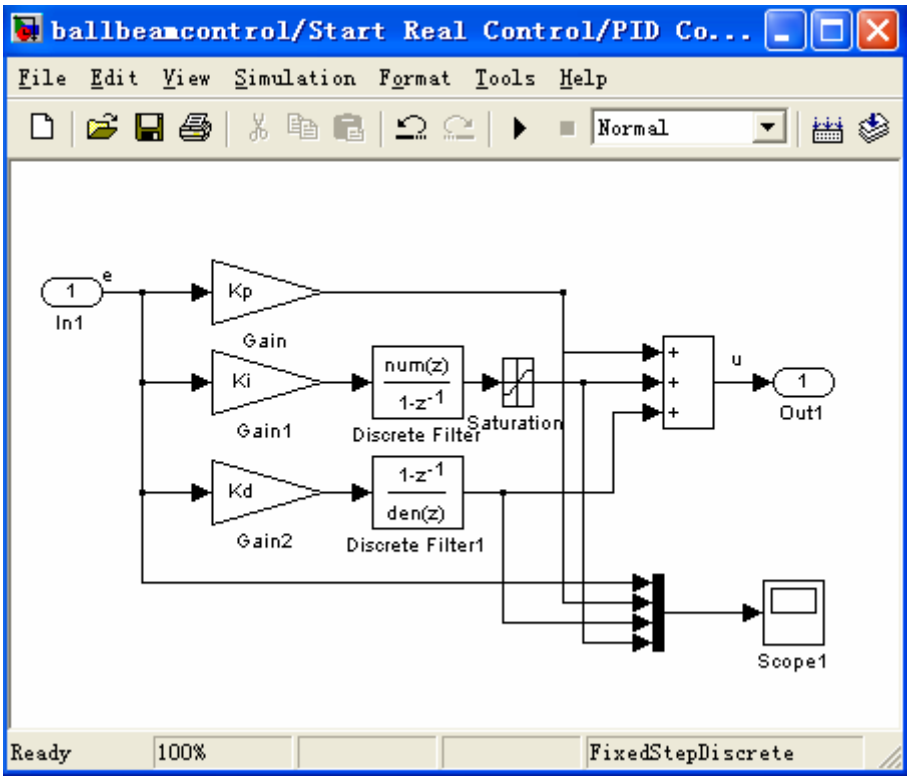



FIGURE 2-7 PID CONTROLLER STRUCTURE

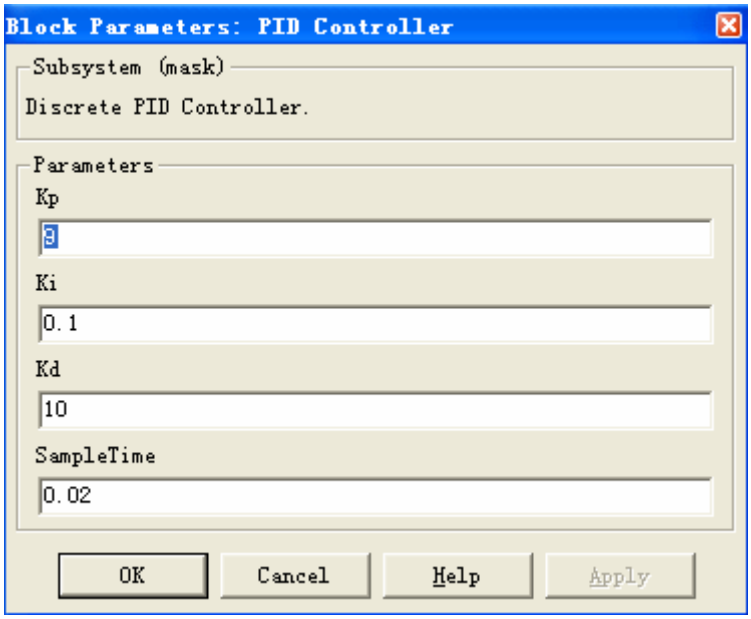



FIGURE 2-8 PID CONTROLLER PARAMETERS

For there is a gear between the motor and the beam, the direction of the motion is reverse, so the "***Gain***" is set "-1".

Then send the target position to the motor driver using a S-Function, drive the motor to the target position, double click on the "***SetMotorPos***" module as showed in Figure 2-13:

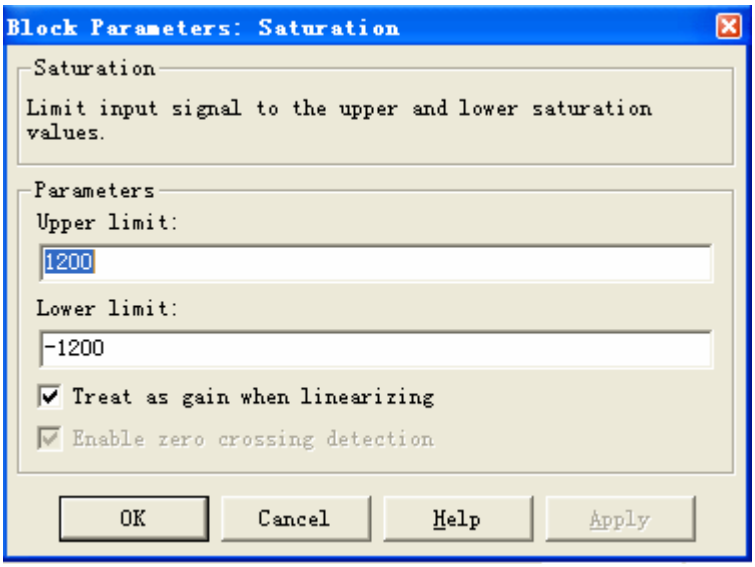



FIGURE 2-9 MOTOR POSITION SATURATION

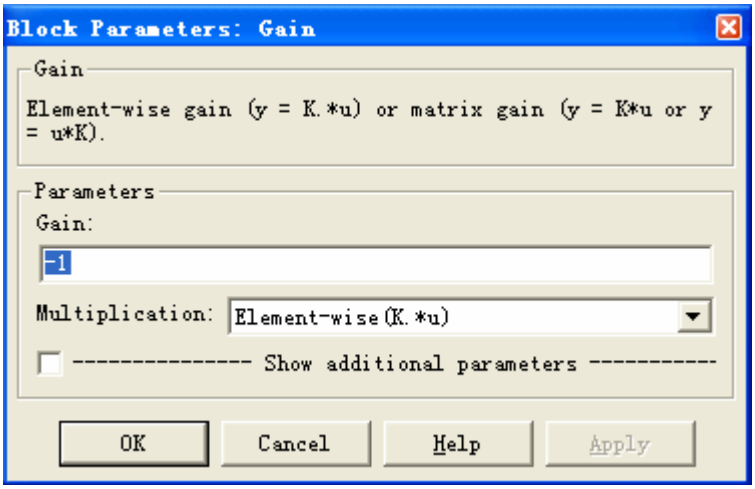



FIGURE 2 -10 MOTOR POSITION REVERSE

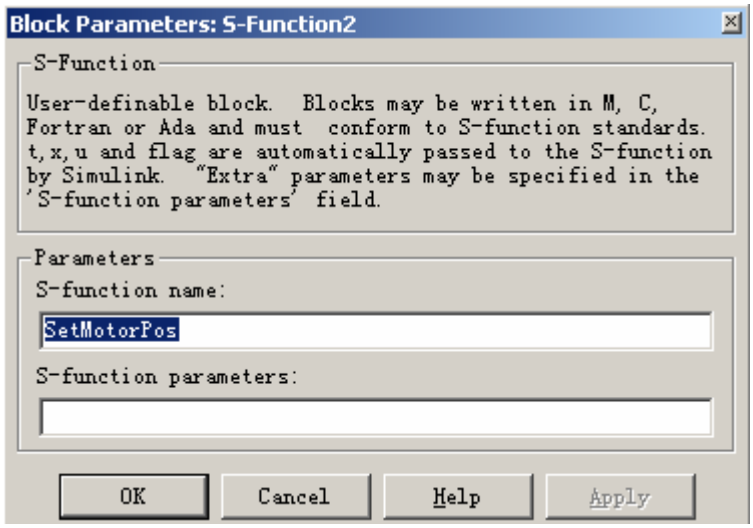



FIGURE 2 -11 SETMOTORPOS S-FUNCTION

Click on (start simulation) to run the control program, the experiment results can be shown by the scope, as depicted in Figure 2-14 and Figure 2-15.

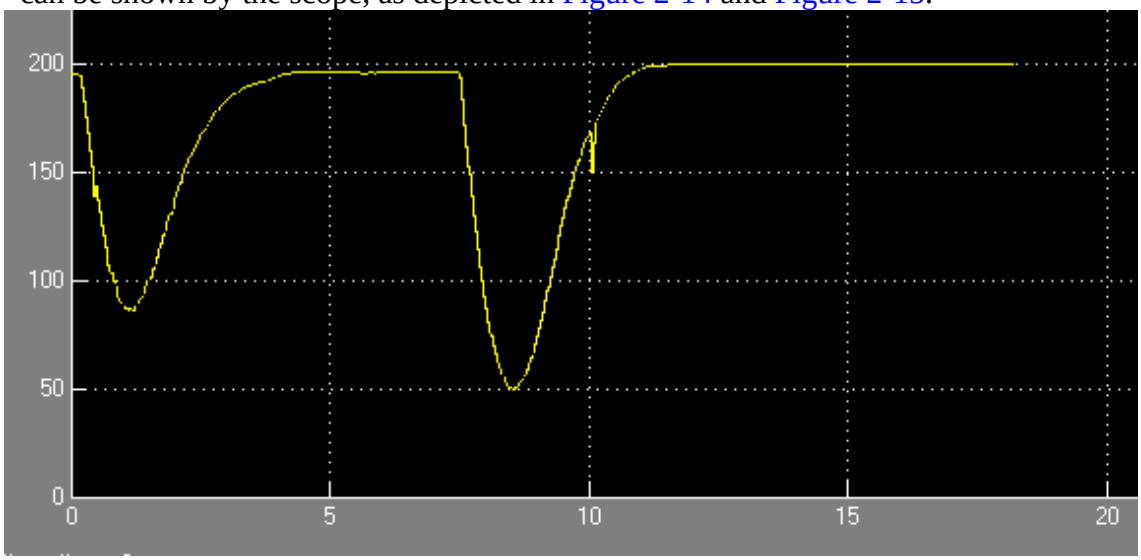



FIGURE 2 -12 EXPERIMENT RESULTS(BALL POSITION)

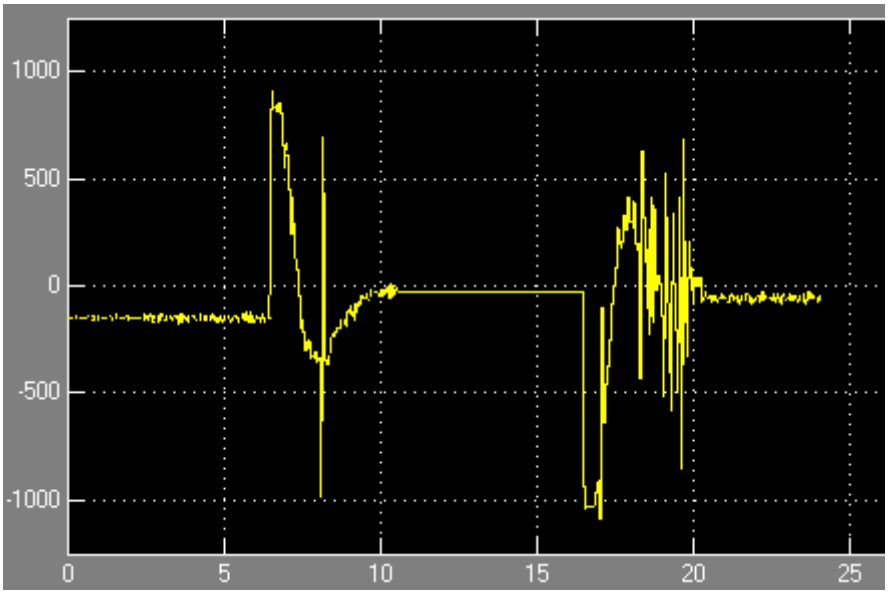



FIGURE 2 -13 EXPERIMENT RESULTS (MOTOR POSITION)

# CHAPTER 3 SYSTEM MODELING

## 3.1 Mechanical Model of Ball&Beam

The schematics of the Ball&Beam mechanical system is shown in Figure 3-1:

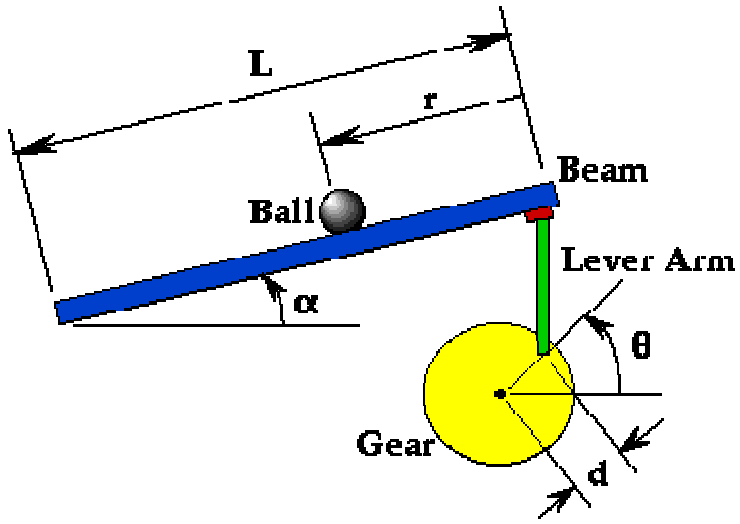



FIGURE 3-1 BALL&BEAM MECHNICAL SYSTEM

- Gear ratio is 4.28:1. (107:25)
- Let the angle between the line that connects the joint of the lever arm with the center of the gear, and the horizontal line be $\theta$ (there should be some boundaries on its range so that it can reach the safe maximum and minimum limits); the distance between the center of the gear and the joint of the lever arm be *d*, and the length of the beam be *L*. Then the beam angle $\alpha$ can be expressed in terms of the rotation angle of the gear $\theta$ according to the following equation:

$$\alpha = \frac{d}{L}\theta \qquad \text{(EQUATION 3-1)}$$

In turn, as it has just been noted above, the angle $\theta$ is connected with the rotational angle of motor shaft through reduction gear ratio $n=4.28$.

The controller design task is to keep the position of the ball $r$ equal to the specified target position by properly manipulating the gear angle $\theta$.

♦ The dynamics of the ball is subjected to the gravity, inertial and centrifugal forces. The ball linear acceleration along the beam is given by the following simple equation:

$$\left(\frac{J}{R^2}+m\right)\ddot{r}+mg\sin\alpha-m\dot{r}(\dot{\alpha})^2=0$$

Where $g$ is the gravitational acceleration.

M is the mass of the ball

J is the ball moment of inertia

r is the position of the ball along the beam

R is the radius of the ball.

Here we assume that the ball rolls without slipping and the friction between the beam and ball is negligible.

Since we are interested in keeping the angle alpha close to 0, we can linearize the above dynamic equation with respect to $\alpha$ in the neighborhood of zero and then taking into account the Equation 3-1 we get the following linear approximation of the system:

$$\ddot{r}=\frac{mg}{\left(\frac{J}{R^2}+m\right)}\alpha=-\frac{mgd}{L\left(\frac{J}{R^2}+m\right)}\theta \qquad \text{(EQUATION 3-2)}$$

This equation can be used to model the dynamic of the Ball&Beam system. The more detailed derivation of dynamical equations together with some additional computations can be found in the Internet.

## 3.2 Modeling the Ball and Beam Experiment in Simulink

In this example, rather than express all the forces and geometric constraints (which is difficult to model in Simulink for dynamic systems with constraints) we will model the nonlinear Lagrangian equation of motion directly. This equation gives d/dt(r) as a function of the state and input variab, r, d/dt(r), alpha, and d/dt(alpha). We will make use of the Nonlinear Function Block to express this function. First, we must express the derivatives of the output, r.

- Open a new model window in Simulink.
- Insert an Integrator block from the Linear block library.

- Insert a second Integrator to the right of the first, and connect the two with a line.
- Label the line connecting the two "d/dt(r)". To label a line, double-click near the line where you want the label (in this case, just below the line)
- Draw a line from the second Integrator and label it "r".
- Insert an Out block from the Connections block library and connect it to the "r" signal line. This will form the output of the system.
- Change the label of the Out block to "r" by single-clicking on the existing "Out" label.

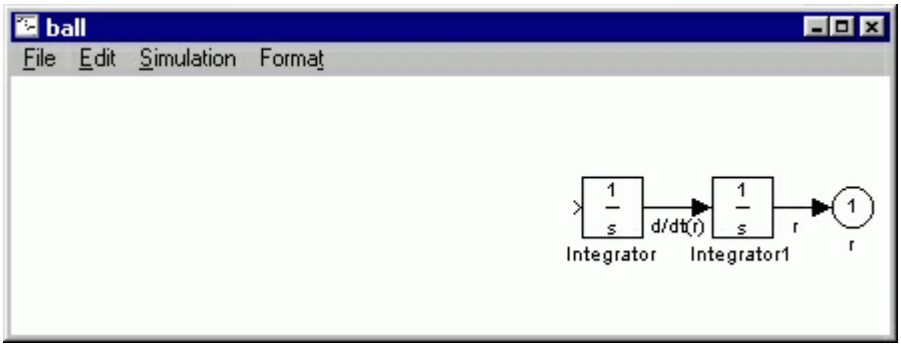



FIGURE 3-2 INTEGRATOR BLOCK

Now, we will insert the function which takes the vector [r d/dt(r) alpha d/dt(alpha)] and returns d/dt(r).

- Insert a Fcn block from the Nonlinear library and connect its output to the input of the first Integrator.
- Edit the Fcn block by double clicking it, and change it's function to the following:

```
(-1/ (J/(R^2) +m))*(m*g*sin (u [3])-m*u [1]*(u [4]) ^2)
```

This function block takes an input vector, u, where each component is referred to as u[1], u[2], etc. In our case, u[1]=r, u[2]=d/dt(r), u[3]=alpha, and u[4]= d/dt(alpha).

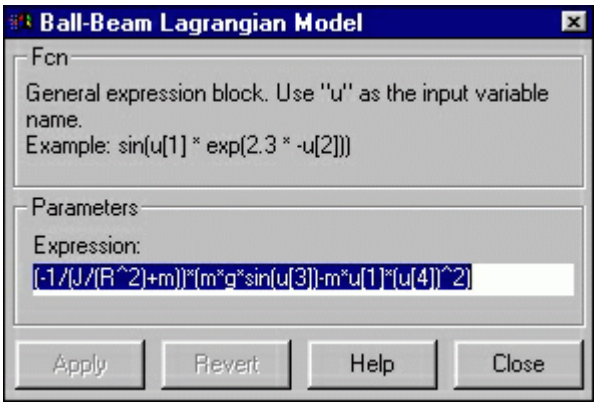



FIGURE 3-3 MODEL FUCITION

- Close the dialog box and change the label of the Fcn block to "Ball-Beam Lagrangian Model" (you can add newlines in the label by hitting return).

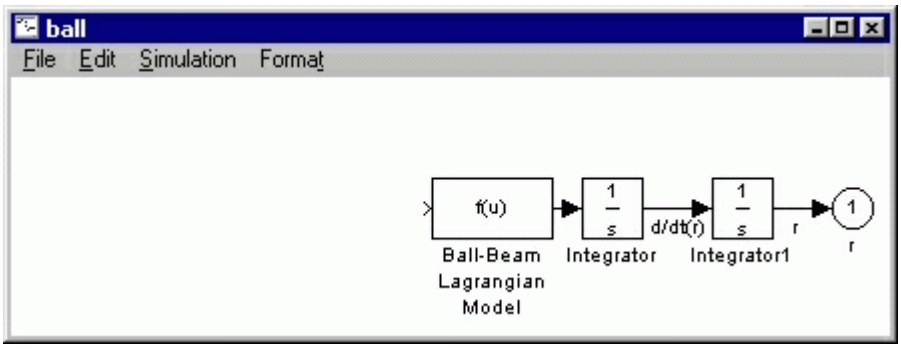



FIGURE 3-4 ADD MODEL TO THE SYSTEM

Now, we will begin to construct the function input vector u by feeding back the state signals from the integrators and forming a vector from them with a Mux block.

- Insert a Mux block from the Connections block library and connect its output to the input of the Ball-Beam block.
- Edit the Mux block (by double-clicking on it) and change its number of inputs to 4. The Mux block should now have four inputs.
- Tap a line off the d/dt(r) signal (hold Ctrl while drawing) and connect it to the second input of the Mux block.
- Tap a line of the r signal and connect it to the first input of the Mux block.

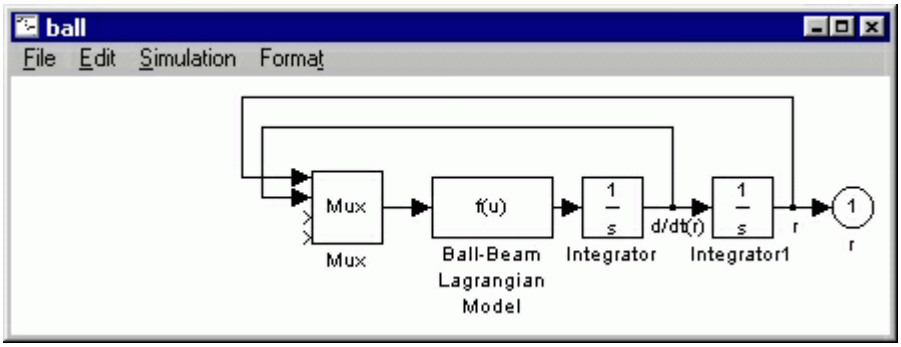



FIGURE 3-5 ADD MUX BLOCK TO THE SYSTEM

Now we will construct the signals alpha and d/dt (alpha) from the input theta.

- Insert an In block on the left side of your model window. Change its label to "theta".
- Insert a Gain block and connect it to the theta block. Change its gain value (double-click on it) to "d/L".
- Connect the output of the gain block to the third input of the Mux block. Label this line "alpha".
- Insert a Derivative block from the Linear block library and place it underneath the alpha signal line.
- Tap a line off the output of the Gain block and connect it to the input of the Derivative block.

- Connect the output of the Derivative block to the fourth input off the Mux block.

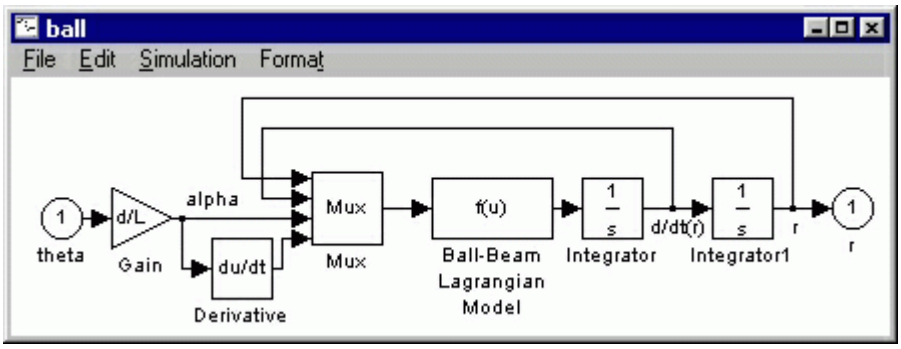



FIGURE 3-6 ADD THETA SIGNS TO THE SYSTEM

Save your model as "ball.mdl". Open Loop Response to generate the open-loop response, it is first necessary to contain this model in a subsystem block.

- Create a new model window (select New from the File menu in Simulink or hit Ctrl-N).
- Insert a Subsystem block from the Connections block library.
- Open the Subsystem block by double clicking on it. You will see a new model window labeled "Subsystem".
- Open your previous model window named ball.mdl. Select all of the model components by selecting Select All from the Edit menu (or hit Ctrl-A).
- Copy the model into the paste buffer by selecting Copy from the Edit menu (or hit Ctrl-C).
- Paste the model into the Subsystem window by selecting Paste from the Edit menu (or hit Ctrl-V) in the Subsystem window
- Close the Subsystem window. You will see the Subsystem block in the untitled window with one input terminal labeled theta and one output terminal labeled r.
- Resize the Subsystem block to make the labels visible by selecting it and dragging one of the corners.
- Label the Subsystem block "Ball and Beam Model".

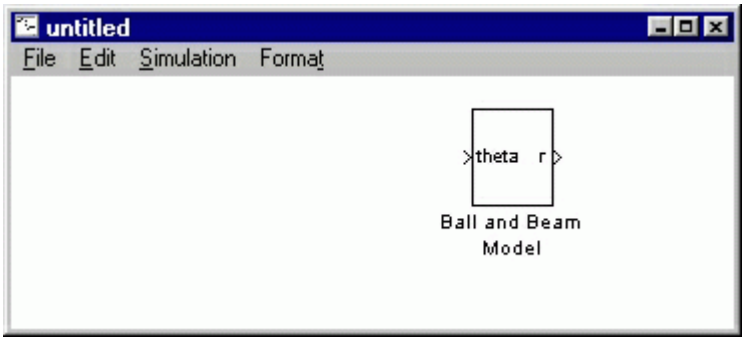



FIGURE 3-7 BUILD BALL AND BEAM MODEL

- Insert a Step block (from the Sources block library) and connect it to the input of the Ball and Beam Model.

- Edit the Step block (by double clicking on it to bring up the dialog box) and change the Step Time value to 0. Close the Step block dialog box.
- Insert a Scope block (from the Sinks block library) and connect it to the output of the Ball and Beam Model.

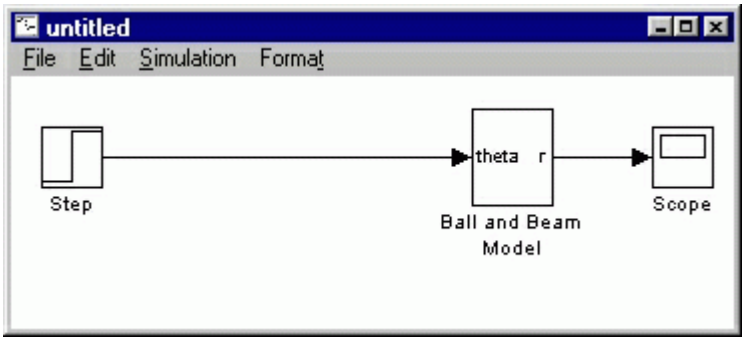



FIGURE 3-8 OPEN-LOOP SYSTEM OF BALL AND BEAM

Before obtaining a step response, we must set the physical parameters Enter the following commands at the MATLAB prompt.

```
m = 0.028;
R = 0.01;
g = -9.8;
L = 0.4;
d = 0.04;
J = 2*m*R^2/5;
```

We are now ready to run the simulation. Start the simulation by selecting Start from the Simulation menu (or hit Ctrl-t). When the simulation is finished, open the Scope by double clicking on it and hit the Scope's autoscale button. You will see the following response.

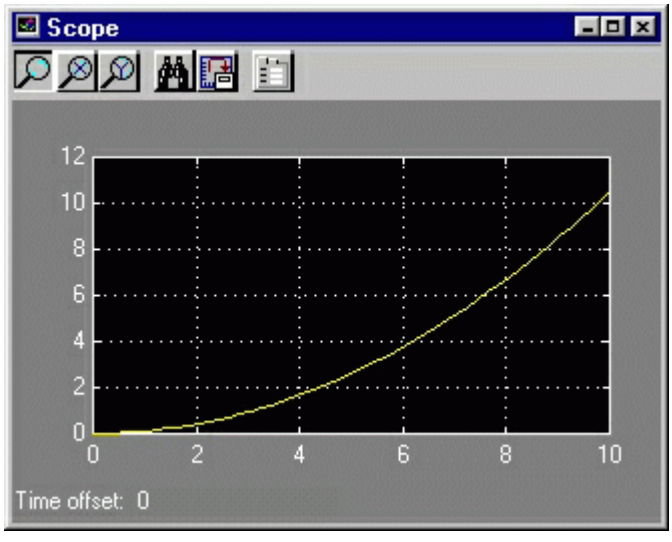



FIGURE 3-9 RESPONSE OF OPEN-LOOP SYSTEM

From this plot it is clear that the system is unstable in open-loop causing the ball to roll right off the end of the beam. Therefore, some method of controlling the ball's position in this system is required. Later in this tutorial, we will implement a lead compensator.

The Simulink model can be extracted into an equivalent state-space or transfer function model in MATLAB. This is done through the use of In and Out Connection blocks and the MATLAB function linmod.

To extract a model, it is necessary to start with a model file with inputs and outputs defined as In and Out blocks. Earlier in this tutorial this was done, and the file was saved as ball.mdl. In this model, one input, theta (the input crank angle) and one output, r (ball position), were defined.

At the MATLAB prompt, enter the following commands

```
[A,B,C,D]=linmod('ball')
[num,den]=ss2tf(A,B,C,D)
step(num,den);
```

You will see the following open-loop response:

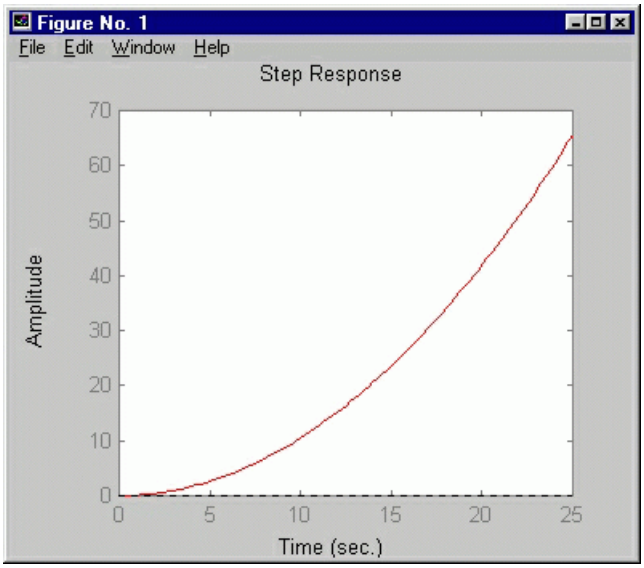



FIGURE 3-10 OPEN-LOOP RESPONSE

## 3.3 Electrical Model

- A built-in encoder of the DC motor senses the angle position of the motor shaft and, thus, the angle $\theta$ via transmission reduction ratio $n=7$. The encoder translates motion into electrical pulses, which are fed back into the IPM100 drive. The drive accepts feedback from incremental encoder, which provides two channels in quadrature, as well as a third channel (or index) for synchronization. Each quadrature channel consists of a square wave offset each other by 90 degrees. Channel A leading Channel B by 90 degrees reveals positive (or forward) motion, or vice versa. The controller decodes the signal into quadrature states, or 4 times the number of cycles, hence the resolution of the position is enhanced by 4 times. The index channel provides one transition low impulse per revolution. It is typically used for synchronization or precise positioning.
- Model of the motor

In this part, we will build the model of the servo motor; first, the parameters of the servo motor are measured:

Resistance: 9 ohms
Inductance: 0.2 mh

the torque of the motor is direct proportion to the current：

$$T = K_2 i_a \qquad \text{(EQUATION 3-3)}$$

where $K_2$ is torque constant of the motor.

$i_a$ is current of the armature.

When the armature is turning, the armature will induce a voltage, it's value is direct proportion to the magnetism and the speed of armature, when the magnetism is constant, the induced voltage $e_b$ is direct proportion to the $d\theta / dt$：

$$e_b = K_3 \frac{d\theta}{dt} \qquad \text{(EQUATION 3-4)}$$

where $e_b$ is induce voltage.

$K_3$ is constant of the motor's induced voltage

$\theta$ is angle of the axes.

The speed of the direct current servo motor is due to the voltage of the armature $e_a$, ($e_a = K_1 e_v$ is the output of the amplifier. The electric equation of the motor is：

$$L_a \frac{di_a}{dt} + R_a i_a + e_b = e_a \qquad \text{(EQUATION 3-5)}$$

or：

$$L_a \frac{di_a}{dt} + R_a i_a + K_3 \frac{d\theta}{dt} = K_1 e_v \qquad \text{(EQUATION 3-6)}$$

the torque balance equation is :

$$J_0 \frac{d^2\theta}{dt^2} + b_0 \frac{d\theta}{dt} = T = K_2 i_a \qquad \text{(EQUATION 3-7)}$$

where $J_0$ is the total torque of the system amounted to the motor.

$b_0$ is the friction of the system amounted to the motor.

So, the transfer function of the motor's position and the error sign is:

$$\frac{\Theta(s)}{E_v(s)} = \frac{K_1 K_2}{s(L_a s + R_a)(J_0 s + b_0) + K_2 K_3 s} \qquad \text{(EQUATION 3-8)}$$

(11)

Elements frame of the servo system is showed as following:

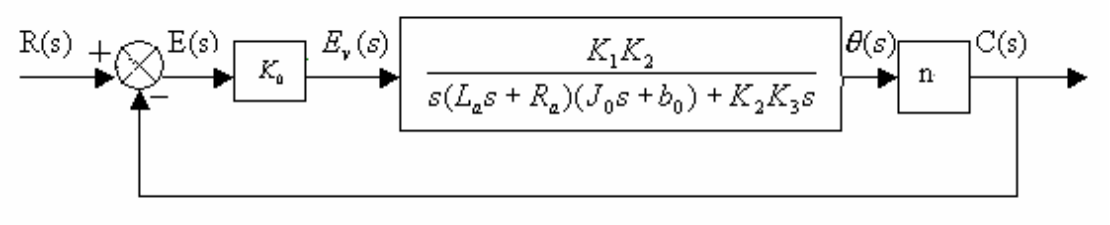



FIGURE 3-11 ELEMENTS FRAME OF THE SERVO SYSTEM

The ratio of the gear is n，then

$$C(s) = n\Theta(s)$$

The transfer function of the forward access is:

$$G(s) = \frac{C(s)\Theta(s)E_v(s)}{\Theta(s)E_v(s)E(s)} = \frac{K_0K_1K_2n}{s[(L_as + R_a)(J_0s + b_0) + K_2K_3]} \quad \text{(EQUATION 3-9)}$$

for $L_a$ is very little， predigest it as 13：

$$G(s) = \frac{K_0K_1K_2n}{s[R_a(J_0s + b_0) + K_2K_3]} = \frac{K_0K_1K_2n/R_a}{J_0s^2 + (b_0 + \frac{K_2K_3}{R_a})s} \quad \text{(EQUATION 3-10)}$$

EQUATION 3-11 can been simplified as EQUATION 3-12：

$$G(s) = \frac{K}{Js^2 + Bs} \quad \text{(EQUATION 3-13)}$$

or：

$$G(s) = \frac{K_m}{s(T_ms + 1)} \quad \text{(EQUATION 3-14)}$$

where $K_m = \frac{K}{B}$.

$T_m = \frac{J}{B} = \frac{R_aJ_0}{R_ab_0 + K_2K_3}$.

The transfer function contains a 1/s item, it has a character of integral, usually $R_a$、$T_m$ and $J_0$ is little, the servo motor can been considered as a integrator.

- The Ball&Beam system uses potentiometer mounted within a slot inside the beam to sense the linear position of the ball according to the Figure 3-12. The analog voltage proportional the linear position of the ball along the beam is fed to the A/D converter of IPM100 motion drive.

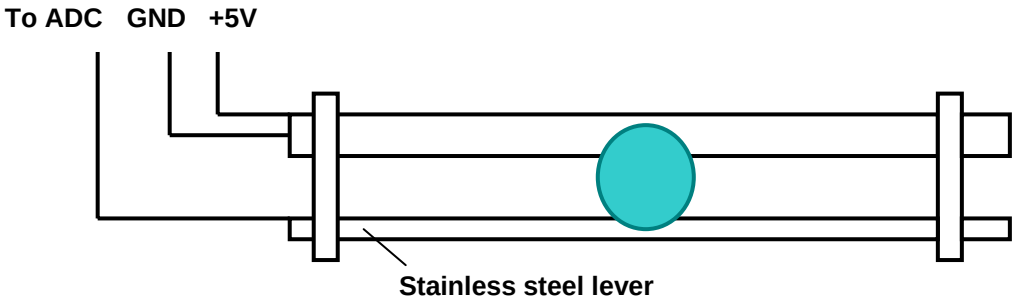



Figure 3-12 Measuring the linear position of the ball along the beam

## 3.4 Control Model

The closed-loop control algorithm employed in the demo application example is given in Figure 3-13:

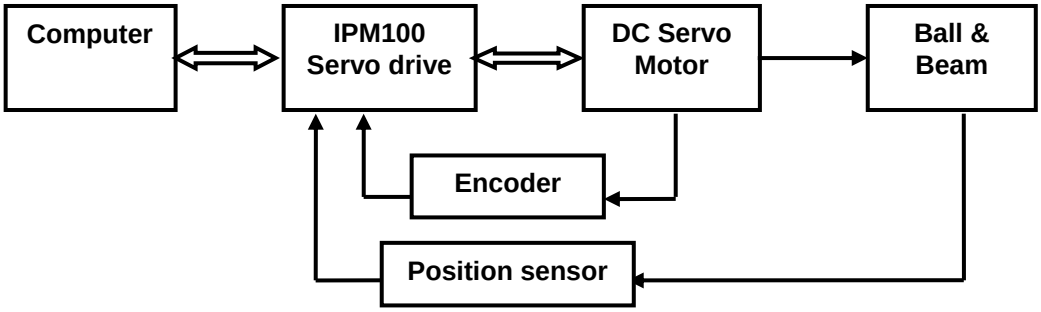



FIGURE 3-13 STRUCTURE OF THE CONTROL ALGORITHM

The DC motor provides actuation of the beam via a gear. The PID control algorithm inside IPM100 intelligent drive has been employed in an inner control loop as a motor position controller. The PID gains have been chosen in such a way that the motor exhibits a fast response without overshoot.

The drive realizes the control according to the following sequence:

- **i.** It downloads the control program written by the user via RS232 interface and stores it in the on-board internal memory
- **ii.** The feedback information is read from motor encoder and linear potentiometer of the beam every discrete sample time (the sample time is also programmable; 5 ms is default value for the demo program)
- **iii.** The on-board DSP decodes the downloaded program and computes the control signal according to the programmed algorithm based on feedback position information
- **iv.** The computed resulting control signal is amplified and PWM modulated by IPM's power stage
- **v.** The motor of the mechanical plant is powered in such a way that the rotation of its shaft balances the ball on the beam around the user-specified position.

The most general flowchart inside the control program is depicted in Figure 3-14. The example of its realization is given in appendix.

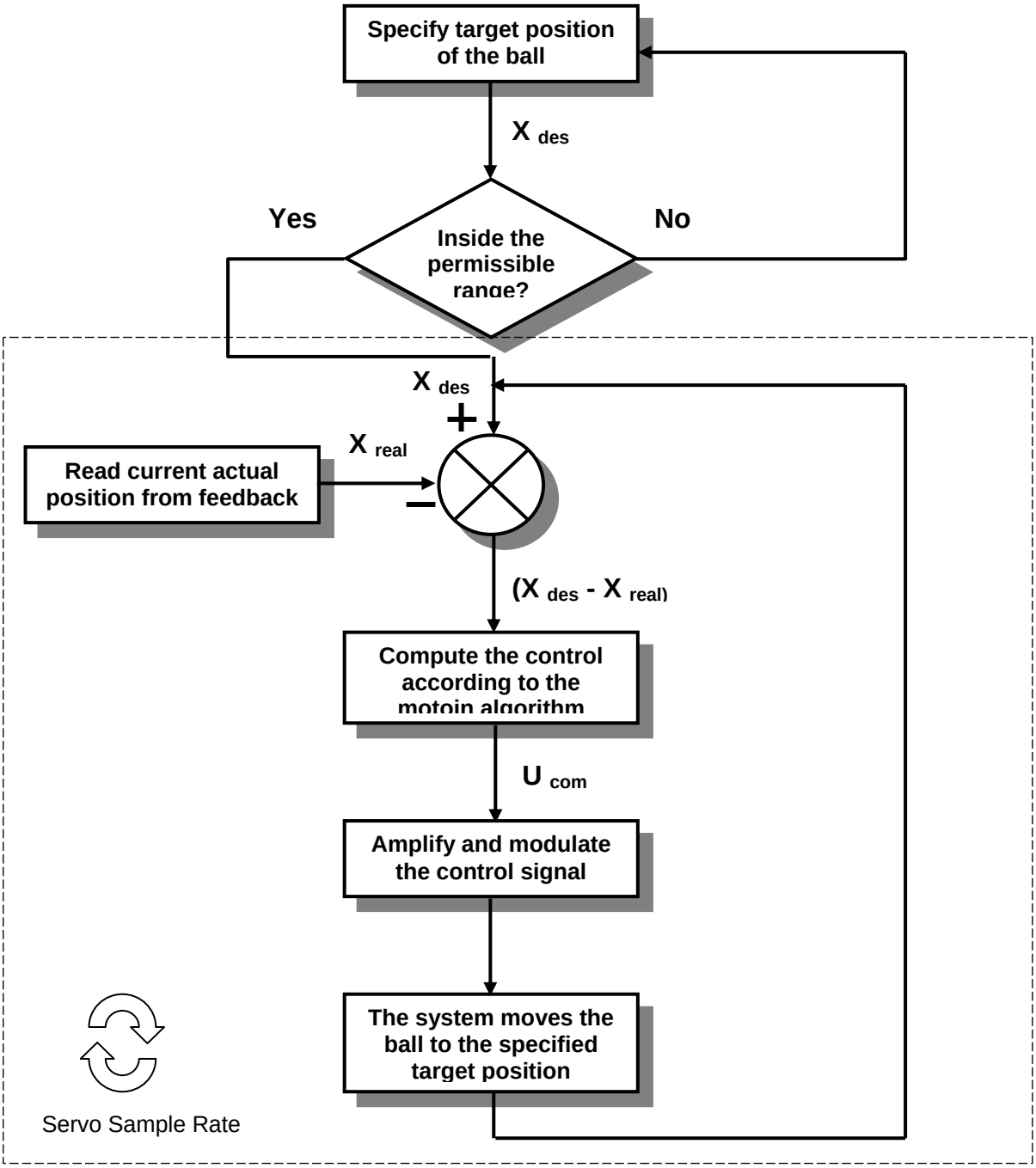



FIGURE 3-14 THE FLOW-CHART OF THE CONTROL PROGRAM

# CHAPTER 4 SAMPLE EXPERIMENTS

## 4.1 Data Collection and Processing

Sensor is the device that responds to a stimulus, such as heat, light, or pressure, and generates a signal that can be measured or interpreted. Sensors are widely used in the area of automatic detection and control, in which they convert the pressure, position or flux, etc to electrical signal. In this experiment, we are going to get the position data of the ball from the potentiometer by IPM Motion Studio and MATLAB Simulink respectively and give the reference on digital filter design techniques.

The position of the ball is detected by the output voltage of potentiometer, which is connected to the AD convert channel AD5 of IPM100. AD5 (16bit) range from 0 to 65535, the corresponding voltage range from 0-5V and the corresponding ball position range from 0-400mm.

**Data Collection by IPM Motion Studio.**

IPM Motion Studio is professional motion control development environment for IPM100 motion controller. The step by step implementation of IPM Motion Studio please refer to appendix A. More detailed please refer to the instruction P091.069.UM.1001.PDF.

The feedback voltage signal can be obtained as depicted in Appendix A. To get the actual ball position in the application program, please refer to the following steps:

**i.** Declare a real integer (32bit), eg: UserVar

**ii.** Give it a value: UserVar(L)=AD5

**iii.** Change UserVar to the real position of the ball: UserVar*400

Run program to observe the collected data. Further filter design is left as a assignment. Detail please refer to “Application Data Collection And Filter Design”

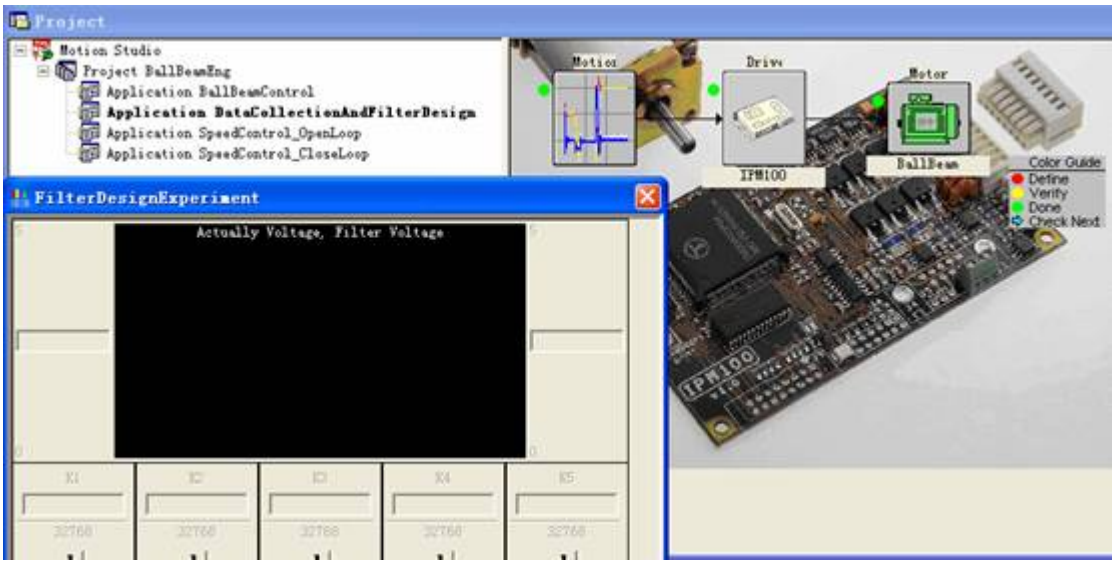



FIGURE 4-1 IPM MOTION STUDIO INTERFACE

## Data Collection in MATLAB Simulink.

Before experiment, please set the MATLAB path to the Ball&Beam directory, such as "C:\MATLAB6p5 \toolbox \googoltech\ball&beam":

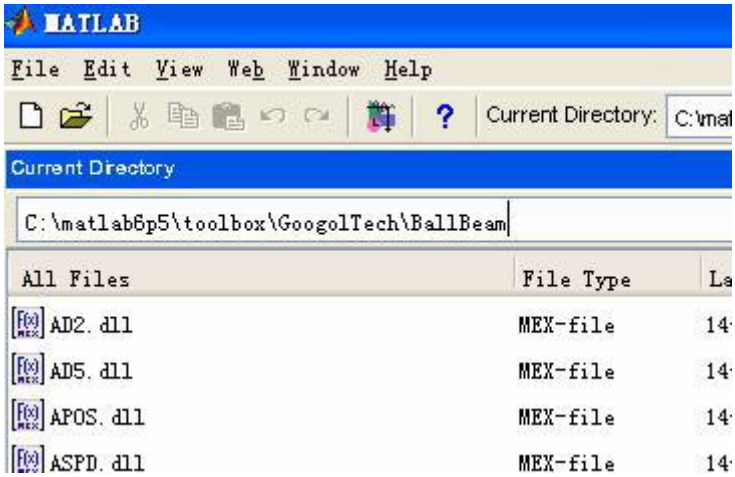



FIGURE 4-2 MATLAB PATH

The data collection and processing can be conveniently done by the powerful data processing toolbox of MATLAB.

Please follow the instructions:

**i.** Open "Googol Educational Products" toolbox in Simulink, and open " Ball&Beam \Control Demo\Ball&Beam Data Collection And Filter Design"

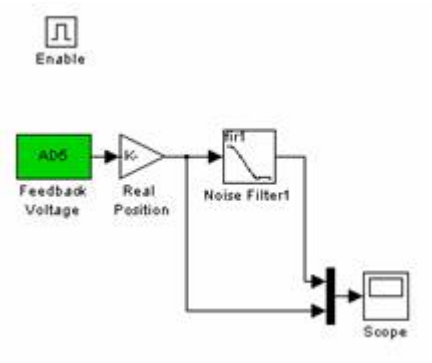



FIGURE 4-3 MATLAB DEMO

The functions of each part in Figure 4-3 are as follows:

"Feedback Voltage" module collects the value of AD5 channel in IPM100 controller and the "Real Position Gain" module convert the value to actual position of the ball (0-400mm). "Noise Filter1" is the filter that users can design according to their requirement. By observing the difference from the original and filtered signal from "Scope", users can make comparison to different filters. It is a training assignment to the user to implement filter design algorithms in MATLAB Simulink and run them using a real hardware.

**ii.** Run the control demo, make the ball rolling along the beam, the result look like the following should be obtained:

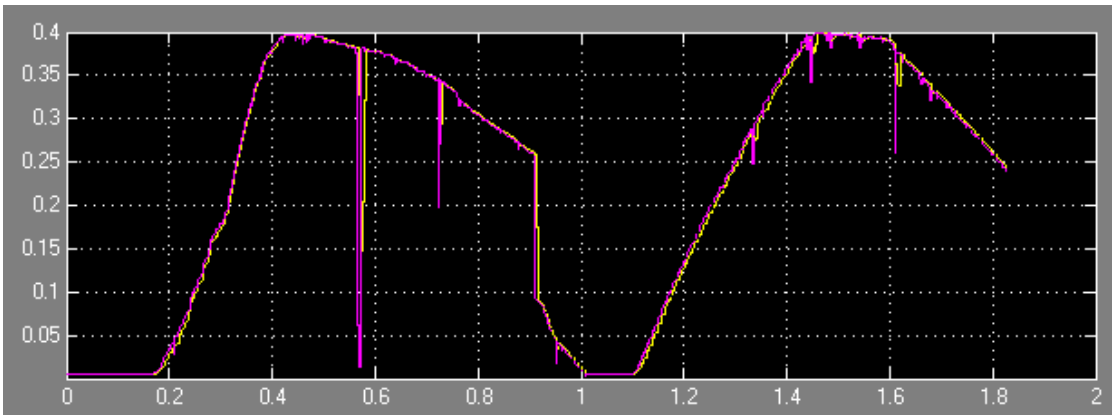



FIGURE 4-4 EXPERIMENT RESULT

This part could also refer to the MATLAB Simulink introduction in Chapter 2, which gives detailed parameter set and simple controller design.

## 4.2 Open-Loop Model of Ball&Beam System

Ball&Beam represents a Single Input Single Output (SISO) system. Based on Equation 3-2, the open-loop transfer function of Ball&Beam mechanics can be approximated by double integrator:


FIGURE 4-5 OPEN-LOOP SYSTEM

where

$$W(s)=\frac{X(s)}{\theta(s)}=\frac{mgd}{L(\frac{J}{R^2}+m)}\frac{1}{s^2}=c\frac{1}{s^2} \qquad \text{(EQUATION 4-1)}$$

is the Laplace representation of the open-loop transfer function,

$X(s)$ and $\theta(s)$ are the Laplace representation of the output (position of the ball on the beam) and input (beam angle) of Ball&Beam.

The open-loop step response of such system to a step input is given in Figure 4-6. From this plot it is clear that the system is unstable in open loop causing the ball to roll to one of the ends of the beam. Therefore, some closed loop method of controlling the ball's position in this system is required.

Let's define the design criteria to be met as following:

- Settling time less than 3 seconds
- Overshoot less than 5%

The sections below provide a general introduction into some of the controller design methods suitable for this problem:

- ✓ Proportional-Integral-Derivative (PID)
- ✓ Root Locus
- ✓ Frequency Response.

Although the IPM100 intelligent drive represents a digital (discrete) control system, without loss of generality the transfer functions in this chapter will be considered as for continuous case for the sake of simplicity of explanation. The same design principles are easily transferred to the discrete systems.

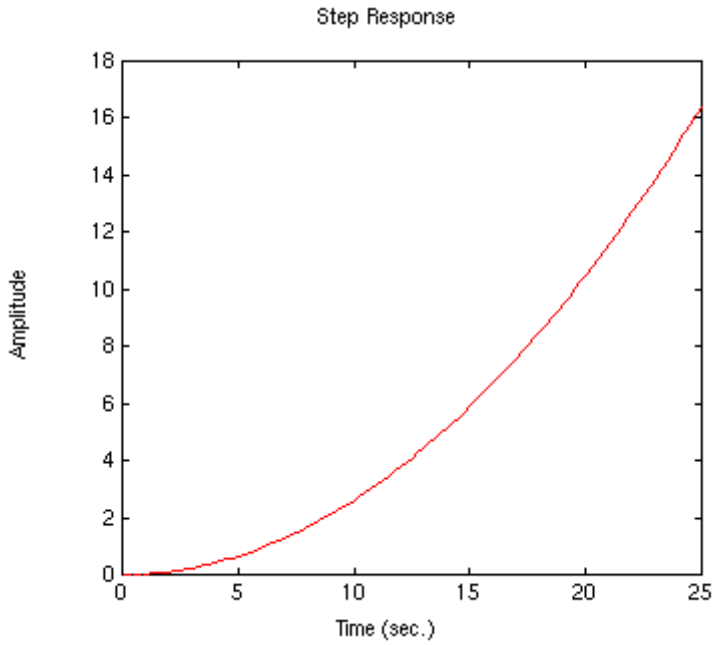



FIGURE 4-6 OPEN-LOOP STEP RESPONSE OF BALL&BEAM PLANT

The illustrative simulations facilitating the understanding of the material will be done in MATLAB environment.

## 4.3 Ball and Beam Control Using PID

In this version of the Ball & Beam experiment, we are going to use the PID control method to design the digital controller.

Recall that the transfer function for a general PID controller is:

$$K_P + \frac{K_I}{s} + K_D s = \frac{K_D s^2 + K_P s + K_I}{s},$$

where $K_P$, $K_I$ and $K_D$ are the proportional, integral and derivative gains respectively.

Let us consider some particular cases of PID control in greater details.

### 4.3.1 Proportional Control

First, we will study the response of the closed-loop system when a proportional controller is used. Then, derivative and/or integral control will be added if necessary.

**Theory**

The diagram for this example with a controller, Ball&Beam mechanics and a feedback of the ball position is shown below:

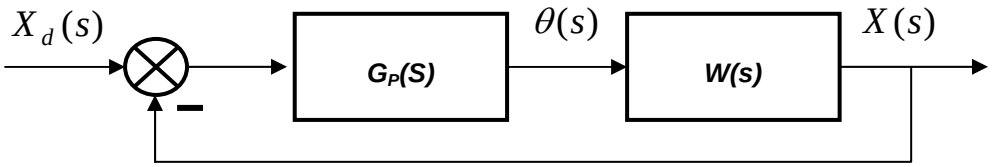



FIGURE 4-7 PROPORTIONAL CLOSED-LOOP CONTROL

Here $X_d(s)$ is the Laplace representation of the desired target position of the ball.

The equation of the proportional controller is:

$$G_P(s) = K_P$$

The transfer function of the closed loop system is:

$$\frac{X(s)}{X_d(s)} = \frac{G_P(s)W(s)}{1+G_P(s)W(s)} = \frac{cK_P}{s^2 + cK_P}$$

As it is clear from the above equation, this is a second-order system.

**Simulation**

The closed-loop transfer function for such control with a proportional gain $K_P$ equal to 100 can be simulated by the following lines of MATLAB code:

```
m = 0.028;
R = 0.01;
g = -9.8;
L = 0.4;
d = 0.04;
J = 2*m*R^2/5;
K = (m*g*d)/(L*(J/R^2+m));   %simplifies input
num = [-K];
den = [1 0 0];
plant=tf(num,den);
kp = 3;
sys_cl=feedback (kp*plant, 1);
```

In this piece of code the parameters `m, R, L, g, d` and `J` are assigned some numerical values and then the open-loop transfer function of the plant is simulated using MATLAB's built-in `tf` function. Finally the control loop is closed by `feedback` command and the result is assigned to `sys_cl` variable.

Now, we can model the system response to a step input of 0.25 *m* = 250 *mm*. Copy the above code into a new MATLAB **m-file**, add another line to simulate the specified step input

```
step(0.25*sys_cl)
```

and run the file in MATLAB environment.

The step response of your simulated system should look like the one shown in Figure 4-8.

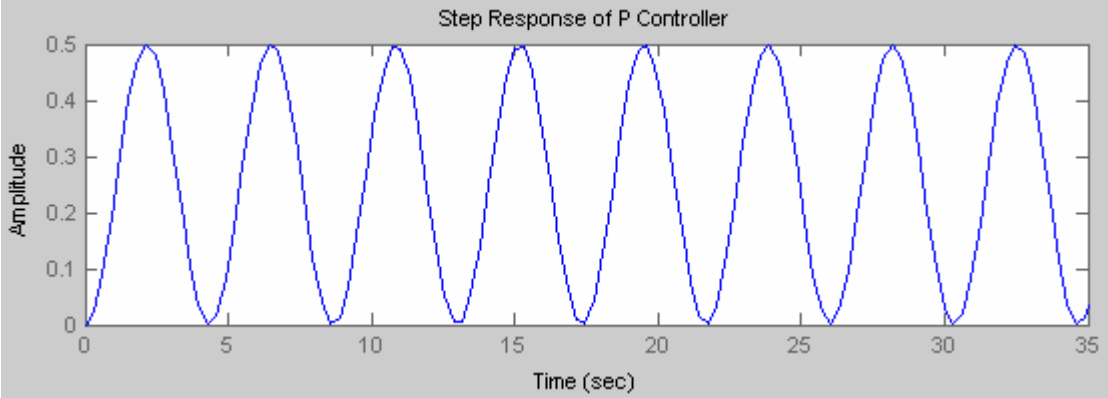



FIGURE 4-8 SIMULATION RESULT OF STEP RESPONSE UNDER P CONTROLLER

As you can see the addition of proportional controller does not make the closed-loop system stable. Try to change the value of *Kp* and observe that the system remains unstable. The inertial system, which is unstable, has equal breadth surging signals under the Proportional Control and the period is 5s.

## Experiment

**i.** Follow the steps described in Chapter 2 - Getting Started to run the provided demo program in MATLAB Simulink.

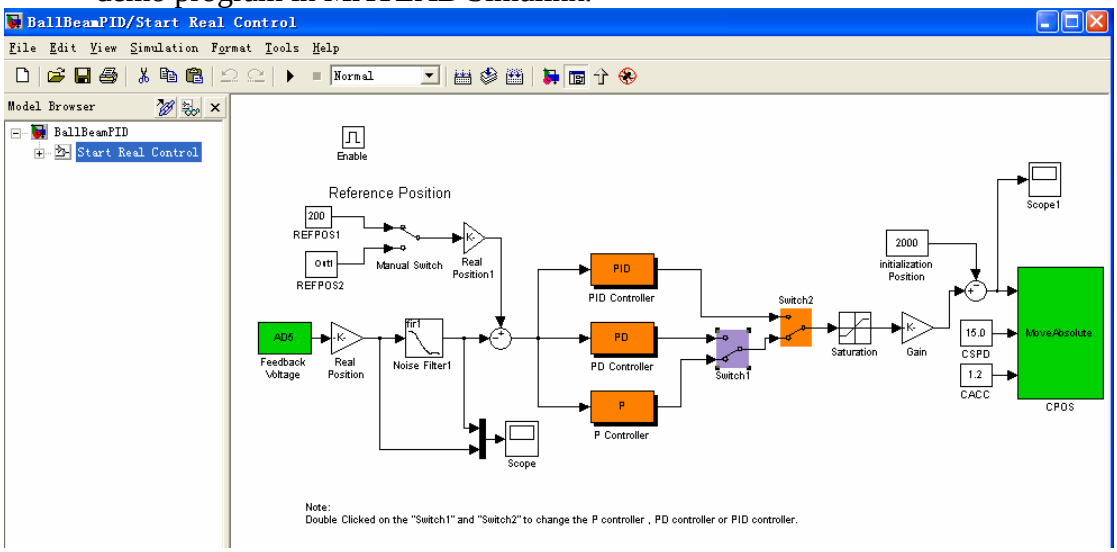



FIGURE 4-9 PID CONTORLLER MATLAB DEMO

**ii.** Switch the control mode to P controller.

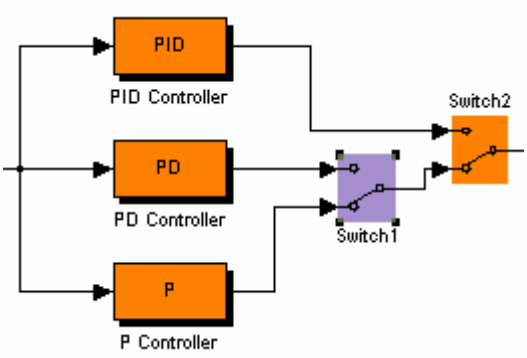



FIGURE 4-10 CONTROLLER MODE SWITCH

**iii.** Set the target reference position to 200 mm.

**iv.** Move the ball by fingers along the beam's slide to the actual position of 100 mm.

**v.** Release the ball. System will make attempts to stabilize the ball

**vi.** Change *Kp* and observe the response. The experiment result should look like the figure below. Compare the results with the simulation results in MATLAB and explain the difference.

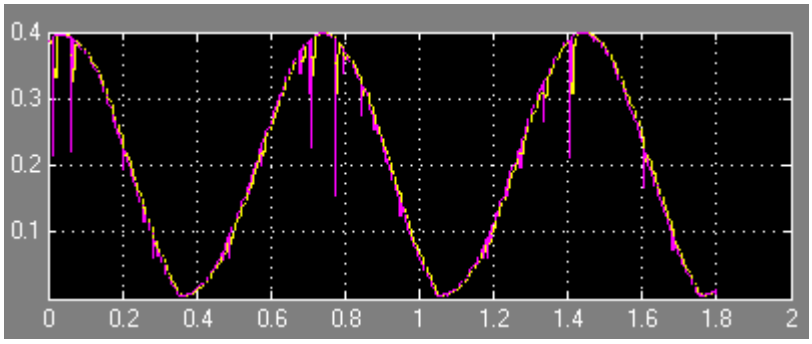



FIGURE 4-11 EXPERIMENTAL RESULTS OF STEP RESPONSE UNDER P CONTROLLER

### 4.3.2 Proportional-Derivative Control

**Theory**

Now, lets add a derivative term to the controller. The block diagram of such closed loop system is shown:

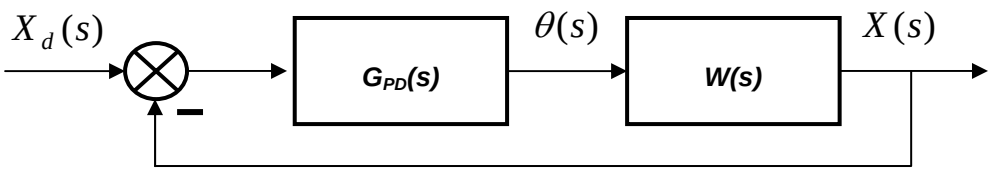



FIGURE 4-12 PROPORTIONAL-DERIVATIVE CLOSED-LOOP CONTROL

Here the transfer function of the PD controller is represented as

$$G_{PD}(s) = 1 + K_D s$$

what implies that the proportional gain $K_P$ is set to 1 for simplicity of analysis, and the derivative gain is $K_D$.

The closed loop transfer function of the whole system is:

$$\frac{X(s)}{X_d(s)} = \frac{G_{PD}(s)W(s)}{1 + G_{PD}(s)W(s)} = \frac{c(1 + K_D s)}{s^2 + K_D s + c}$$

## Simulation

Insert the following program to MATLAB **m-file** and run it to view the system step response for this type of control method.

```
m = 0.028;
R = 0.01;
g = -9.8;
L = 0.4;
d = 0.04;
J = 2*m*R^2/5;
K = (m*g*d)/(L*(J/R^2+m));   %simplifies input
num = [-K];
den = [1 0 0];
plant=tf(num,den);
kp = 6;
kd = 6;
contr=tf([kd kp],1);
sys_cl=feedback(contr*plant,1);
t=0:0.01:5;
step(0.25*sys_cl)
```

The simulation result is demonstrated in Figure 4-13.

Now the system is stable but both the overshoot and settling time are too high. The inertial system has minus breadth surging signals under the PD controller.

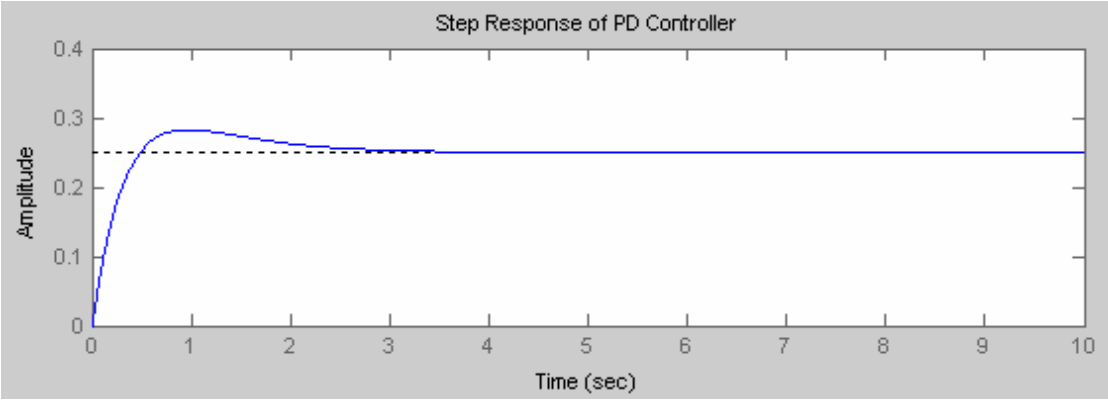



FIGURE 4-13 SIMULATION RESULT OF STEP RESPONSE UNDER PD CONTROLLER

**Experiment**

**i.** Follow the steps described in Chapter 2 - Getting Started to run the provided demo program in MATLAB Simulink.

**ii.** Switch to PD controller and set parameters as following:.

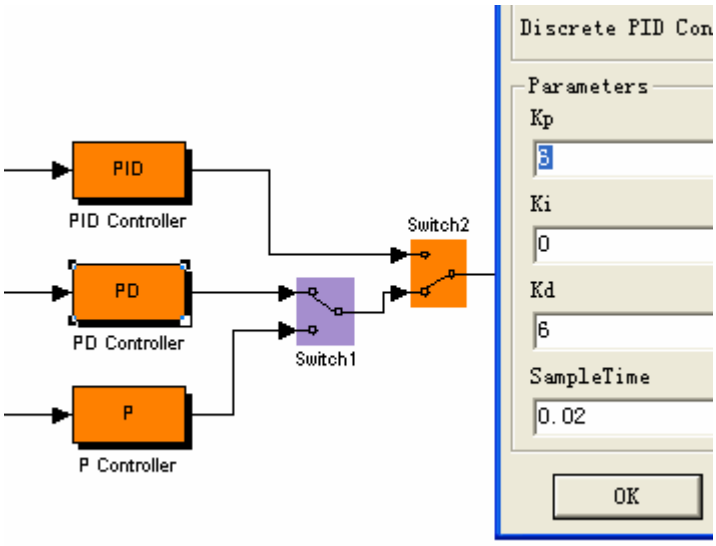



FIGURE 4-14 CONTROLLER MODE AND PARAMETERS

**iii.** Set the target reference position to 200 mm.

**iv.** Move the ball by fingers along the beam's slide to the actual position of 50.

**v.** Release the ball. System will make attempts to stabilize the ball

**vi.** Change $Kp$ and $K_D$ to observe the response.

The system stabilizes quickly under the PD controller though the steady state errors are big as in Figure 4-15. Analyze the relationship between the change of the position and the input angle. Compare the simulation and experiment result and explain the difference.

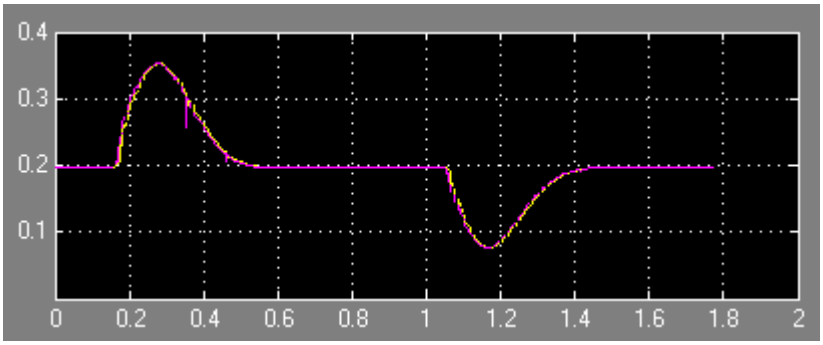



FIGURE 4-15 EXPERIMENTAL RESULTS OF STEP RESPONSE UNDER PD CONTROLLER

### 4.3.3 Analysis of PID control

**Theory**

The next experiment is to study the behavior of a full PID controller. The diagram of the closed loops system is depicted in Figure 4-1616:

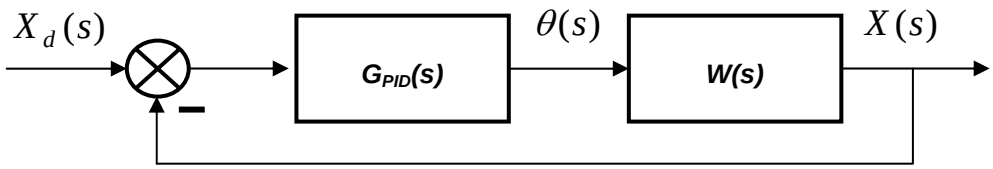



FIGURE 4-16 PROPORTIONAL-DERIVATIVE-INTEGRAL CLOSED-LOOP CONTROL

Here the transfer function of the PID controller is:

$$G_{PID}(s) = K_P(1 + K_D s + \frac{K_I}{s}),$$

with $K_D$ and $K_I$ being the corresponding I and D gains, and $K_P$ is a common proportional amplification gain.

The transfer function of the closed-loop system is described by the following equation:

$$\frac{X(s)}{X_d(s)} = \frac{cK_P(K_D s^2 + s + T_I)}{s^3 + cK_P(K_D s^2 + s + T_I)}$$

**Simulation**

Write MATLAB simulation program with the following parameters $K_P$ =10, $K_d$ =10, $K_I$ =10:

```
kp3 = 10;         % PID Controller
kd3 = 10;
Ki=1;
contrPID=tf([kd3 kp3 Ki],[1 0]);
sys_cl_PID=feedback(contrPID*ball,1);
t=0:0.01:10;
SUBPLOT(3,1,3)

step(0.25*sys_cl_PID,t)
title('Step Response of PID Controller')
```

From the simulation results you can see that increasing $K_D$ can lower the overshoot and decrease slightly the settling. Therefore, select $K_D$ =20 and your step response should be like the one shown in Figure 4-17.

The overshoot criterion is met but the settling time needs to be cut. To decrease the settling time at the expense of a slight increase of the rise time, .we may try to set $K_P$ to a bit higher value.

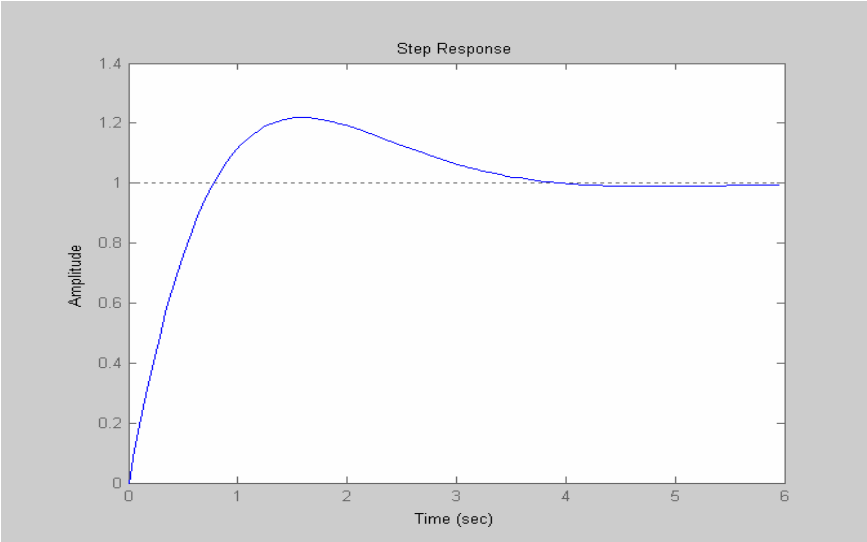



FIGURE 4-17 RESULT OF STEP RESPONSE UNDER PID CONTROLLER WITH INCREASED D GAIN

The derivative gain $K_D$ can also be made larger to take off some of the overshoot that the increased $K_P$ will cause. After playing with values of the gains for some time, the step response as in Figure 4-18 can be achieved with $K_P$ = 15 and $K_D$ = 40. Such system is considered as properly tuned.

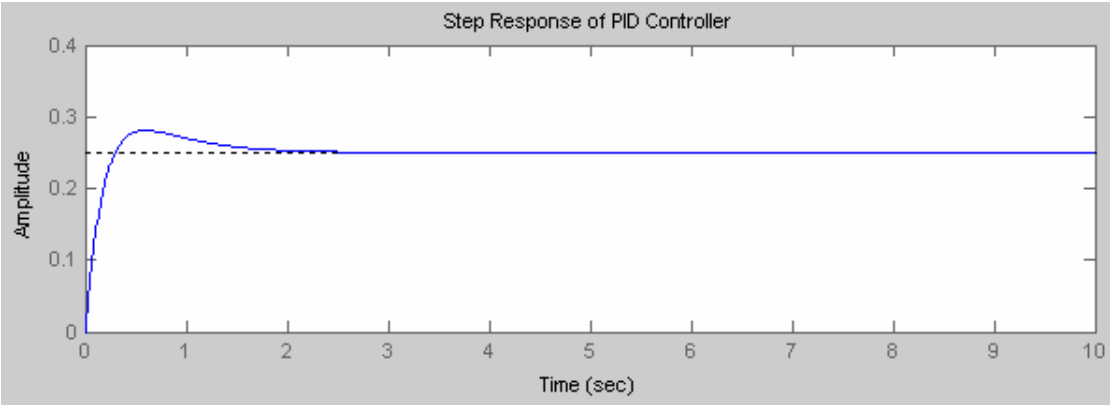



FIGURE 4-28 SIMULATION RESULT OF STEP RESPONSE UNDER TURNED PID CONTROLLER

**Experiment**

i. Follow the steps in previous section, do experiment with Ball&Beam system according to the instructions given in the above examples. Selecting the PID gains as $K_P$ =10, $K_I$ =1, $K_D$ =10, the result is shown below:

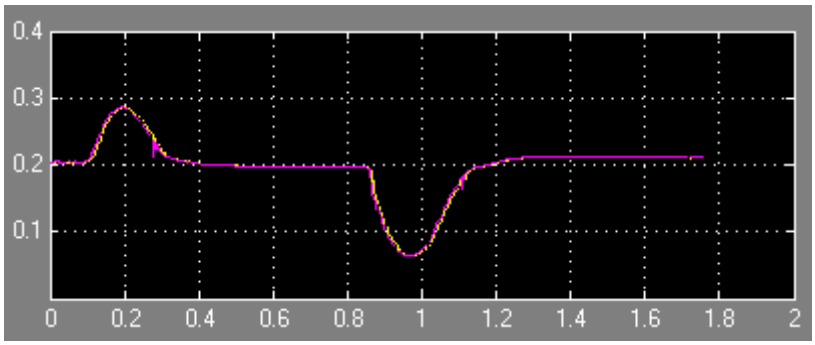



FIGURE 4-39 EXPERIMENTAL RESULTS OF STEP RESPONSE UNDER PID CONTROLLER (I)

ii. Change controller parameters, set $K_P$ = 15 and $K_D$ = 10, $K_I$ =0.5, the following result will be obtained:

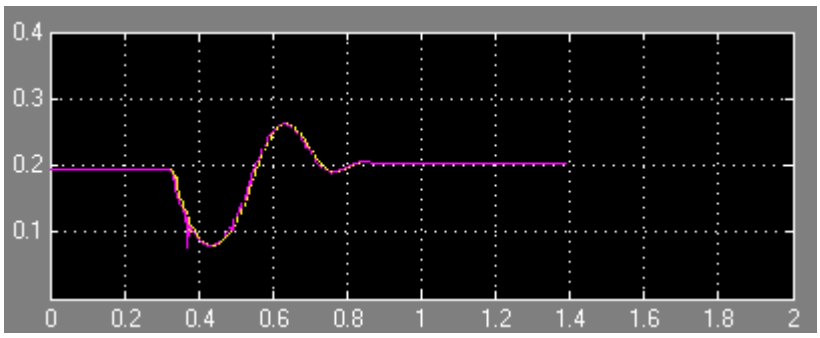



FIGURE 4-20 EXPERIMENTAL RESULTS OF STEP RESPONSE UNDER PID CONTROLLER (II)

The steady state error has been clearly lowered. As you can see all of the design requirements are satisfied. For this particular control problem, no implementation of an integral control was needed. But remember there is more than one solution for a control problem. You may try different parameters to obtain a satisfactory response in practice.

## 4.4 Root Locus method

The main idea of the root locus design is to estimate the closed-loop response from the open-loop root locus plot. By adding zeroes and/or poles to the original system (adding a compensator), the root locus and thus the closed-loop response will be modified.

A typical schematic of the closed loop system with a controller is given below:

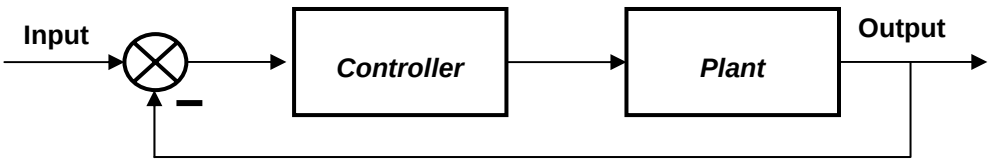



FIGURE 4-21 A TYPICAL CLOSED-LOOP CONTROL SYSTEM

Let's first give the design criteria for this problem:

- Settling time less than 3 seconds

- Overshoot less than 5%

Create an **m-file** with the following MATLAB code in order to simulate the plant and plot its root locus.

```
m = 0.028;
R = 0.01;
g = -9.8;
L = 0.4;
d = 0.04;
J = 2*m*R^2/5
K = (m*g*d)/(L*(J/R^2+m));   %simplifies input
num = [-K];
den = [1 0 0];
plant=tf(num,den);
rlocus(plant)
```

Run the m-file in MATLAB environment. You should see the system has two poles at the origin, which go off to infinity along the imaginary axes:

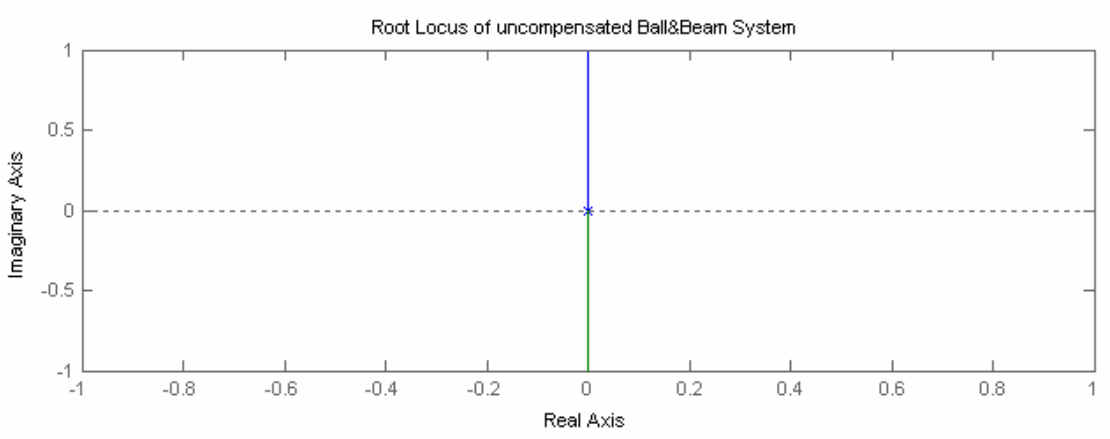



FIGURE 4-22 ROOT LOCUS OF THE PLANT

From the design criteria, we will get:

$$e^{-\varsigma\pi/\sqrt{1-\varsigma^2}} = 5\%$$

$$\frac{4}{\varsigma\omega_n} = 3$$

Now let us design and add the controller to the plant and view the resulting root locus. We will position the zero near the origin to cancel out one of the poles. The pole of our compensator will be placed to the left of the origin to pull the root locus further into the left-hand plane. Add the following lines of MATLAB code to your m-file:

```
zo = 0.01;
po = 4;
contr=tf([1 zo],[1 po]);
rlocus(contr*plant)
sgrid(0.70, 1.9)
```

Run your m-file in the MATLAB command window and you should see the plot as demonstrated in Figure 4-23.

As you can see the branches of the root locus are within our design criteria.

Now after we moved the root locus into the left-hand plane, we can select a gain that would satisfy our design requirements.

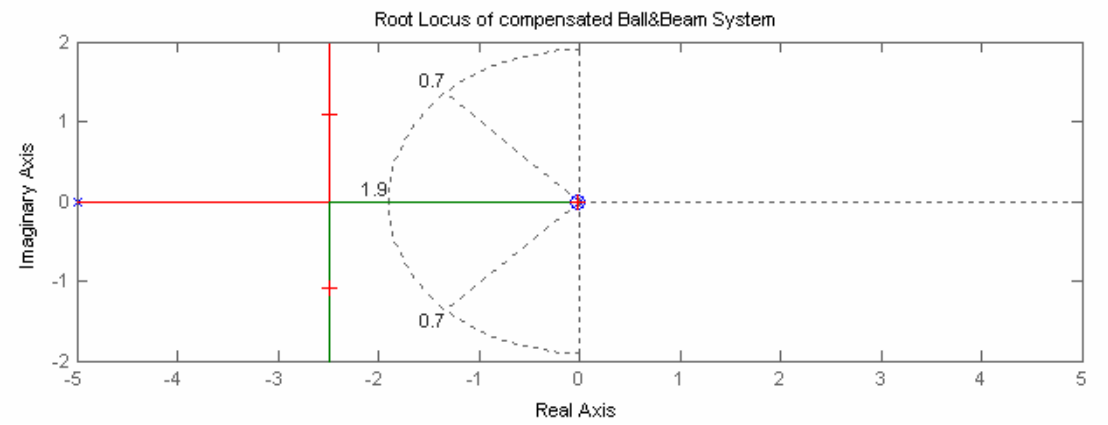



FIGURE 4-23 ROOT LOCUS OF THE OPEN-LOOP CONTROL SYSTEM WITH COMPENSATOR

We can use the `rlocfind` command to help us to do this. Add the following code to the end of your m-file:

```
[k,poles]=rlocfind(contr*plant)
```

Select a point near those indicated by the cross marks on the Root Locus plot. Add the following lines to your m-file to perform the analysis of the Root Locus method:

```
sys_cl=feedback(k*contr*plant,1);
t=0:0.01:5;
figure
step(sys_cl,t)
```

Run your m-file and select a point on the root locus similar to the selected point above. The step response should look like the following:

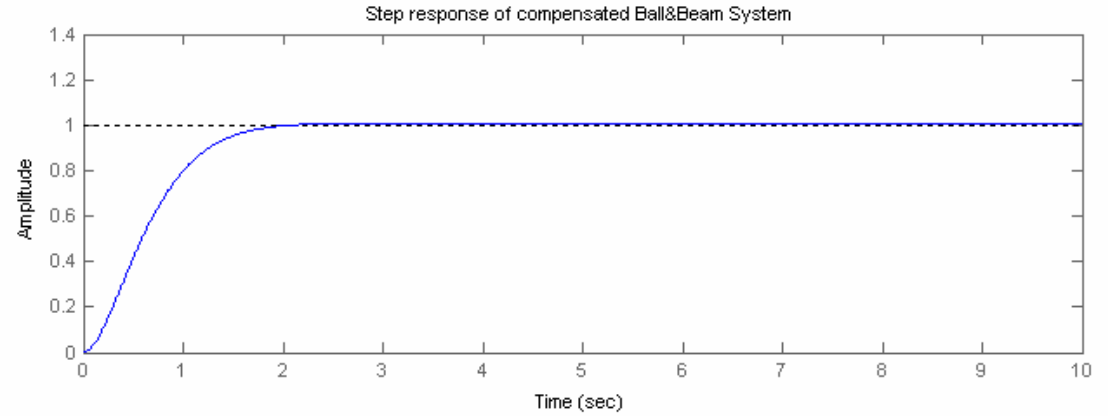



FIGURE 4-24 STEP RESPONSE WITH ROOT LOCUS DESIGNED CONTROLLER

From this plot we see that when a step input is given to the system, both the settling time and percent overshoot design criteria are perfectly met.

## Experiment

i. Open the Root Locus Control Demo in MATLAB Simulink Googol Educational Products Toolbox:

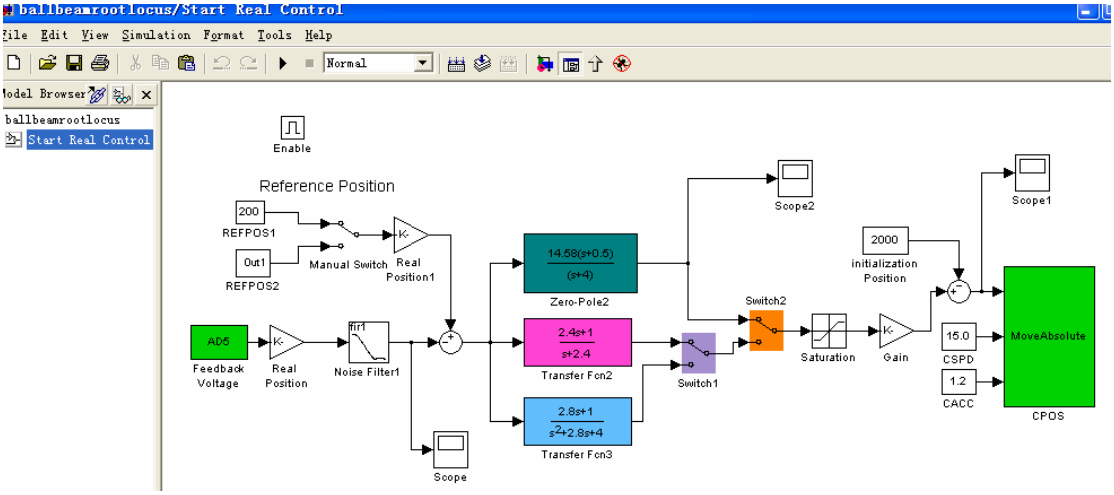



FIGURE 4-25 ROOT LOCUS CONTROLLER DEMO

ii. Set the parameters of the controller as the computed result, then run the demo, the result is as the following:

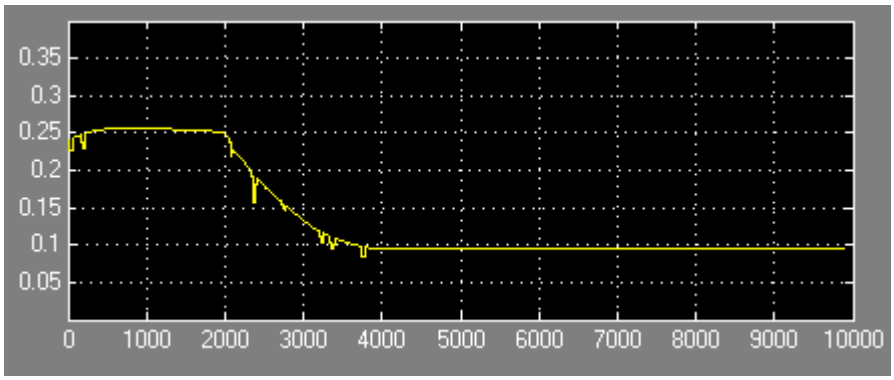



FIGURE 4-26 EXPERIMENT RUSULT OF ROOT LOCUS CONTROLLER (I)

iii. Change parameters of the controller:

Set

```
zo = 0.5;
po = 4;
```

selected_point = -3.7270 + 2.1250i

k =14.5801

poles =-1.6842 + 2.2895i, -1.6842 - 2.2895i, -0.6317

The following responses will be obtained:

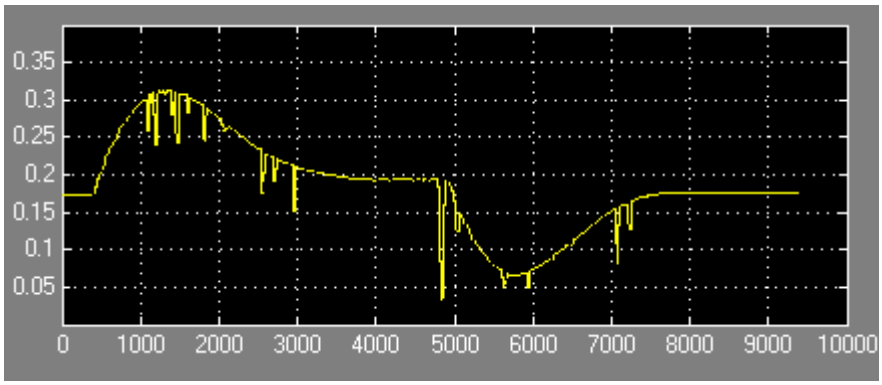



FIGURE 4-27 EXPERIMENT RUSULT OF ROOT LOCUS CONTROLLER (II)

Analyse the different results and compare them with the simulation result. What's the differences and explain why.

## 4.5 Frequency Response Method

The main idea of the frequency response method is to use the Bode plot of the open-loop transfer function of the plant (see Figure 4-21) to design the desired response of the closed-loop system. Adding a controller to the system changes the open-loop Bode plot, therefore changing the closed-loop system response.

Let's first draw the Bode plot for the original open-loop transfer function *W(s)* given by Equation 4-1. Create an m-file with the following code and then run it in the MATLAB command window:

```
m = 0.028;
R = 0.01;
g = -9.8;
L = 0.4;
d = 0.04;
J = 2*m*R^2/5;
K = (m*g*d)/(L*(J/R^2+m));   %simplifies input
num = [-K];
den = [1 0 0];
plant=tf(num,den);
bode(plant)
```

The resulting plot is demonstrated in Figure 4-28:

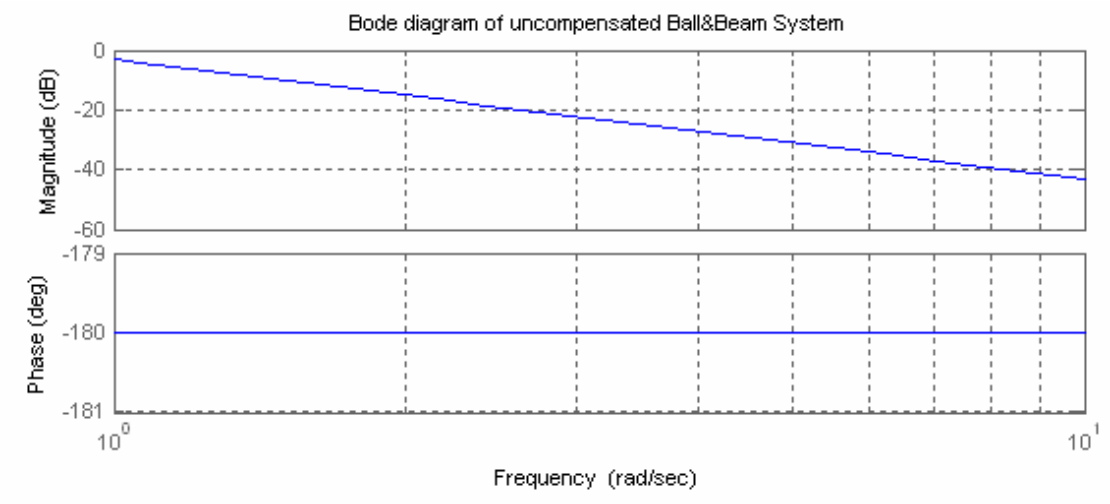



FIGURE 4-28 BODE DIAGRAM OF OPEN-LOOP SYSTEM BEFORE CONTROLLER DESIGN

From this plot we see that the phase margin is zero. Since the phase margin is defined as the change in open-loop phase shift necessary to make a closed-loop system unstable this means that our zero phase margin indicates our system is unstable. We want to increase the phase margin and we can use a lead compensator controller to do this.

We can add the following lead controller to our system to improve system response:

$$G_O(s) = K\frac{1+Ts}{1+aTs}$$

This compensator will add positive phase to our system over the frequency range *1/aT* and *1/T* which are called the corner frequencies. For our controller design we need a percent overshoot of less than 5% which corresponds to a *zeta* of 0.7. *zeta*100* provides the minimum phase margin to obtain the desired overshoot. So finally for our system requirements we require a phase margin greater than 70 degrees.

The following algorithm can be applied to obtain *T* and *a* parameters:

**i.** Determine the positive phase needed (as we concluded above the system needs al least 70 degrees)

**ii.** Determine the frequency where the phase should be added (center frequency). If we define the bandwidth frequency in our case as approximately 1.9 rad/s, we can choose a center frequency just before this, for example, 1.0 rad/s.

**iii.** Determine the constant *a* from the equation below

$$a = \frac{1-\sin\varphi}{1+\sin\varphi}$$

where phi refers to the desired phase margin. For 70 degrees, a=0.0311

**iv.** Determine *T* and *aT* from the following equations:

$$T = \frac{1}{w\sqrt{a}} \quad \text{and} \quad aT = \frac{\sqrt{a}}{w},$$

where *w* is the center frequency.

For 70 degrees and center frequency $w$=1, $aT$=0,176 and $T$=5.67.

So finally the transfer function of the computed lead compensator is:

$$G_O(s) = \frac{5.670s + 1}{0.176s + 1}$$

Remove the `bode` command from your last m-file and add the following program:

```
phi=70*pi/180;
a=(1-sin(phi))/(1+sin(phi));
w=1;
T=1/(w*sqrt(a));
kk= 1;
numlead = k*[T 1];
denlead = [a*T 1];
contr = tf(numlead,denlead);
bode(contr*ball)
```

You should get the following bode plot as shown in Figure 4-29. As we can see the system now has a phase margin in 70 degrees.

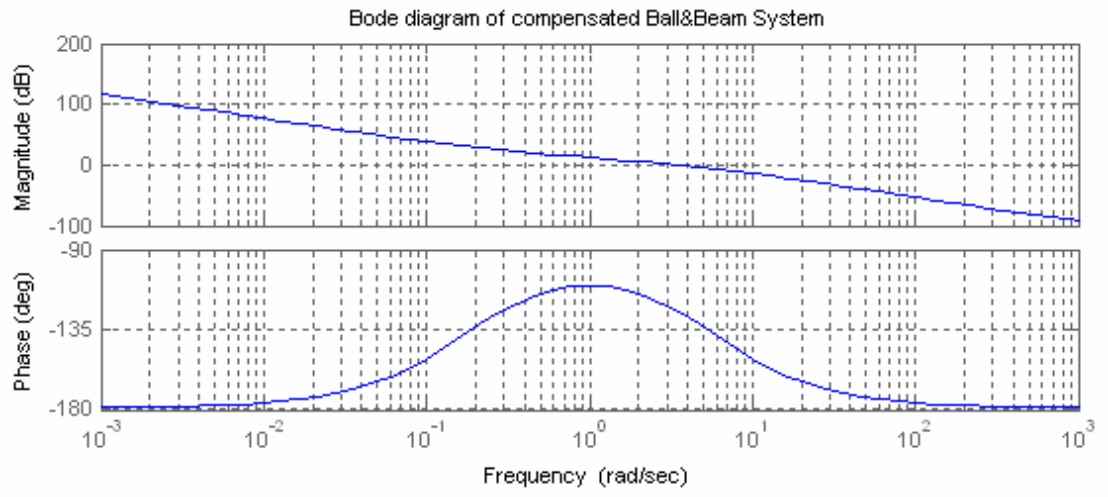



FIGURE 4-29 BODE PLOT OF SYSTEM WITH THE DESIGNED LEAD COMPENSATOR

The closed-loop response to a step input of 0.25 m can be simulated by adding the following lines of code into your m-file:

```
sys_cl = feedback(contr*ball,1);
t = 0:0.01:5;
step(0.25*sys_cl,t)
```

The resulting plot is depicted in Figure 4-30:

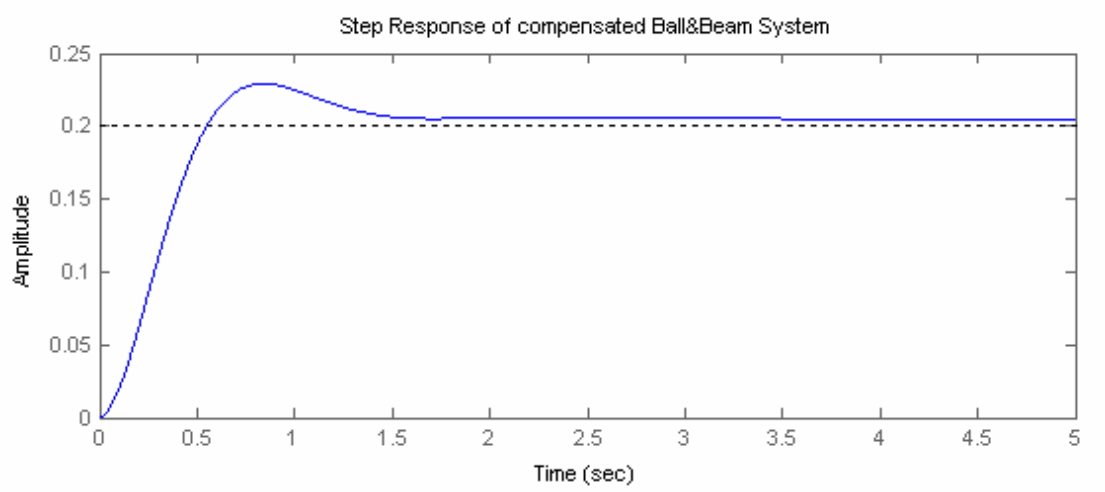



FIGURE 4-30 RESPONSE OF THE CLOSED-LOOP SYSTEM WITH LEAD COMPENSATOR

Although the system is now stable and the overshoot is slight, the settling time is not satisfactory. Increasing the gain will increase the crossover frequency and make the response faster.

With `kk=4` your response should look like:

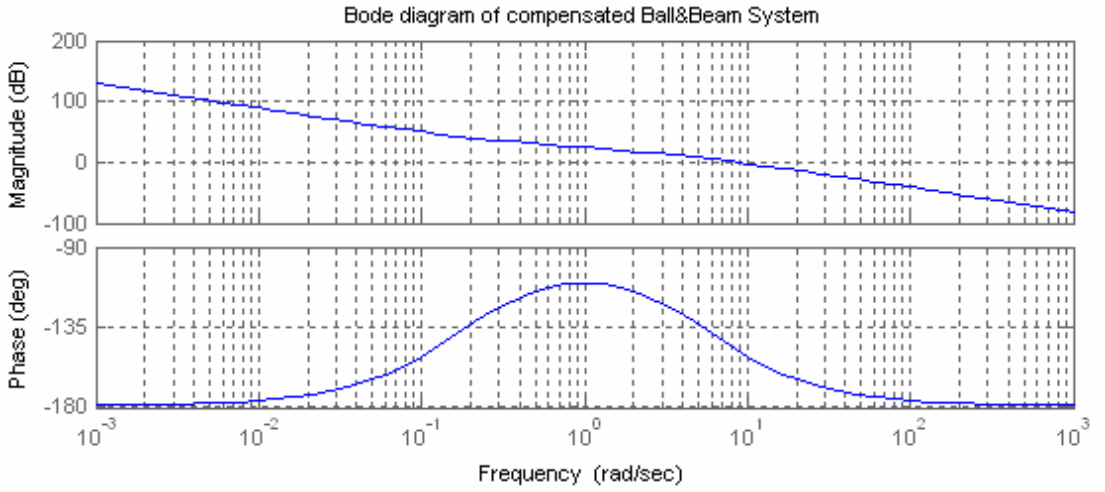



FIGURE 4-31 BODE PLOT OF THE OPEN-LOOP SYSTEM WITH THE DESIGNED LEAD COMPENSATOR AND INCREASED VALUE OF AMPLIFICATION GAIN

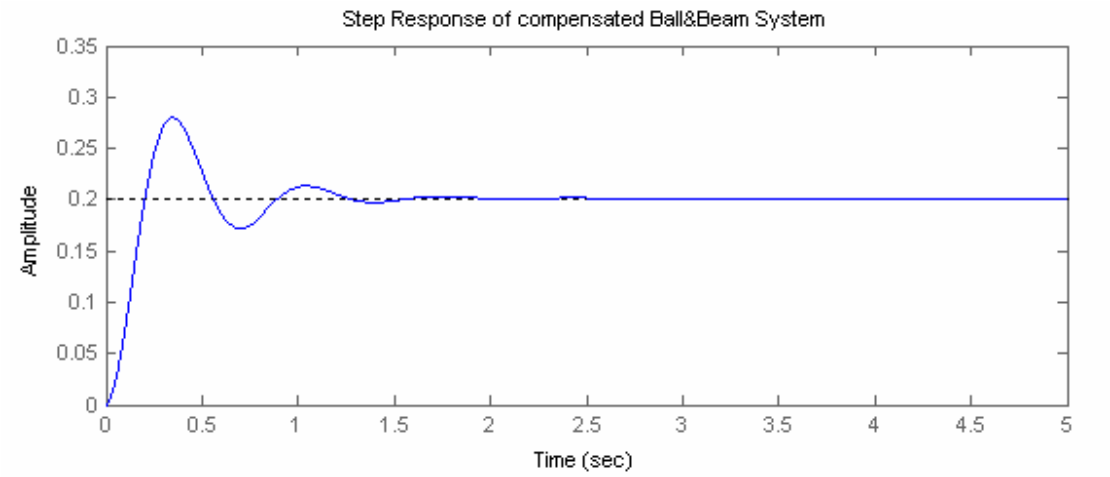



FIGURE 4-32 STEP RESPONSE OF THE CLOSED-LOOP SYSTEM WITH LEAD COMPENSATOR AND INCREAD VALUE OF AMPLIFICATION GAIN

You will see the response is faster, however, the overshoot is much too high. Increasing the gain further will just make the overshoot worse. Instead we can increase our lead compensator to decrease the overshoot. Try different numbers and see what happens. For example for phase margin 80 degrees, center frequency 1.9 rad/s and gain kk set to 2 the following response is obtained:

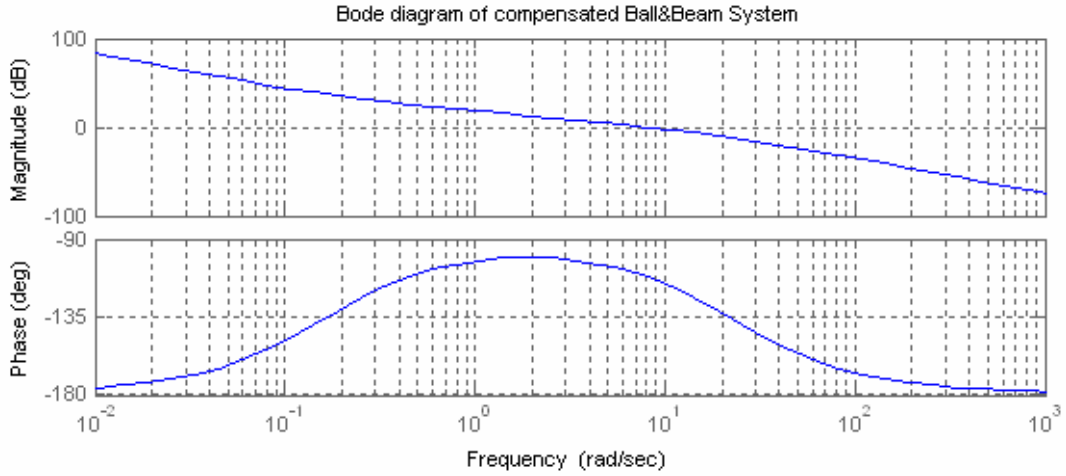



FIGURE 4-33 BODE DIGRAM WITH TUNED LEAD COMPENSATOR

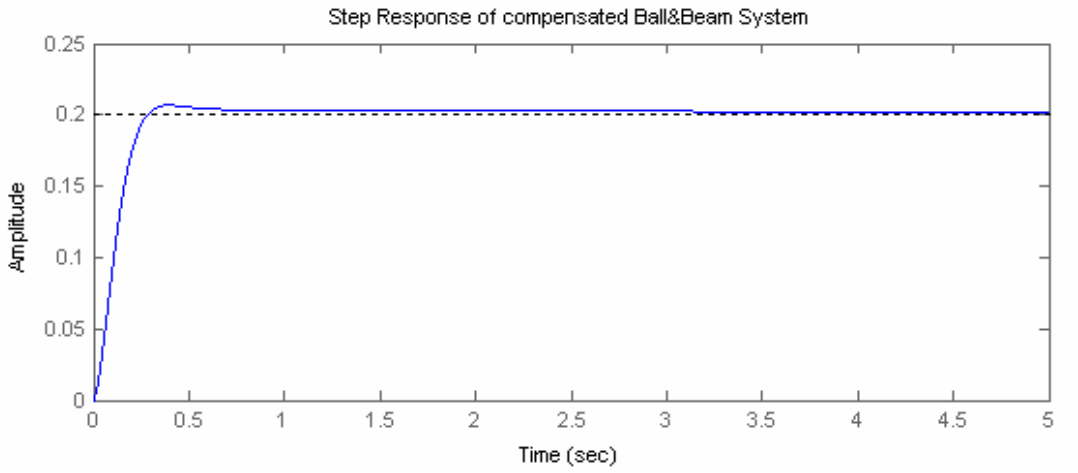



FIGURE 4-34 STEP RESPONSE OF THE CLOSED-LOOP SYSTEM WITH TURNED LEAD COMPENSATOR

## Experiment

i. Open the Frequency Response Control Demo in MATLAB Simulink Googol Educational Products Toolbox:

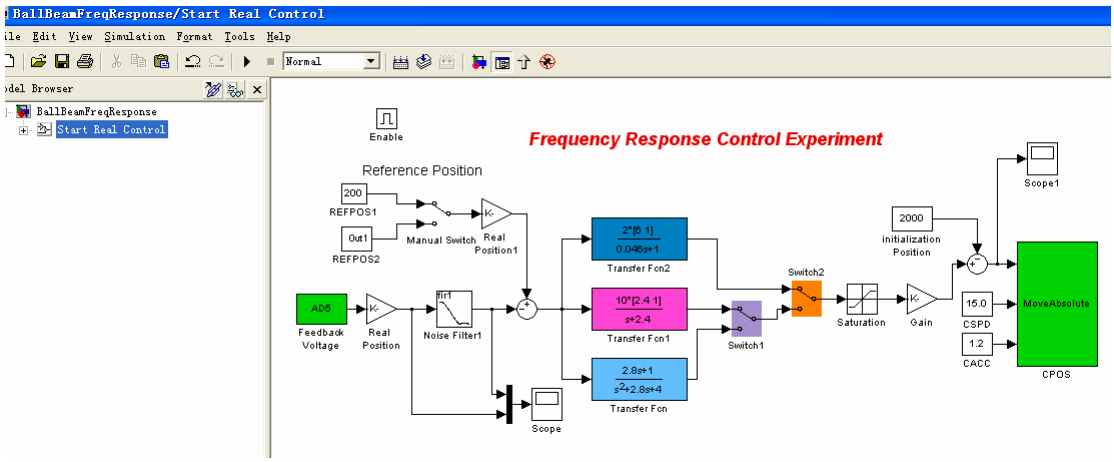



FIGURE 4-35 FREQUENCY RESPONSE CONTROL DEMO IN MATLAB SIMULINK

ii. Set the parameters of the controller as the results of Simulink, run demo and get the following result:

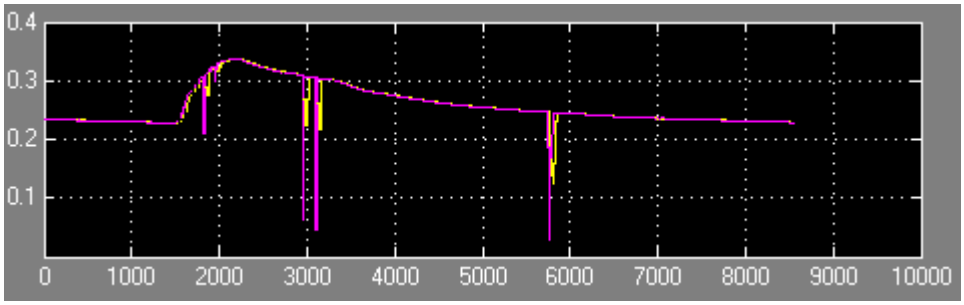



FIGURE 4-36 RESULT OF FREQUENCY RESPONSE

## 4.6 User define Controllers

Users are able to use their own advanced control algorithms on our ball&beam platform for research purpose by simply replace the control module with their own and run the demo.

# CHAPTER 5 TROUBLESHOOTING

The following trouble shooting table may help you to get your system to work. If the problem persists, consult your technical support or contact Googol Technology, Ltd.

| Symptom | Cause | Remedy |
|---|---|---|
| "**Board NOT Found**" error message is displayed in status bar at the bottom of IPM Motion Studio | No communication between PC and Ball&Beam intelligent drive | **1**. Make sure the Ball&Beam apparatus is powered up. Check the serial cable. Exit the IPM Motion Studio and launch it again. Use "**Tools \| Refresh Serial Settings**" menu command in IPM Motion Studio to retry the communication with the drive. Proceed to step 2 below if the problem persist. |
| | | **2.** From the **Tools** menu choose **Options**. This command will open **Options** dialog box. Select **Serial Port** tab. Make sure that "RS232" is selected in **Serial Type** dialog box and your serial cable is connected to the same PC port as selected in **COM Port Number**. **Axis ID** should be set to 255. Adjust the settings if required. Click "**OK**" button. Use "**Tools \| Refresh Serial Settings"** to retry the communication. If the communication works properly the product ID (as **P069.001.C01.S000A**) is displayed in the status bar instead of the error message. Proceed to step 3 if the problem is not solved |
| | | **3.**Open "**Tools \| Options \| Serial Port**" again. Try to start using a smaller **Baud Rate** value (as 9600). Click "**OK**" button. Use "**Tools \| Refresh Serial Settings"** to retry the communication. In case of success, try to increase the baud rate up to the maximal 115K allowed by the PC. In case of failure consult with "IPM Motion Studio User Manual" for more troubleshooting details. |

| | | |
|---|---|---|
| Communication between PC and Ball&Beam drive produces error message from time to time | Unstable communication caused by noise, operating conditions, etc. | **1.** Try to use smaller baud rates for communication as explained in step 3 of the above problem. |
| | | **2.** Try to increase "**Read Interval Timeout**", "**Timeout multiplier**" and "**Timeout Constant**" parameters in the "**Options \| Serial Port"** dialog box. Note that this is related to the operation of the PC's hardware, and the default values for these parameters normally should not be modified. |
| The motor works abnormally | Damaged connection between the motor and the servo drive | Open the chassis of the Ball&Beam mechanical plant and check that the connectors of the drive are fastened properly. Refer to "IPM 100 User Manual" for wiring schematics if necessary. |
| The demo program doesn't balance the position of the ball on the beam | The position error is too big | The position of the ball at the start of the experiment is incorrect. Exit the IPM Motion Studio program and reset Ball&Beam apparatus. Set the ball as described in Getting Started. Repeat the demo again. |
| | | The gains of the PID controller are too small. Increase proportional gain $K_P$ |

# APPENDIX A REAL CONTROL IN IPM MOTION STUDIO

## Step by Step Installation using IPM Motion Studio

Please follow the following installation steps to set-up your Ball&Beam servo system:

Step 1: Installing Control Software
Step 2: Establishing Communication
Step 3: Running the Demo Program
Step 4: Analyzing Results of the Demo Experiment

### Step 1: Installing Control Software

This section explains the process of installing the software on a hard disk of the PC.

**i.** Insert the setup CD-ROM into CD-ROM drive. Browse it.

**ii.** Install IPM Motion Studio software from the CD by selecting the corresponding installation program. If you use Windows NT/2K/XP please make sure that you own administrator privileges. The setup will create a specific group for all components of the IPM Motion Studio package.

**iii.** Copy a demo application example “BallBeamProject” from CD-ROM to “…\IPM Motion Studio\Projects” folder in the installation catalog on the hard drive.

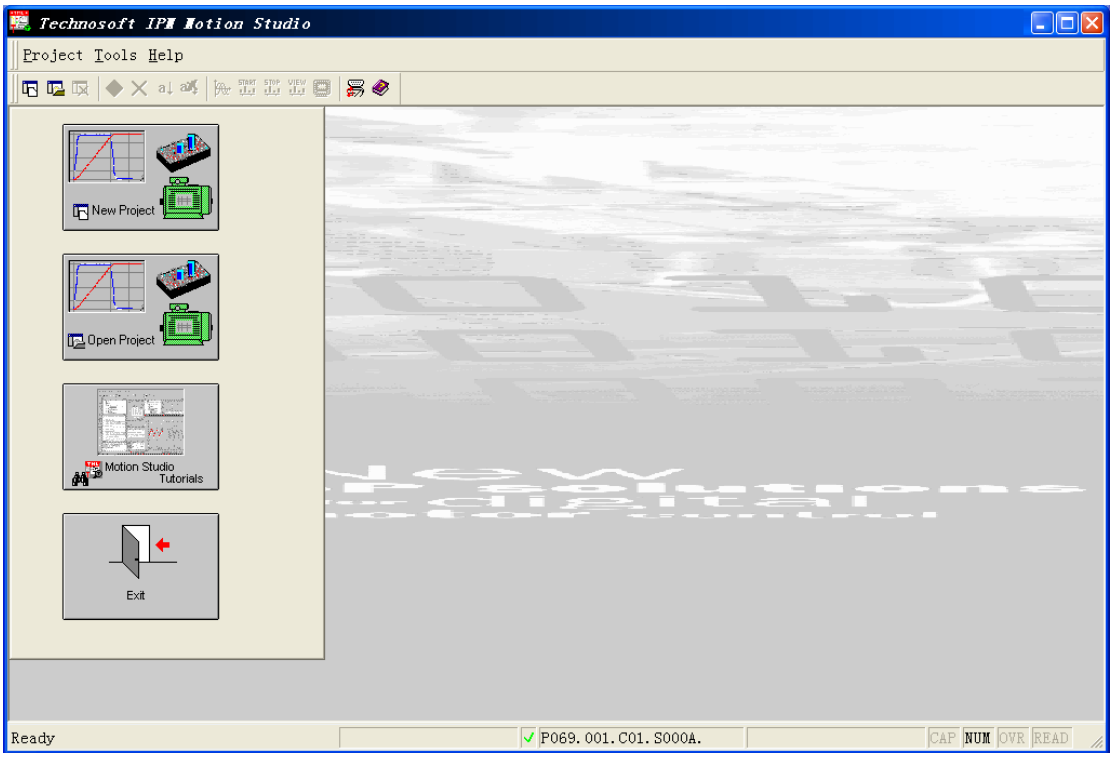



FIGURE A-1 IPM MOTION STUDIO MAIN WINDOW

## Step 2: Establishing Communication

**i.** First power up the PC if you turned it off after installation procedure

**ii.** Secondly power up the Ball&Beam apparatus

**iii.** Launch IPM Motion Studio through "**Start | Programs | IPM Motion Studio | IPM Motion Studio**" menu command (see Figure A-1)

**iv.** If everything operates properly the program will detect the Ball&Beam drive as communicating with the PC. If the drive is not detected, the "**Board NOT found**" message is displayed in the status bar at the bottom of the window. Refer to Chapter 5 Troubleshooting to correct the problem.

**v.** If the drive is detected, the drive's product ID (as **P069.001.C01.S000A**) will be displayed in the status bar of the window as shown in Figure A-1. The environment is ready to start the demo program.

## Step 3: Running the Demo Program

All the applications in IPM Motion Studio are organized as projects. A project is specific for a selected type of the intelligent drive and a specific motor type (DC brushed or brushless). The application demo project "BallBeamProject" provided with the Ball&Beam product has already been pre-configured and properly tuned so that the novel user can start it immediately without any efforts just following the procedure described in this section. In order to run the demo successfully from the

first attempt, please don't change any settings or parameters in the project while following the explanations below.

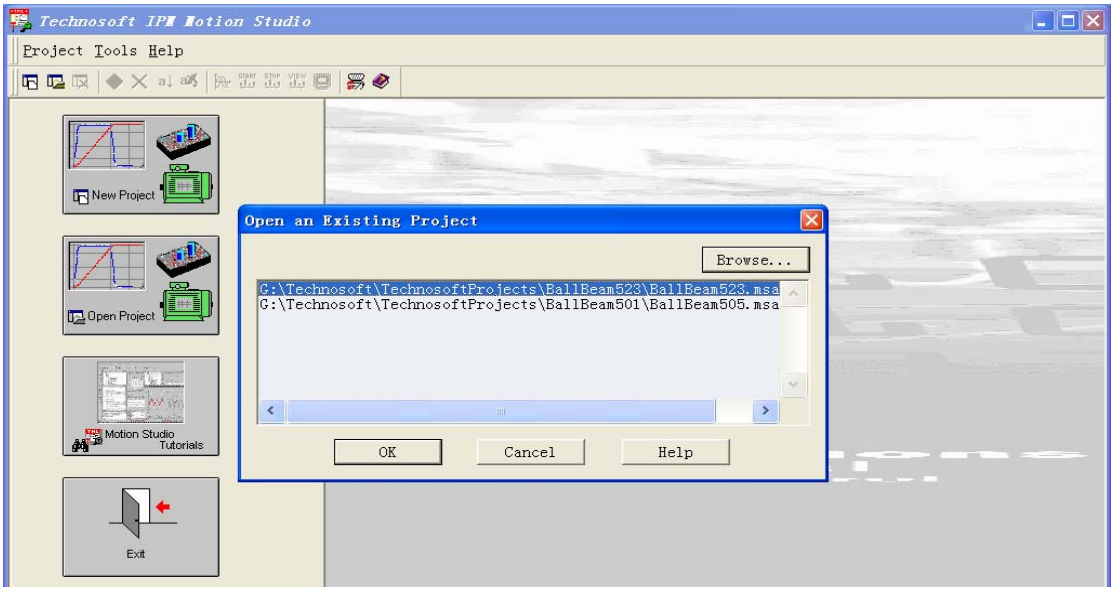



FIGURE A-2 OPENING AN EXISTING PROJECT

**i.** *Opening the Demo Project*

Click "**Open Project**" button in the Main Window. The dialog window "**Open an Existing Project**" is displayed (see Figure A-2). This window contains all the projects currently stored in "…\IPM Motion Studio\Projects" folder of the IPM Motion Studio installation catalog. Double click the "BallBeamProject" entry to open the demo project. If you haven't copied the demo project from the CD to the hard drive before or coppied it to different location, use "**Browse**" button to locate the demo program.

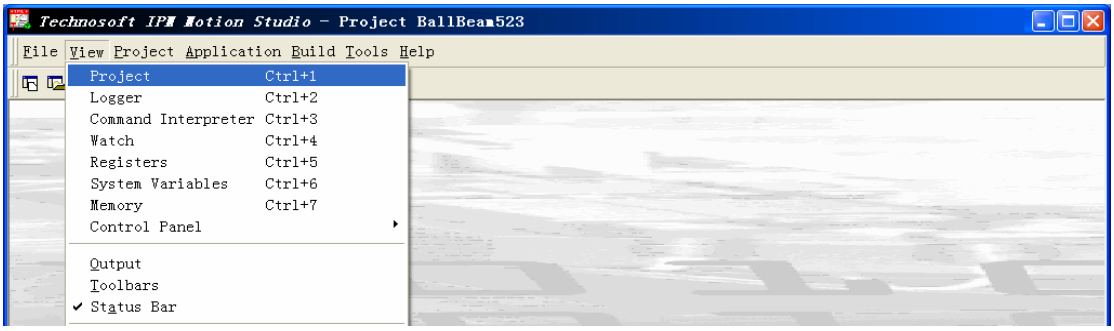



FIGURE A-3 OPENING THE PROJECT WINDOW

**ii.** *Studying the project diagram structure*

Select "**View | Project**" from the Main Window menu as depicted in Figure A-3 if you don't see a schematics diagram of the motion system like the one shown below. It contains the tree basic elements of the system, i.e. the motor and sensors, the drive and, as a separate component, the motion reference generator module. Each of these

elements can be selected and opened to define and edit the parameters and specific settings associated with the element. If the new project is created, then it is opened with some elements already preloaded. In this case the programmer just needs to change/test/validate these elements in order to fit the configuration of his/her motion application.

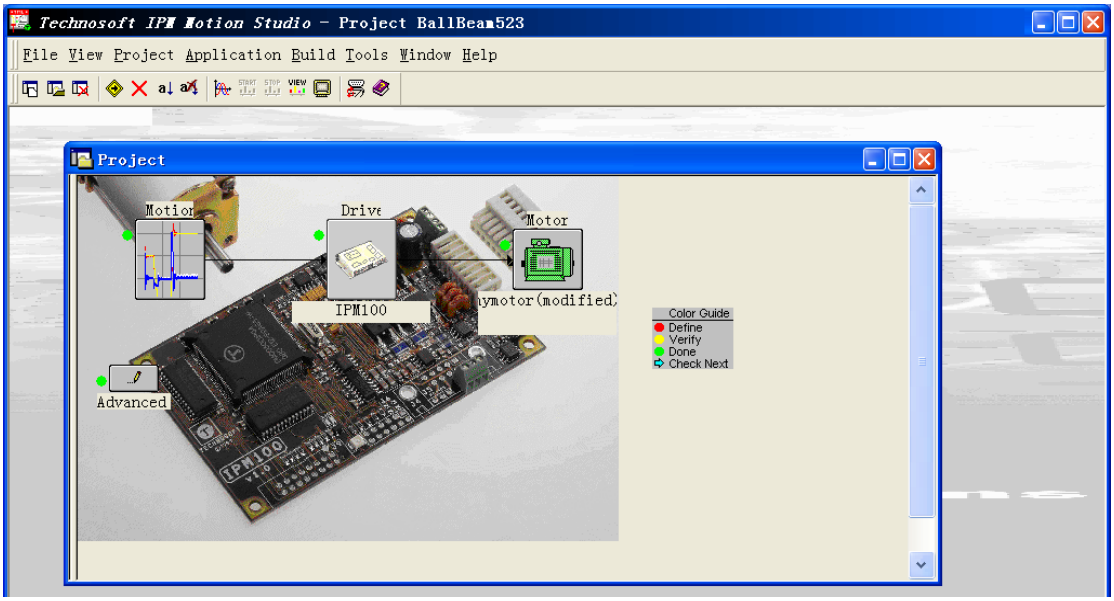



FIGURE A-4 MOTION SYSTEM DIAGRAM IN IPM MOTION STUDIO

In our "BallBeamProject" demo project all the elements have already been adjusted to Ball&Beam motion application. You can examine any of them by clicking on the corresponding icon, but, it has been mentioned above, please don't change any parameters and values.

For example, click the "Motion" element. It will open the Motion Wizard dialog window that helps to create motion sequences/programs without actually writing the code (see Figure A-5). Expand "Variables declarations" and "Main loop" in the "Code" table to see the control program. It contains the actual TML code for Ball&Beam demo. The listing of the program with detailed comments is given in Appendix B.

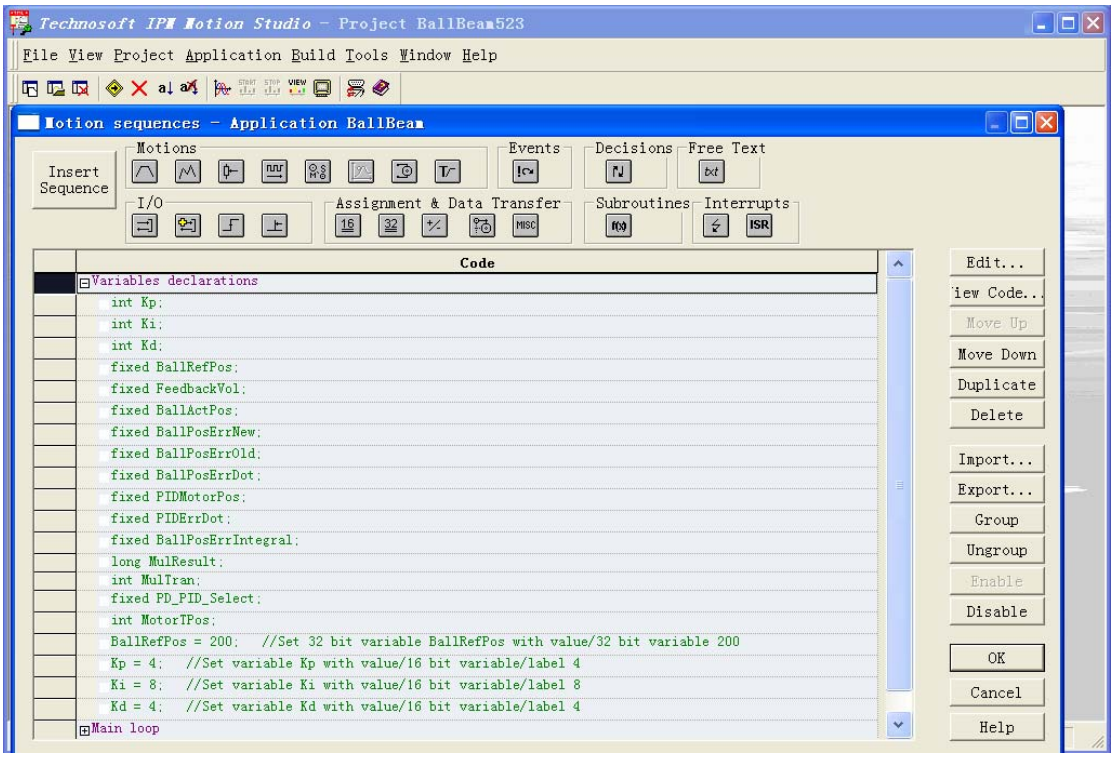



FIGURE A-5 THE MOTION WIZARD DIALOG

Click “OK” button to close the window and return to the interface.

If the communication with the IPM drive is lost, an error message “**Cannot open the serial port**” is displayed. Refer to Troubleshooting to correct the problem and then repeat all the steps above again.

**iii.** *Preparing the mechanical plant*

Move the beam to the lower position and place as shown in Figure A-6:

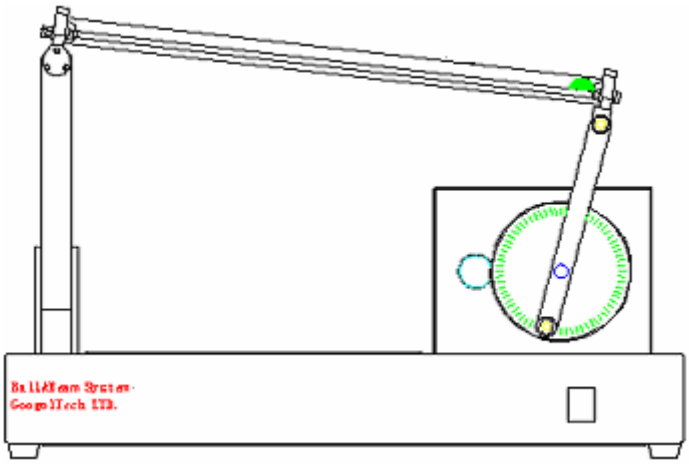


FIGURE A-6 CORRECT SETUP ARRANGEMENT BEFORE RUNNING THE DEMO

**iv.** *Starting the program*

Click the button in the Toolbar to run the program. The drive will balance the position of the ball around the central point of the beam. You can apply small disturbances to the system by pushing the ball by finger in any direction along the beam. The control system will immediately try to return the ball back to the original point and stabilize its position.

At any time you can abort the experiment by clicking or buttons on the Toolbar, or simply switching off the power of the Ball&Beam apparatus.

## Step 4: Analyzing results of the demo experiment

While running the demo, choose "**View | Control Panel | Ball&Beam**" as depicted in Figure A-7 to open the Control Interface Window of the experiment.

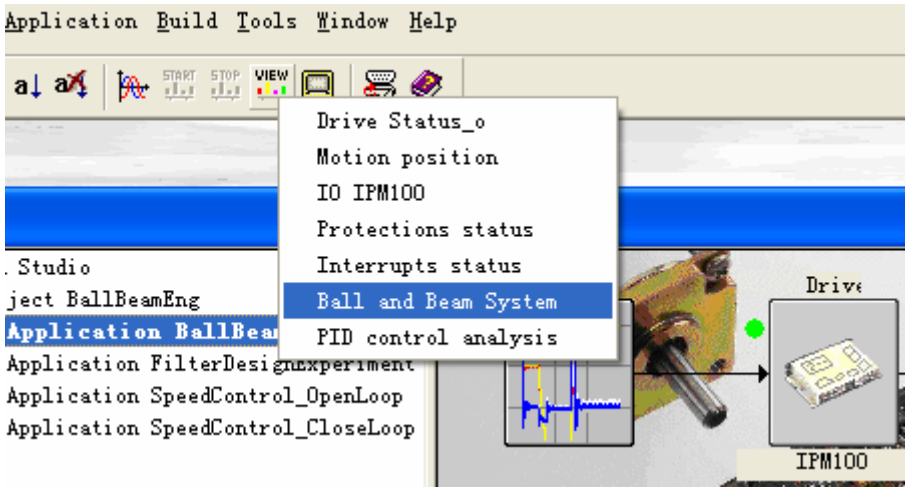



FIGURE A-7 OPENING THE BALL&BEAM CONTROL WINDOW

You will see that the control interface (see Figure A-8) is divided into two parts: the left panel shows the position and velocity of the ball, position errors, etc, while the right one contains control elements which allow the user to specify the target position of the ball on beam, turn the PID parameters and enable/disable the motion.

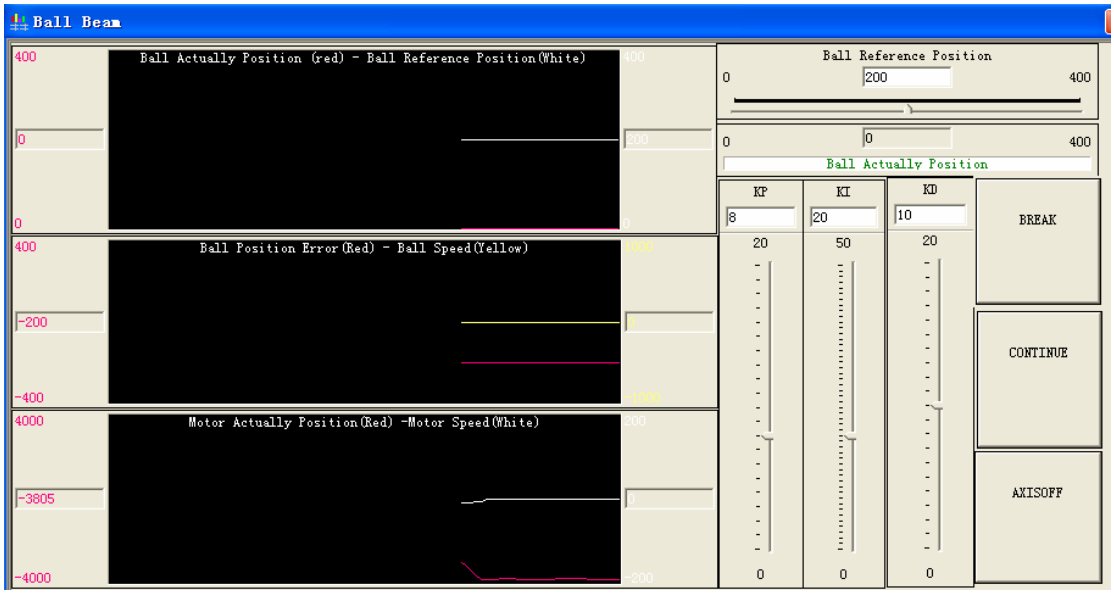



FIGURE A-8 THE DEMO CONTROL WINDOW

Position the mouse pointer in front of the Control Window, click the right key of the mouse and select "**Start"** in the pop-up menu to start collecting and visualizing the historical data from the control system:

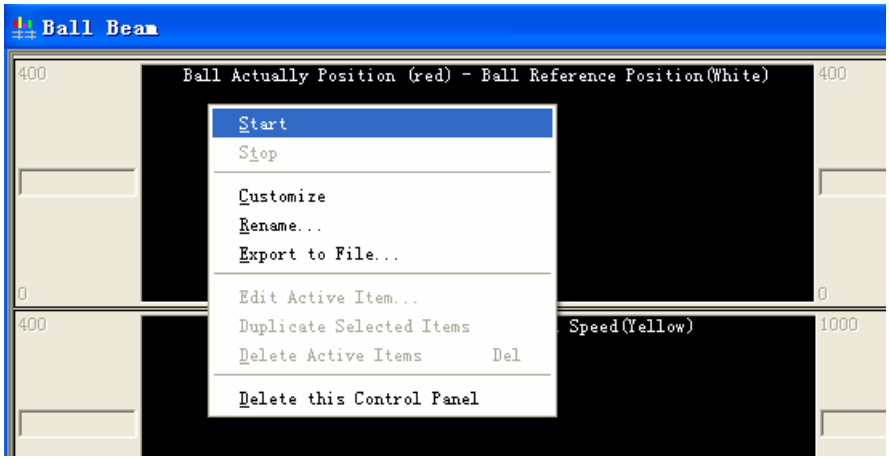



FIGURE A-9 STARTING DATA COLLECTION

As a result the real-time data about ball's actual and reference positions, position error, actual ball velocity, etc can be conveniently observed in the corresponding graphs:

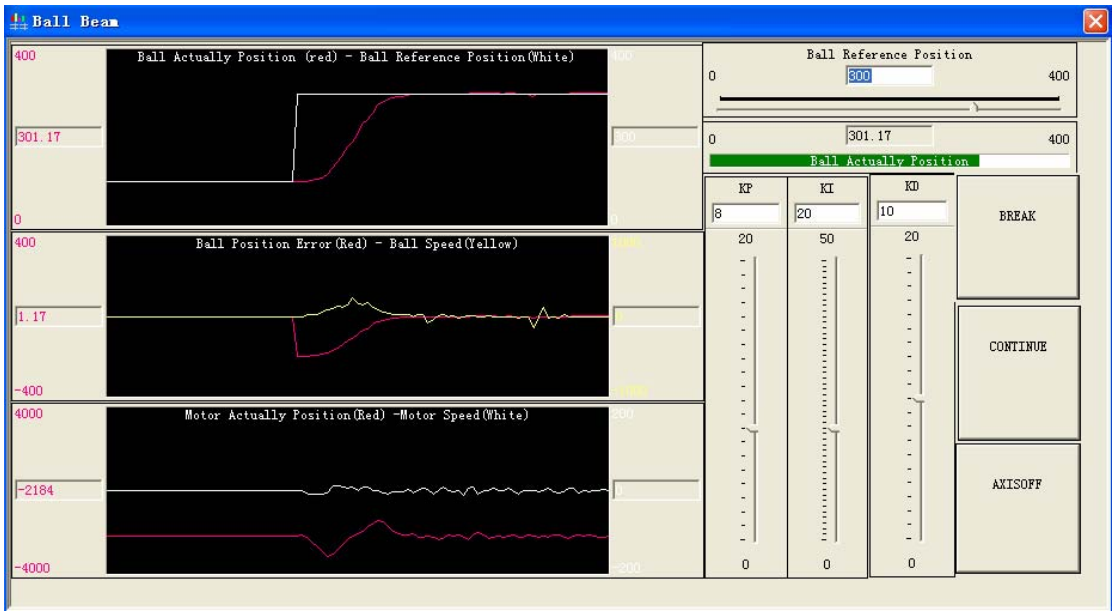



FIGURE A-10 RESULTS OF THE DEMO EXPERIMENT

Click "**Break**" button to pause the motion, "**Continue**" to continue the experiment and "**Axis off**" button to stop the motor.

Try to change the reference position of the ball in the right panel.

When you are finished with the experiment, click "Axis off" button and then switch off the Ball&Beam apparatus.

# APPENDIX B LISTING OF DEMO

## Part one: variables declaration

```
int Kp;                          //proportional gain of PID controller
int Ki;                          //integral gain of PID controller
int Kd;                          //differential gain of PID controller

fixed BallRefPos;                //target position
fixed FeedbackVol;               //feedback voltage
fixed BallActPos;                //actual position
fixed BallPosErrNew;             //current position error
fixed BallPosErrOld;             //last position error

fixed BallPosErrDot;             //velocity of the ball
fixed PIDMotorPos;               //position of the motor caused by
proportion

fixed PIDErrDot;                 //position of the motor caused by
differental
fixed BallPosErrIntegral;        //target position of the motor caused
by integral
long MulResult;                  //multiplication result(used in DSP)
int MulTran;                     //multiplication transfer variable
fixed PD_PID_Select;             //PD or PID controllers selection
int MotorTPos;                   //target position of the motor
BallRefPos = 200;                //Set 32 bit variable BallRefPos with
                                 value/32 bit variable 200
Kp = 4;                          //Set variable Kp with value/16 bit
                                 variable/label 4
Ki = 8;                          //Set variable Ki with value/16 bit
                                 variable/label 8
Kd = 4;                               //Set variable Kd with value/16
                                 bit variable/label 4
```

## Part two: The main control loop

```
loop:
   FeedbackVol(H) = 0;
                  //Set HIGH part of variable FeedbackVol with
                  value/16 bit variable 0
   FeedbackVol(L) = AD5;
                  //Set LOW part of variable FeedbackVol with
                  value/16 bit variable AD5
   FeedBackVol*400 << 0;
                  //Set P register with the product of variable
                  FeedBackVol with value/variable 400 LEFT shifted
                  with 0 bits.
   MulResult = PROD(L);
                  //Set 32 bit variable MulResult with value/32 bit
                  variable PROD(L)


   MulTran = MulResult(H);
                  //Set variable MulTran with HIGH part of 32 bit
                  variable MulResult
   BallActPos(H) = MulTran;
                  //Set HIGH part of variable BallActPos with
                  value/16 bit variable MulTran
   MulTran = MulResult(L);
                  //Set variable MulTran with LOW part of 32 bit
                  variable MulResult
   BallActPos(L) = MulTran;
                  //Set LOW part of variable BallActPos with value/16
                  bit variable MulTran
   BallPosErrNew = BallActPos;
                  //Set 32 bit variable BallPosErrNew with value/32
                  bit variable BallActPos
   BallPosErrNew -= BallRefPos;
                  //Substract from variable BallPosErrNew the
                  value/variable BallRefPos
//calculate position errors
   BallPosErrDot = BallPosErrNew;
                  //Set 32 bit variable BallPosErrDot with value/32
                  bit variable BallPosErrNew
   BallPosErrDot -= BallPosErrOld;
                  //Substract from variable BallPosErrDot the
                  value/variable BallPosErrOld
   BallPosErrOld = BallPosErrNew;
                  //Set 32 bit variable BallPosErrOld with value/32
                  bit variable BallPosErrNew// position errors
                  defficiental
```

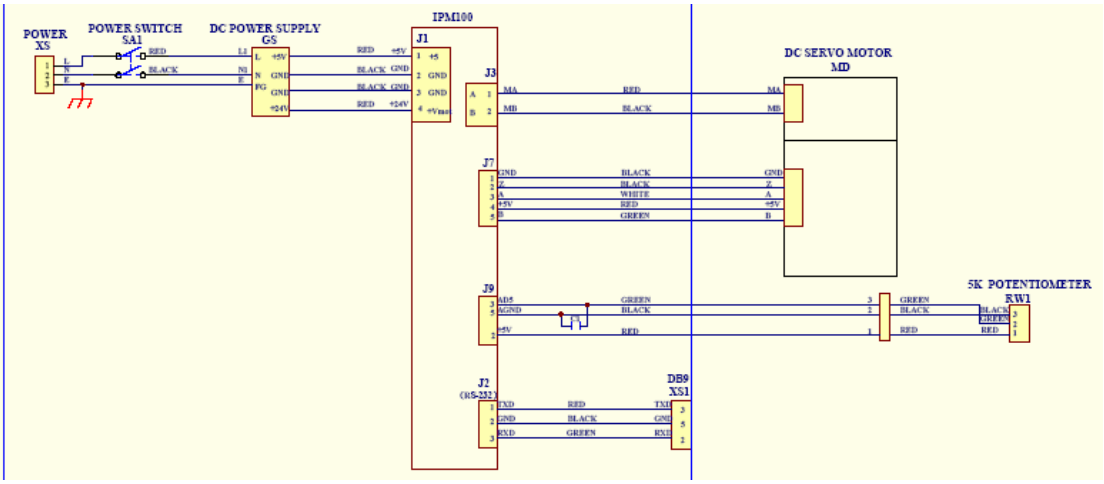
```
   BallPosErrDot*20 << 0;
                  //Set P register with the product of variable
                  BallPosErrDot with value/variable 20 LEFT shifted
                  with 0 bits. // sample period 50ms.
   MulResult = PROD;
                  //Set 32 bit variable MulResult with value/32 bit
                  variable PROD
   MulTran = MulResult(H);
                  //Set variable MulTran with HIGH part of 32 bit
                  variable MulResult
   BallPosErrDot(H) = MulTran;
                  //Set HIGH part of variable BallPosErrDot with
                  value/16 bit variable MulTran
   MulTran = MulResult(L);
                  //Set variable MulTran with LOW part of 32 bit
                  variable MulResult
   BallPosErrDot(L) = MulTran;
                  //Set LOW part of variable BallPosErrDot with
                  value/16 bit variable MulTran
   PD_PID_Select = BallPosErrDot;
                  //Set 32 bit variable PD_PID_Select to value/32 bit
                  variable BallPosErrDot   // select PID controller
                  according to errors deferential


   GOTO BallPosErrDot_GT_0, PD_PID_Select, GT;
                  //GOTO BallPosErrDot_GT_0 if PD_PID_Select > 0
   PD_PID_Select += 50;
                  //Add to variable PD_PID_Select the value 50
   GOTO PD_control, PD_PID_Select, LT;
                  //GOTO PD_control if PD_PID_Select < 0
   GOTO CheckPosErr;
BallPosErrDot_GT_0:
   PD_PID_Select -= 50;
                  //Substract   from   variable   PD_PID_Select   the
                  value/variable 50
   GOTO PD_Control, PD_PID_Select, GT;
                  //GOTO PD_Control if PD_PID_Select > 0
   GOTO CheckPosErr;
                  //choose PID controller when the vecocity is under
                  50mm/s
CheckPosErr:
   PD_PID_Select = BallPosErrNew;
                  //Set 32 bit variable PD_PID_Select with value/32
                  bit variable BallPosErrNew
```

```
   GOTO BallPosErr_GT_0, PD_PID_Select, GT;
                  //GOTO BallPosErr_GT_0 if PD_PID_Select > 0
   PD_PID_Select += 50;
                  //Add to variable PD_PID_Select the value/variable
                  50
   GOTO PD_control, PD_PID_Select, LT;
                  //GOTO PD_control if PD_PID_Select < 0
   BallPosErrIntegral += BallPosErrNew;
                  //Add    to    variable    BallPosErrIntegral    the
                  value/variable BallPosErrNew
   GOTO PID_control;
BallPosErr_GT_0:
   PD_PID_Select -= 50;
                  //Substract   from   variable   PD_PID_Select   the
                  value/variable 50
   GOTO PD_Control, PD_PID_Select, GT;
                  //GOTO PD_Control if PD_PID_Select > 0
   BallPosErrIntegral += BallPosErrNew;
                  //Add    to    variable    BallPosErrIntegral    the
                  value/variable BallPosErrNew
   GOTO PID_Control;
                  //choose PID controller when the position error is
                  less than 50mm
PD_control:
   BallPosErrIntegral = 0;
                  //Set  32  bit  variable  BallPosErrIntegral  with
                  value/32 bit variable 0   //integral is zero
PID_control:
//proportion
BallPosErrNew*Kp << 0;
                  //Set  P  register  with  the  product  of  variable
                  BallPosErrNew with value/variable Kp LEFT shifted
                  with 0 bits.
   MulResult = PROD;
                  //Set 32 bit variable MulResult with value/32 bit
                  variable PROD
   MulTran = MulResult(H);
                  //Set variable MulTran with HIGH part of 32 bit
                  variable MulResult
   PIDMotorPos(H) = MulTran;
                  //Set  HIGH  part  of  variable  PIDMotorPos  with
                  value/16 bit variable MulTran
   MulTran = MulResult(L);
```

```
                //Set variable MulTran with LOW part of 32 bit
                variable MulResult
   PIDMotorPos(L) = MulTran;
                //Set LOW part of variable PIDMotorPos with
            value/16 bit variable MulTr
//integral
BallPosErrIntegral >>= 6;
                //Shift RIGHT variable BallPosErrIntegral by 6 bits.
   BallPosErrIntegral*Ki << 0;
                //Set P register with the product of variable
                BallPosErrIntegral with value/variable Ki LEFT
                shifted with 0 bits.
   MulResult = PROD;
                //Set 32 bit variable MulResult with value/32 bit
                variable PROD
   MulTran = MulResult(H);
                //Set variable MulTran with HIGH part of 32 bit
                variable MulResult
   PIDMotorPos(H) += MulTran;
                //Add to variable PIDMotorPos(H) the value/variable
                MulTran
   MulTran = MulResult(L);
                //Set variable MulTran with LOW part of 32 bit
                variable MulResult
   PIDMotorPos(L) += MulTran;
                //Add to variable PIDMotorPos(L) the value/variable
                MulTran
   BallPosErrIntegral <<= 6;
                //Shift LEFT variable BallPosErrIntegral by 6 bits.
//differental
BallPosErrDot*Kd << 0;
                //Set P register with the product of variable
                BallPosErrDot with value/variable Kd LEFT shifted
                with 0 bits.
   MulResult = PROD;
                //Set 32 bit variable MulResult with value/32 bit
                variable PROD
   MulTran = MulResult(H);
                //Set variable MulTran with HIGH part of 32 bit
                variable MulResult
   PIDMotorPos(H) += MulTran;
                //Add to variable PIDMotorPos(H) the value/variable
                MulTran
   MulTran = MulResult(L);
```

```
                  //Set variable MulTran with LOW part of 32 bit
                  variable MulResult
   PIDMotorPos(L) += MulTran;
                  //Add to variable PIDMotorPos(L) the value/variable
                  MulTran
//set up the aim position and motion mode of the motor
MotorTPos = PIDMotorPos(H);
                  //Set variable MotorTPos with HIGH part of 32 bit
                  variable PIDMotorPos
   CPOS = MotorTPos<<0;
                  //Set 32 bit variable CPOS with 16 bit value of
                  variable MotorTPos left shifted with 0bits
   CACC = 1.27324;
                  //Acceleration command for position profile
                  acceleration of the motor
   CSPD = 8.00000;
                  //Speed command for position profile velocity of
                  the motor
   CPA;           //Position command is Absolute
   MODE PP3;      //Set Position Profile Mode 3
   TUM1;          //Set Target Update Mode 1
   UPD;           //Update immediately
                  //The hour hand is 10 DSP period: 10*5ms=50ms
   !RT 10;        //Set event  if RelativeTime >= 10
   WAIT!;         //WAIT until event occurs
GOTO loop;        //the main loop ends
```

# APPENDIX C ELECTRIC DIAGRAM



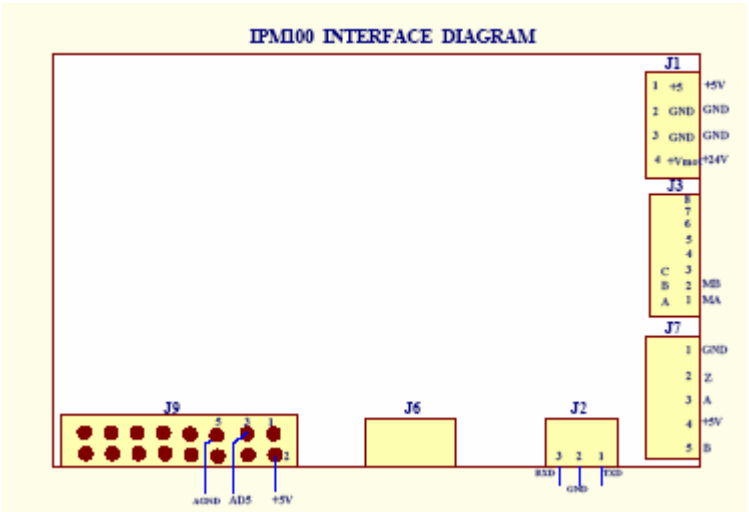




FIGURE C-1 ELECTRIC WIRE DIAGRAM