Experiment 1: Simulation of continuous and digital PID-Controllers

1. Introduction

Figure 1 shows a typical continuous feedback system. Almost all of the continuous controllers can be built using analog electronics.

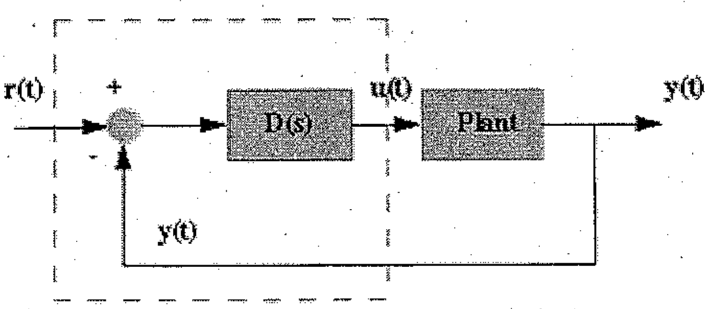


Fig. 1

The continuous controller, enclosed in the dashed square, can be replaced by a digital controller, shown in Fig. 2, that performs the same control task as the continuous controller. The basic difference between these controllers is that the digital system operates on discrete signals (or samples of the sensed signal) instead of continuous signals.

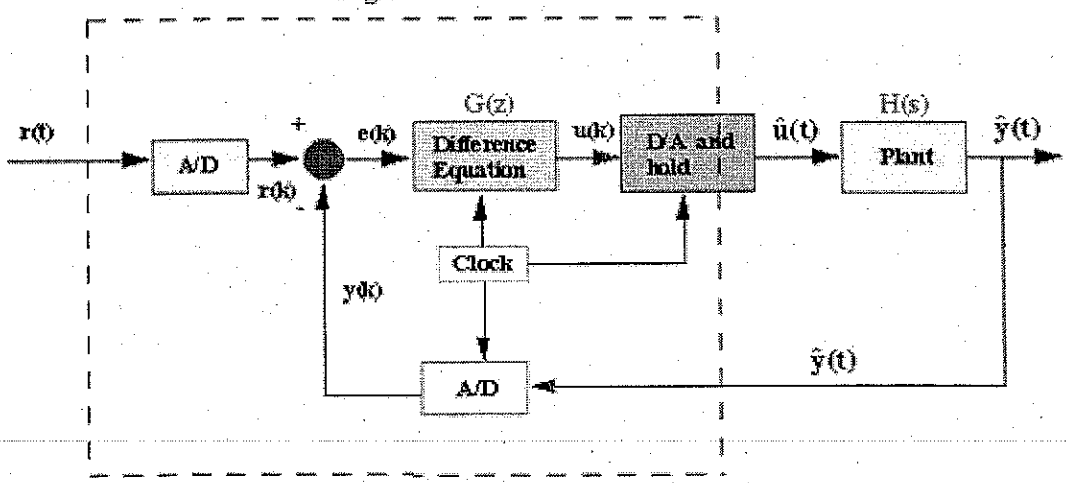


Fig. 2

The different types of signals in Fig. 2 can be represented by the following plots:

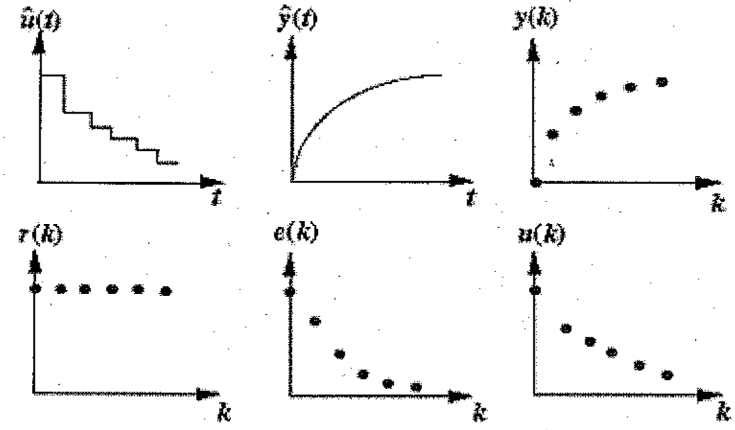


Fig.3

As seen in Fig.3 the digital control system contains both discrete and the continuous portions. When designing a digital control system, the discrete equivalent of the continuous portions has to be determined.
For this technique, a part of the digital control system will be rearranged as seen in Fig. 4.

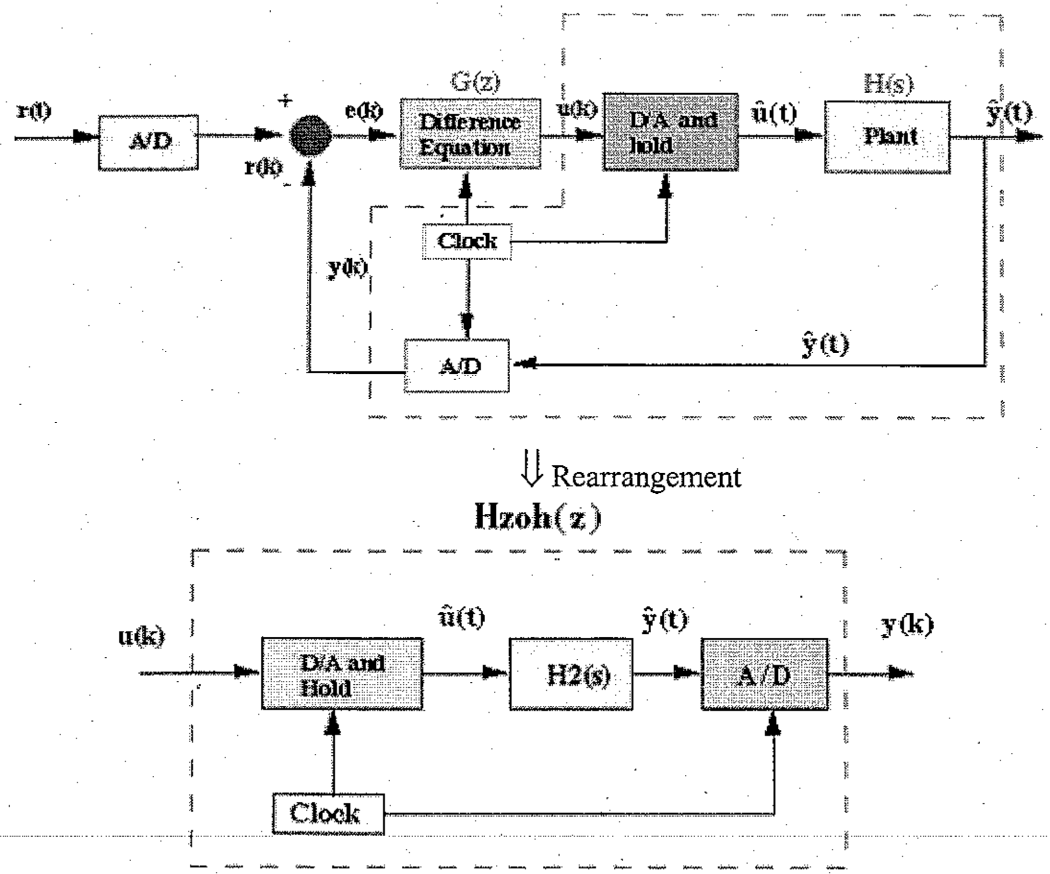


Fig. 4

The **clock** connected to the **D/A** and **A/D converters** supplies a pulse every T seconds and each D/A and A/D sends a signal only when the pulse arrives. The time T is called the **sample time.** The purpose of this pulse is to produce samples u(k) for $H_{zoh}(z)$ to work on and to produce samples of output y(k); thus, $H_{zoh}(z)$ can be realized as a discrete function.

The philosophy of the design is as follows: The discrete function $H_{zoh}(z)$ has to be determined, so that for a piecewise constant input to the continuous system H(s), the sampled output of the continuous system equals the discrete output. Supposing the signal u(k) represents a sample of the input signal this sample u(k) is held to produce a continuous signal $u_{hat}(t)$. Fig. 5 shows that the $u_{hat}(t)$ is held constant at u(k) during the interval kT to (k+1)T. This operation of holding $u_{hat}(t)$ constant for the sampling time is called **zero-order hold.**

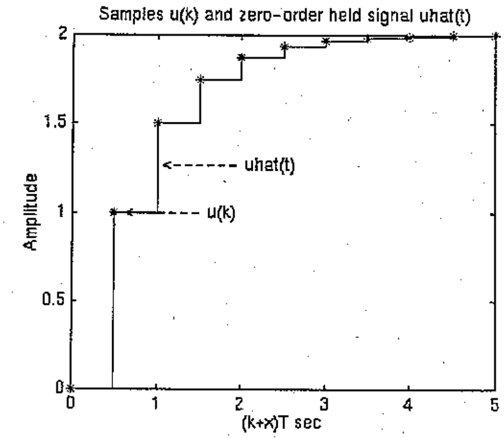


Fig.5

The zero-order held signal $u_{hat}(t)$ goes through H2(s) and A/D to produce the output y(k) that will be the piecewise same signal as if the continuous u(t) goes through H(s) to produce the continuous output y(t) (see Fig.6).

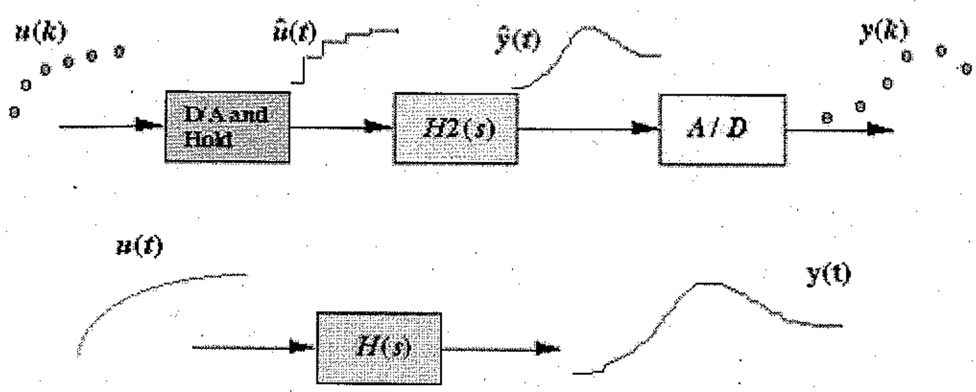


Fig.6

Now the schematic will be redrawn by replacing the continuous portion by $H_{zoh}(z)$ (Fig.7).

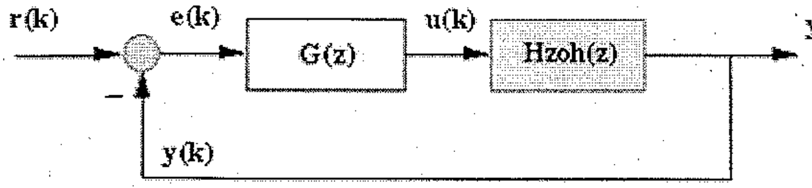


Fig. 7

By the rearrangement of H(s) to $H_{zoh}(z)$, the digital control system can be designed with only discrete functions.

To get $H_{zoh}(z)$ the Z-Transformation of the Laplace transfer function of the plant H(s) together with the zero-order hold has to be determined. The transfer function of the zero-order hold is given by $(1-e^{-sT})/s$. Using the definition of z ($z = e^{sT}$) $H_{zoh}(z)$ can be determined by

$$H_{zoh}(z) = \frac{z-1}{z} Z\{\frac{H(s)}{s}\}$$

The Z-Tranformation of $H(s)/s$ can be done by the use of transformation tables (see Appendix).

The system with either a continuous or a discrete PID-Controller has to be simulated the corresponding transfer functions must be considered:

- for a continuous PID-Controller the algorithm in time domain is

$$y(t) = K_P[e(t) + \frac{1}{T_n}\int_0^t e(\tau)d\tau + T_v\dot{e}(t)] = K_p e(t) + K_i\int_0^t e(\tau)d\tau + K_d\dot{e}(t)$$

- the Laplace transfer function of a PID-Controller is known as

$$G(s) = \frac{Y(s)}{E(s)} = K_p(1 + \frac{1}{sT_n} + sT_v) = K_p + \frac{K_i}{s} + sK_d$$

- for a digital PID-Controller the algorithm in time domain is

$$y_k = K_P[e_k + \frac{T}{T_n}\sum_{i=0}^{k-1} e_i + T_v\frac{e_k - e_{k-1}}{T}] = K_p e_k + K_i T\sum_{i=0}^{k-1} e_i + K_d\frac{e_k - e_{k-1}}{T}$$

$$y_k = K_P[e_k + \frac{T}{T_n}(\sum_{i=0}^{k} e_i - e_k) + \frac{T_v}{T}(e_k - e_{k-1})] = K_p e_k + K_i T(\sum_{i=0}^{k} e_i - e_k) + \frac{K_d}{T}(e_k - e_{k-1})$$

- the Z-Transformation of the digital PID-Controller is

$$G(z) = \frac{Y(z)}{E(z)} = K_P(1 + \frac{T}{T_n}\frac{z}{z-1} - \frac{T}{T_n} + \frac{T_v}{T}\frac{z-1}{z}) = K_P + K_i T\frac{z}{z-1} - K_i T + \frac{K_d}{T}\frac{z-1}{z}$$

A proportional controller (Kp) has the effect of reducing the rise time and will reduce, but never eliminate, the steady-state error. An integral control (Ki) has the effect of eliminating the steady-state error, but it may make the transient response worse. A derivative control (Kd) has the effect of increasing the stability of the system, reducing the overshoot, and improving the transient response. Effects of increasing each of controllers Kp, Kd, and Ki on a closed-loop system are summarized in Table 1.

CL RESPONSE	RISE TIME	OVERSHOOT	SETTLING TIME	S-S ERROR
Kp	Decrease	Increase	Small Change	Decrease
Ki	Decrease	Increase	Increase	Eliminate
Kd	Small Change	Decrease	Decrease	Small Change

Table 1

Note that these correlations may not be exactly accurate, because Kp, Ki, and Kd are dependent of each other. In fact, changing one of these variables can change the effect of the other two. For this reason, the Table 1 should only be used as a reference when you are determining the values for Ki, Kp and Kd.

For determination of optimal control parameters the PID tuning method by Chien, Hrones and Reswick can be used.

For that method the step response of the system without any controller has to be measured. The result should look like the Fig. 8. After drawing a tangent at the inflection point of the curve the system parameters K_S (process gain), T_g (time constant) and T_u (dead time) can be determined.

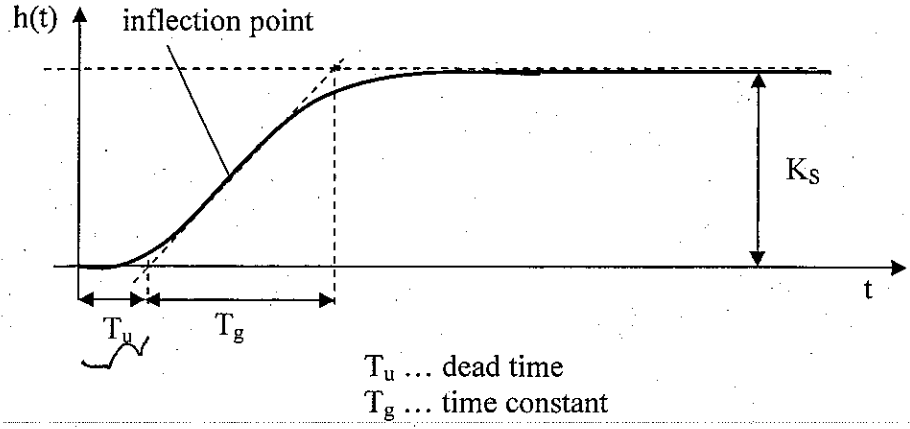


Fig.8

4'5 → 0'5
0'5×6'1 / 4'5

With these system parameters we are now able to determine optimal control parameters according to Table 2.

Controller-Type		Control Parameters
P	$K_p K_s$	$0{,}7 \frac{T_g}{T_u}$
PI	$K_p K_s$ T_n	$0{,}6 \frac{T_g}{T_u}$ $1 T_g$
PID	$K_p K_s$ T_n T_v	$0{,}95 \frac{T_g}{T_u}$ $1{,}35\ T_g$ $0{,}47\ T_u$

Table 2

As long as the sample rate is small enough, these parameters can be used for both continuous and digital controllers. In this case the digital controller design is called "quasi-continuous controller design". But the sample rate must fulfill the following condition:

$$T \leq \frac{1}{5} T_s$$

In this equation T is the sample rate and T_s is the smallest system time that occurs in the controlled system.

The second condition for a "quasi-continuous controller design" is that the sum of the time the microcontroller needs for the calculation of the control algorithm and the time for the D/A and A/D converters must be very small according to the sample rate. If this condition is not valid, we must add an additional dead time element to our system model to simulate this time consumption.

2. Experiment

Suppose we have a simple mass, spring, and damper problem.

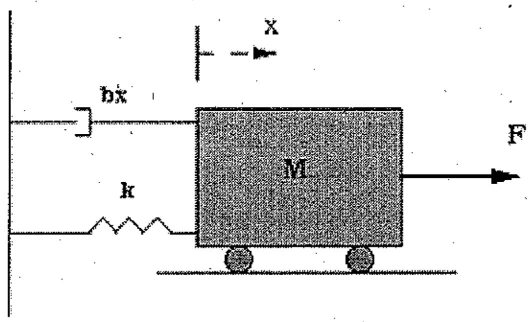


Fig. 9

The modeling equation of this system is

$$M\ddot{x} + b\dot{x} + kx = F$$

Taking the Laplace transform of this modeling equation

$$Ms^2X(s) + bsX(s) + kX(s) = F(s)$$

the transfer function between the displacement X(s) and the input F(s) then becomes

$$\frac{X(s)}{F(s)} = \frac{1}{Ms^2 + bs + k}$$

With M = 1kg, b = 10 Ns/m, k = 20 N/m and F(s) = 1 we will get

$$\frac{X(s)}{F(s)} = \frac{1}{s^2 + 10s + 20}$$

2.1 Open-loop step response

Start MATLAB SIMULINK and build a model with a PT_2 Block simulating the system described above. Add a step source and a scope so that the system looks like shown in Fig. 10. Set the step time of the source to "0" and the final value to "1". Set the stop time of the simulation parameters to "3" seconds and start the simulation.

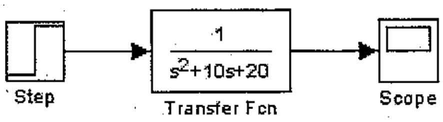


Fig. 10

After a double click on the Scope you can see the step response of the system. Click on the Autoscale-Button to get a better scaled figure. Print out the open-loop step response and determine the system parameters K_S, T_g and T_u from the figure. Use these values to determine the controller parameter for P-, PI- and PID-Controller for our system according to the tuning method by Chien, Hrones and Reswick.

2.2 Different continuous controllers

- Build a closed loop with the PT_2 Block and the different types of continuous controllers (P-, PI- and PID). For example the closed loop with a PID-Controller should look like Fig. 11.

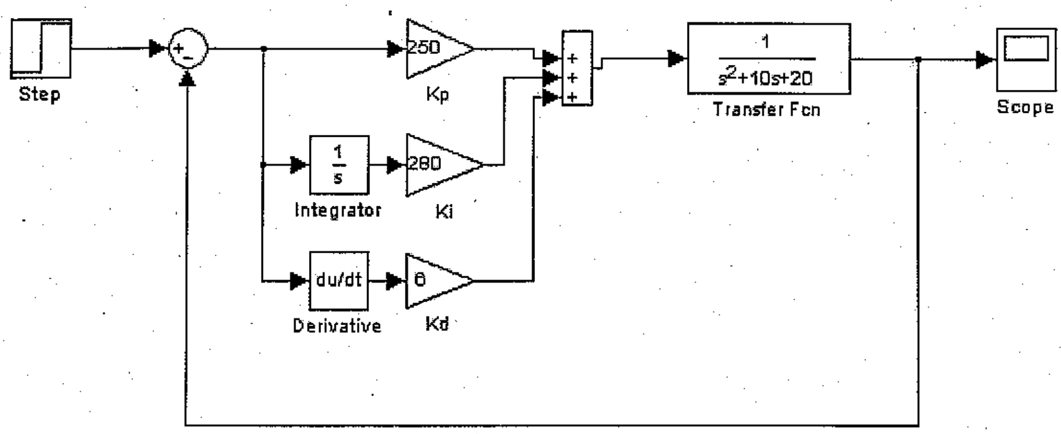


Fig. 11

- Verify the statements of the effects of rise time, overshoot, settling time and steady-state error made above. Vary the parameters and try to find the best parameters for all types of controllers. Print out the step responses of the system with the different types of continuous controllers and write down the used parameters.

2.3 Digital controller

Now exchange the continuous controller with a digital PID-controller. As the digital PID-controller should be used as a quasi-continuous controller, the sample time must be small enough.

- Calculate both time constants of our PT_2-system. Decide which one is smaller and determine the limit for a quasi-continuous controller design.
- In the MATLAB workspace define a constant T with a sample time that should be smaller than the determined limit.
- Build a closed loop with the system and a digital PID-controller that should look like Fig. 12. You must set the sample time of every discrete block (for example the discrete transfer functions and the zero-order hold) to the constant T you defined in the MATLAB workspace. By this it easy to change the sample rate for the whole system by changing the value for the constant T in the MATLAB workspace.

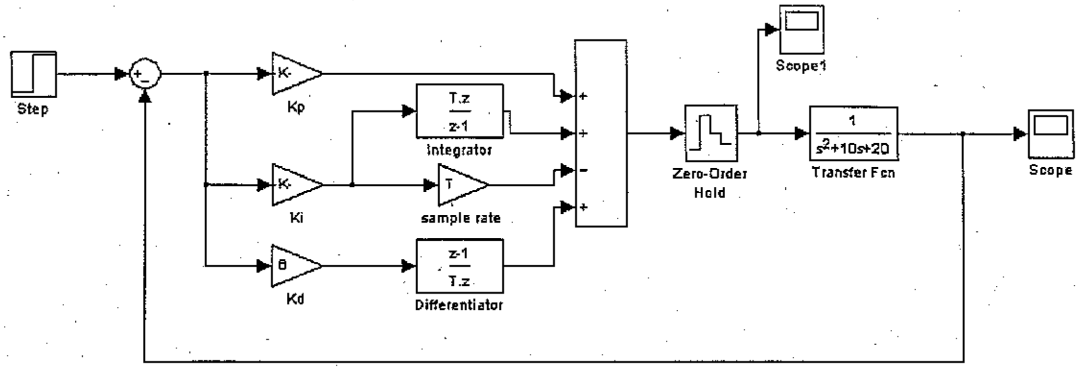


Fig. 12

**Hint: A real digital control system will contain a sample-block with an A/D-converter before the digital controller and a D/A-converter after the controller. As MATLAB only deals with digital values (even when it simulates continuous signals), these blocks are not necessary in this laboratory.*

- Test the system behavior with the different digital controller-types (P-, PI- and PID-controller) by setting the parameters of the discrete controllers to the same values you used for the continuous controllers.
- Test the system behavior for shorter and for greater sample times. Determine the sample time at which the system becomes unstable.
- Look at the input function of the controlled system (Scope1). This function is no longer a continuous function (as it was with a continuous controller) but it is a stair function.
- To simulate the time consumption of the microcontroller and the A/D- and D/A-converter add an additional transport delay block (dead time element) to the system as shown in Fig. 13. Test different delay times and different sample times and determine for which delay time the system becomes unstable.

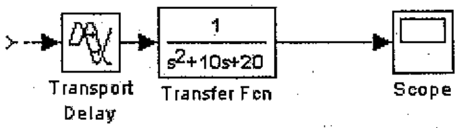


Fig. 13

Appendix:

Table of corresponding transformation pairs

Table: LAPLACE AND $\mathscr{Z}$-TRANSFORMATION PAIRS

	$F(s)$	$f(t)=\mathscr{L}^{-1}[F(s)]$	$f_{(kT)}=f_k$	$\mathscr{Z}[f_k]=F(z)$
1	$\frac{1}{s}$	1	1	$\frac{z}{z-1}$
2	$\frac{1}{s^2}$	t	kT	$\frac{Tz}{(z-1)^2}$
3	$\frac{1}{s^3}$	$\frac{1}{2}t^2$	$\frac{1}{2}(kT)^2$	$\frac{T^2}{2}\frac{z(z+1)}{(z-1)^3}$
4	$\frac{1}{s+a}$	e^{-at}	$c^k,\quad c=e^{-aT}$	$\frac{z}{z-c}$
5	$\frac{1}{(s+a)^2}$	te^{-at}	$(kT)c^k$, $c=e^{-aT}$	$\frac{cTz}{(z-c)^2}$
6	$\frac{1}{(s+a)^3}$	$\frac{1}{2}t^2e^{-at}$	$\frac{1}{2}(kT)^2c^k$, $c=e^{-aT}$	$\frac{T^2}{2}\frac{cz(z+c)}{(z-c)^3}$
7	$\frac{a}{s(s+a)}$	$1-e^{-at}$	$1-c^k$, $c=e^{-aT}$	$\frac{(1-c)z}{(z-1)(z-c)}$
8	$\frac{a}{s^2(s+a)}$	$\frac{1}{a}[at-(1-e^{-at})]$	$\frac{1}{a}[kaT-1+c^k]$, $c=e^{-aT}$	$\frac{Tz}{(z-1)^2}-\frac{(1-c)z}{a(z-1)(z-c)}$
9	$\frac{s}{(s+a)^2}$	$(1-at)e^{-at}$	$(1-kaT)c^k$, $c=e^{-aT}$	$\frac{z^2-c(1+aT)z}{(z-c)^2}$
10	$\frac{a^2}{s(s+a)^2}$	$1-(1+at)e^{-at}$	$1-(1+kaT)c^k$, $c=e^{-aT}$	$\frac{z}{z-1}-\frac{z}{z-c}-\frac{caTz}{(z-c)^2}$

(Cont.)

Table: (CONTINUED)

	$F(s)$	$f(t)=\mathscr{L}^{-1}[F(s)]$	$f_{(kT)}=f_k$	$\mathscr{Z}[f_k]=F(z)$
11	$\frac{b-a}{(s+a)(s+b)}$	$e^{-at}-e^{-bt}$	c^k-d^k, $c=e^{-aT}$, $d=e^{-bT}$	$\frac{(c-d)z}{(z-c)(z-d)}$
12	$\frac{(b-a)s}{(s+a)(s+b)}$	$-ae^{-at}+be^{-bt}$	$-ac^k+bd^k$, $c=e^{-aT}$, $d=e^{-bT}$	$\frac{(b-a)z^2-(bc-ad)z}{(z-c)(z-d)}$
13	$\frac{ab}{s(s+a)(s+b)}$	$1+\frac{be^{-at}-ae^{-bt}}{a-b}$	$1+\frac{bc^k-ad^k}{a-b}$, $c=e^{-aT}$, $d=e^{-bT}$	$\frac{z}{z-1}+\frac{b}{a-b}\frac{z}{z-c}-\frac{a}{a-b}\frac{z}{z-d}$
14	$\frac{\beta}{s^2+\beta^2}$	$\sin\beta t$	$\sin k\beta T$	$\frac{z\sin\beta T}{z^2-2z\cos\beta T+1}$
15	$\frac{s}{s^2+\beta^2}$	$\cos\beta t$	$\cos k\beta T$	$\frac{z^2-z\cos\beta T}{z^2-2z\cos\beta T+1}$
16	$\frac{\beta}{s^2-\beta^2}$	$\sinh\beta t$	$\sinh k\beta T$	$\frac{z\sinh\beta T}{z^2-2z\cosh\beta T+1}$
17	$\frac{s}{s^2-\beta^2}$	$\cosh\beta t$	$\cosh k\beta T$	$\frac{z^2-z\cosh\beta T}{z^2-2z\cosh\beta T+1}$
18	$\frac{s+\alpha}{(s+\alpha)^2+\beta^2}$	$e^{-at}\cos\beta t$	$c^k\cos k\beta T$, $c=e^{-aT}$	$\frac{z^2-cz\cos\beta T}{z^2-2cz\cos\beta T+c^2}$
19	$\frac{s+\alpha}{(s+a)^2+\left(\frac{\pi}{T}\right)^2}$	$e^{-at}\cos\frac{\pi t}{T}$	$(-c^k)$, $c=e^{-aT}$	$\frac{z}{z-c}$
20	$\frac{\beta}{(s+a)^2+\beta^2}$	$e^{-at}\sin\beta t$	$c^k\sin k\beta T$, $c=e^{-aT}$	$\frac{cz\sin\beta T}{z^2-2cz\cos\beta T+c^2}$

Experiment 2: Position Control with PV- and Dead-Beat Controller

1. Introduction

In this laboratory two different types of controllers for the position-control of a servo-motor should be designed. The first step of designing a controller is to develop the mathematical description of the controlled system. In this lab the system is an electrical motor. Therefore the electrical components of the motor should be examined first. A schematic circuit of these components can be seen in Figure 1.

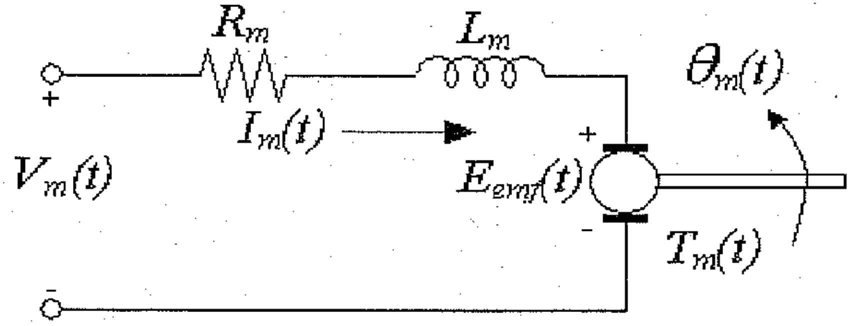


Figure 1 – *motor circuit in time-domain*

Using Kirchhoff's voltage law, the following equation can be written:

$$V_m - R_m I_m - L_m \frac{dI_m}{dt} - E_{emf} = 0 \qquad \textbf{[1.1]}$$

Since $L_m << R_m$ the motor inductance can be disregarded. Then the equation looks like following:

$$I_m = \frac{V_m - E_{emf}}{R_m} \qquad \textbf{[1.2]}$$

It should be known that the back emf created by the motor is proportional to the motor shaft velocity ω_m such that:

$$I_m = \frac{V_m - K_m \dot{\theta}_m}{R_m} \qquad (\dot{\theta}_m = \omega_m) \qquad \textbf{[1.3]}$$

Next the mechanical aspect of the motor should be examined. Therefore Newton's 2nd law of motion to the motor shaft has to be applied:

$$J_m \ddot{\theta}_m = T_m - \frac{T_l}{\eta_g K_g} \qquad [1.4]$$

In this equation $\frac{T_l}{\eta_g K_g}$ is the load torque seen thru the gears and η_g is the efficiency of the gearbox.

The 2nd law of motion at the load of the motor is like following:

$$J_l \ddot{\theta}_l = T_l - B_{eq} \dot{\theta}_l \qquad [1.5]$$

Beq is the viscous damping coefficient as seen at the output. Substituting **[1.4]** into **[1.5]**, it follows:

$$J_l \ddot{\theta}_l = \eta_g K_g T_m - \eta_g K_g J_m \ddot{\theta}_m - B_{eq} \dot{\theta}_l \qquad [1.6]$$

With $\theta_m = K_g \theta_l$ and $T_m = \eta_m K_t I_m$ (where η_m is the motor efficiency), **[1.6]** can be written as:

$$J_l \ddot{\theta}_l + \eta_g K_g^2 J_m \ddot{\theta}_l + B_{eq} \dot{\theta}_l = \eta_g \eta_m K_g K_t I_m \qquad [1.7]$$

Finally, combining the electrical and mechanical equations by substituting **[1.3]** into **[1.7]** yields to the desired transfer function:

$$\frac{\theta(s)}{V_m(s)} = \frac{\eta_g \eta_m K_t K_g}{J_{eq} R_m s^2 + (B_{eq} R_m + \eta_g \eta_m K_m K_t K_g^2) s} \qquad [1.8]$$

In this equation the abbreviation $J_{eq} = J_l + \eta_g J_m K_g^2$ is used. This can be interpreted as being the equivalent moment of inertia of the motor system seen at the output.

Using the additional abbreviations $K_0 = \eta_g \eta_m K_t K_g$, $K_1 = (B_{eq} R_m + \eta_g \eta_m K_m K_t K_g^2)$ and $K_2 = J_{eq} R_m$, this equation can be written as follows:

$$\frac{\theta(s)}{V_m(s)} = \frac{K_0}{K_2 s^2 + K_1 s} \qquad [1.9]$$

For this laboratory the values of constants used in **[1.8]** are following:

$$\eta_g = 0.9, \eta_m = 0.69, K_t = 0.0077, K_g = 70, B_{eq} = 0.004, J_{eq} = 0.0020842, R_m = 2.6 \qquad [1.10]$$

2. PV-Controller

This lab involves designing a PV controller for the servo plant. There is no need for an additional integral gain in the controller, as the main purpose of the integral gain is to reduce the steady state error by introducing a pole located at s = 0 in the open loop. Looking back at the derived equation **[1.9]**, it can be seen that the plant already has a pole at s = 0. For this reason, and for the sake of simplicity, the required specifications will use only a proportional (**P**) and velocity/derivative (**V**) controller gain.

In the classical sense, a PD controller would have the form: **C(s) =K_p + K_ds**. Placing this controller into the forward path would result in introducing an unwanted *'zero'* in the closed loop transfer function. As a result of the *'zero'*, the closed loop transfer function no longer fits equation **[2.2]** (see below) and it becomes increasingly difficult to design the controller to meet time specifications.

For this reason the approach of a state-feedback (**PV**) controller is used in this laboratory. This results in a closed loop transfer function of the form seen in equation **[2.2]**. The **PV** controller implemented in this lab can be seen in Figure 2 below and has the form:

$$V_m = -K_p(\theta - \theta_d) - K_v\dot{\theta} \qquad \textbf{[2.1]}$$

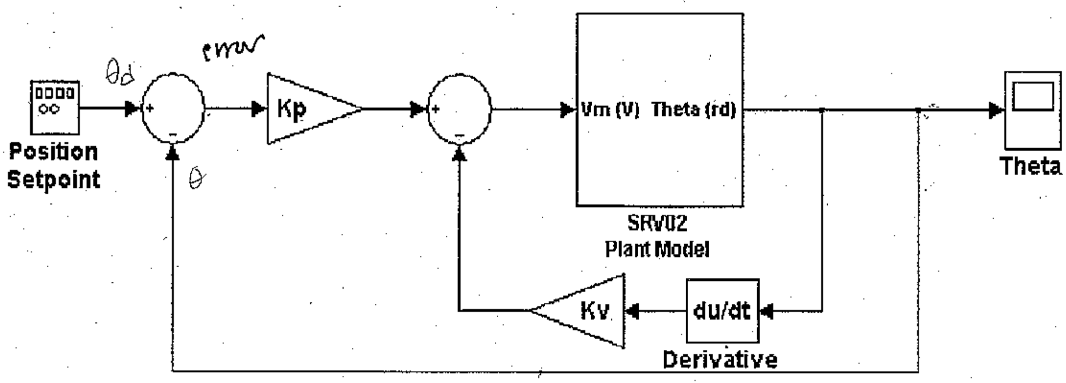


Figure 2 - PV Controller for the servo plant

The purpose of this lab is to design a controller using the 2nd order transfer function representation of the form:

$$G(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \qquad \textbf{[2.2]}$$

with characteristic equation:

$$s^2 + 2\zeta\omega_n s + \omega_n^2 = 0 \qquad \textbf{[2.3]}$$

Coincidentally, the **characteristic equations of the **PV** and **PD** controller closed loop transfer functions are **equal.** A PV controller in essence is a PD controller without the unwanted 'zero', allowing the designer to meet the required specifications using only the characteristic equation.*

2.1 For preparation for the laboratory experiments answer the following questions:

1) Obtain the transfer function of the closed loop model in Figure 2. Extract the characteristic equation and fit it to the form seen in equation **[2.3]**. Obtain two equations expressing ω_n and ζ as functions of $\mathbf{K_p}$ and $\mathbf{K_v}$ as these are the only two variables in your system. Using your newly obtained formulas and referring to your in-class notes, what changes to your response would you expect to see by varying the values of $\mathbf{K_p}$ and $\mathbf{K_v}$?

   (What happens to ω_n when you increase/decrease $\mathbf{K_p}$?)
   (What happens to ζ when you increase/decrease $\mathbf{K_v}$ and/or $\mathbf{K_p}$?)

   Keep your answers simple. (i.e. will ω_n and ζ increase or decrease?)

2) For the in-lab portion of this experiment, you are required to design a PV controller that will yield the following time requirements:
   - The Overshoot should be less than 5% ($\zeta \geq 0.707$).
   - The time to first peak should be 100ms ($\mathbf{T_p} = 0.100$).

   Using the formulas from Question 1, choose values for $\mathbf{K_p}$ and $\mathbf{K_v}$ to meet the requirements.

**Hint:*

$$T_p = \frac{\pi}{\omega_n \sqrt{1-\zeta^2}} \qquad \%OS = e^{\frac{-\zeta\pi}{\sqrt{1-\zeta^2}}} \qquad \textbf{[2.4]}$$

3. Dead-Beat Controller

In addition to the PV-Controller a dead-beat Controller for the servo plant should be designed in this lab. The dead-beat controller is an example of a control algorithm specifically worked out for sampled systems. It has no equivalent continuous time companion. Its main idea is to bring the system's response to its expected steady state value after a prefixed number of samples. This number of samples depends on the order and on the length of a possible existing dead time of the controlled system.

To perform a deadbeat design, the controller is constructed in the way that it cancels all the unwanted system dynamics, and introduces terms such that a specific closed-loop transfer function is obtained. This is dependent on the type of input to be followed (for example a step, ramp, etc.). Hence a deadbeat design is tuned for a particular input, and care needs to be taken when a different form of input signal is applied.

The principle design of a digital control loop can be seen in figure 3. In this Lab the input function $w_z(z)$ is the step function. That means:

$$w(t) = \sigma(t) \text{ or } w_z(z) = (1 - z^{-1})^{-1} \qquad \textbf{[3.1]}$$

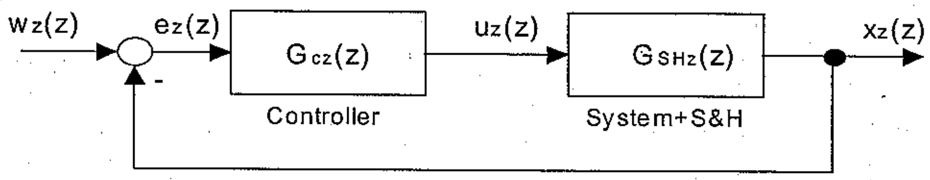


Figure 3 – Standard digital control loop

For the controller design the z-transformation of the controlled system is needed. Looking back at the derived s-transformation of the motor, the following Laplace-transfer function was found:

$$G(s) = \frac{\theta(s)}{V_m(s)} = \frac{K_0}{K_2 s^2 + K_1 s} \quad [3.2]$$

The z-transfer function $G_{SHz}(z)$ consists of the Laplace-transfer function of the motor plant together with the sample and hold block that is needed for digital control algorithms. The transfer function of a zero-order hold is given by $(1-e^{-sT})/s$. Using the definition of z ($z = e^{sT}$), $G_{SHz}(z)$ can be determined by

$$G_{SHz}(z) = \frac{z-1}{z} Z\{\frac{G(s)}{s}\} = \frac{z-1}{z} Z\left\{\frac{K_0}{s^2(K_2 s + K_1)}\right\} \quad [3.3]$$

Using the abbreviations $a = \frac{K_1}{K_2}$ and $b = \frac{K_0}{K_1}$ this equation can be written as:

$$G_{SHz}(z) = b\frac{z-1}{z} Z\left\{\frac{a}{s^2(s+a)}\right\} \quad [3.4]$$

With the following transformation pair the z-transfer function $G_{SHz}(z)$ can be determined:

$$F(s) = \frac{a}{s^2(s+a)} \Leftrightarrow F(z) = \frac{Tz}{(z-1)^2} - \frac{(1-c)z}{a(z-1)(z-c)}; c = e^{-aT} \quad [3.5]$$

Using this transformation pair yields to:

$$G_{SHz}(z) = b\frac{z-1}{z}\left[\frac{Tz}{(z-1)^2} - \frac{(1-c)z}{a(z-1)(z-c)}\right] = b\frac{aT(z-c)-(1-c)(z-1)}{a(z-1)(z-c)}$$

[3.6]

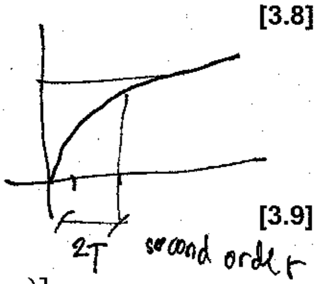
$$= \frac{z(bT + a^{-1}bc - a^{-1}b) + (a^{-1}b - bcT - a^{-1}bc)}{z^2 + z(-1-c) + c} = \frac{(bT + a^{-1}bc - a^{-1}b)z^{-1} + (a^{-1}b - bcT - a^{-1}bc)z^{-2}}{1 + (-1-c)z^{-1} + cz^{-2}}$$

With the use of the abbreviations $a_1 = (-1-c), a_2 = c, b_1 = (bT + a^{-1}bc - a^{-1}b)$ and $b_2 = (a^{-1}b - bcT - a^{-1}bc)$ $G_{SHz}(z)$ becomes to the following:

$$G_{SHz}(z) = \frac{b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} \qquad [3.7]$$

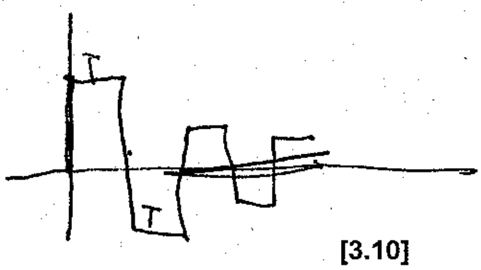
As this transformation function is of second order without any delay time a dead-beat controller should be able to bring the system's response to its expected steady state value after only two samples. That means that the expected output function of the closed loop with the step as input function should be of the form:

$$x_z(z) = \underbrace{x_0}_{=0} + x_1 z^{-1} + 1 \cdot z^{-2} + 1 \cdot z^{-3} + \ldots \qquad [3.8]$$



Thus the closed loop transfer function is:

$$\begin{aligned} G_{clz}(z) = \frac{x_z(z)}{w_z(z)} &= (1 - z^{-1})[x_1 z^{-1} + 1 \cdot (z^{-2} + z^{-3} + \ldots)] \qquad [3.9] \\ &= x_1 z^{-1} + 1 \cdot (z^{-2} + z^{-3} + \ldots) - [x_1 z^{-2} + 1 \cdot (z^{-3} + z^{-4} + \ldots)] \\ &= x_1 z^{-1} + (1 - x_1) z^{-2} \\ &= p_1 z^{-1} + p_2 z^{-2} = P(z) \end{aligned}$$



with

$$p_1 = x_1, p_2 = 1 - x_1, \sum_{i=1}^{2} p_i = 1 \qquad [3.10]$$

The output of the controller will be of the form:

$$u_z(z) = u_0 + u_1 z^{-1} + u_2 z^{-2} + u_2 z^{-3} + u_2 z^{-4} \ldots \qquad [3.11]$$

Thus the transfer function between the input of the system $w_z(z)$ and the output of the controller $u_z(z)$ is:

$$\begin{aligned} G_{cluz}(z) = \frac{u_z(z)}{w_z(z)} &= (1 - z^{-1})[u_0 + u_1 z^{-1} + u_2 z^{-2} + u_2 z^{-3} + u_2 z^{-3} + \ldots] \qquad [3.12] \\ &= u_0 + u_1 z^{-1} + u_2 z^{-2} + u_2 z^{-3} + \ldots - [u_0 z^{-1} + u_1 z^{-2} + u_2 z^{-3} + u_2 z^{-4} + \ldots] \\ &= u_0 + (u_1 - u_0) z^{-1} + (u_2 - u_1) z^{-2} \\ &= q_0 + q_1 z^{-1} + q_2 z^{-2} = Q(z) \end{aligned}$$

with

$$q_0 = u_0, q_1 = u_1 - u_0, q_2 = u_2 - u_1, \sum_{i=0}^{2} u_i = u_2 \qquad [3.13]$$

With these equations the transfer function of the controlled system can be written as:

$$G_{SHz}(z) = \frac{x_z(z)}{u_z(z)} = \frac{G_{clz}(z)w_z(z)}{G_{cluz}(z)w_z(z)} = \frac{P(z)}{Q(z)} \qquad \textbf{[3.14]}$$

The closed loop transfer function of the whole system is:

$$G_{clz}(z) = \frac{x_z(z)}{w_z(z)} = \frac{G_{cz}(z)G_{SHz}(z)}{1+G_{cz}(z)G_{SHz}(z)} \qquad \textbf{[3.15]}$$

The equation **[3.15]** can be written (with the use of **[3.9]** and **[3.14]**):

$$G_{cz}(z) = \frac{1}{G_{SHz}(z)}\frac{G_{clz}(z)}{1-G_{clz}(z)} = \frac{Q(z)}{P(z)}\frac{P(z)}{1-P(z)} = \frac{Q(z)}{1-P(z)} \qquad \textbf{[3.16]}$$

That means, that the transfer function of the dead-beat controller can be determined from the polynomials P(z) and Q(z). The coefficients of these polynomials can be determined from the coefficients of the controlled system. Equating **[3.7]** and **[3.14]** yields to:

$$\frac{b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} = \frac{p_1 z^{-1} + p_2 z^{-2}}{q_0(1 + \frac{q_1}{q_0} z^{-1} + \frac{q_2}{q_0} z^{-1})} \qquad \textbf{[3.17]}$$

Comparison of the coefficients yields to:

$$\begin{aligned} p_1 &= q_0 b_1, p_2 = q_0 b_2 \\ q_1 &= q_0 a_1, q_2 = q_0 a_2 \end{aligned} \qquad \textbf{[3.18]}$$

With

$$1 = \sum_{i=1}^{2} p_i = q_0 \sum_{i=1}^{2} b_i \qquad \textbf{[3.19]}$$

$G(s) = \frac{K_0}{K_2 s^2 + K_1 s}$

the parameter q_0 can be determined by

↓ z transf.

$$q_0 = \frac{1}{\sum_{i=1}^{2} b_i} \qquad \textbf{[3.20]}$$

3.1 For the laboratory experiments prepare the following equations:

- Determine equations for the calculation of q_0, q_1, q_2, p_1 and p_2 from the parameters of the controlled system and the sample time T (use the values given in **[1.10]**). These parameters are required for the transfer function of the dead-beat controller $G_{cz}(z) = \frac{Q(z)}{1-P(z)} = \frac{q_0 + q_1 z^{-1} + q_2 z^{-2}}{1 - p_1 z^{-1} + p_2 z^{-2}}$.

4. Laboratory Execution

This lab requires you to design a Proportional + Velocity (**PV**) controller to control the position of the load shaft with the following specifications:

1. The Overshoot should be less than 5% (0.707).
2. The time to first peak should be 100ms (**T_p** = 0.100).

4.1 Simulation of the Plant with a PV-Controller

In Simulink, open the model called *"s_position_pv.mdl"* in the directory *"C:\Advanced Control Labor\Exp2"*. This model includes the modelled plant (SRV-02 Plant Model), as well as the PV controller. **K_p** and **K_v** are both set by slider gains. Before you begin, you must run an M-File called *Setup_SRV02_Exp1.m"*. This file initializes all the motor parameters and gear ratios. Click on *Simulation->Start*, and bring up the *Simulated Position* scope. As you monitor the response, adjust **K_p** and **K_v** using the slider gains. Try a variety of combinations, and note the effects of varying each parameter.

- Make a table of system characteristics (ω_n and ζ) with respect to changes in **K_p** and **K_v**. (Hold one variable constant while adjusting the other).
- Does the system response react to how you had theorized in section **3.1**?

Now that you are familiar with the actions of each parameter, enter in the designed **K_p** and **K_v** that you had calculated to meet the system requirements.
**Note: the values should fall within the slider limits.*

- Does the response look like you had expected?
  What is your percent overshoot?
- Calculate your **T_p**. Does it match the requirements?

Hint: To get a better resolution when calculating* **T_p, *decrease the time range under the parameters option of the scope.*

If the simulated response is as expected, you can move on and implement your controller, if you are close to meeting the requirements, try fine-tuning your parameters to achieve the desired response. If the response is far from the specifications, you should re-iterate your design process and re-calculate your controller gains.

4.1 Implementation of the PV-Controller

After successfully simulating your controller and achieving your desired response, you are now ready to implement your controller and observe its effect on the physical plant. Open the Simulink model called *"q_position_pv.mdl"*.

The model has two identical closed loops; one is connected similar to the simulation block of the previous section, and the other loop has the actual plant in it. To better familiarize yourself with the model, it is suggested that you open both sub-systems to get a better idea of the systems as well as take note of the I/O connections.

Note: In place of a standard derivative block in the **PV controller, there is placed a derivative with a filter in order to eliminate any high frequencies from reaching the plant as high frequencies will in the long term damage the motor.*

Before running the model, you must set your final values of K_p and K_v in the MATLAB workspace (type it in MATLAB). You can now build the system using the *WinCon->Build* menu. You will see the model compile, and then you can use the WinCon Server to run the system (click on the start button).

Plot the *Measured Theta (deg)* as well as the *Position Setpoint* and the *Simulated Theta (deg)*. This is done by clicking on the scope button in WinCon and choosing *Measured Theta (deg)*. Now you must choose the *Position Setpoint* and the *Simulated Theta (deg)* signals thru the *Scope->File->Variables* menu.

- How does your actual plant response compare to the simulated response?
- Is there a discrepancy in the results? If so, why?
- Calculate your system T_p and **%OS**. Are the values what you had expected?

**You can calculate these parameters by saving these traces as an m-file and making our calculations in MATLAB . You could also make your calculations directly from the WinCon scope by zooming in on the signals. It is suggested to make these calculations thru MATLAB as this method will provide greater accuracy.*

4.3 Simulation of the Plant with a Dead-Beat Controller

In Simulink, open the model called *"s_position_dead_beat.mdl"* in the directory *"C:\Advanced Control Labor\Exp2"*. This model includes the modelled plant, as well as the dead-beat controller. You must run an M-File called *Setup_SRV02_Exp1.m"* to initialize the motor parameters.

Before you can start the model, you must set the desired sample time "T" and the calculated parameters of the dead-beat controller (q_0, q_1, q_2, p_1, p_2) in the MATLAB Workspace.

**Note: The parameters of the dead-beat controller are dependent on the sample time.*

Click on *Simulation->Start*, and bring up the *"output" and the "input"* scope. The output signal should reach the desired steady state value after only two sample times and without any overshoot.

The smaller you choose the sample time the faster the system reaches the final value. But the real motor system is designed for a maximum input of 10V. Therefore you should have in mind that the sample rate for the real system has got a lower limit. Try different sample times and the corresponding parameters of the dead-beat controller to find the optimal controller that combines a small sample time and a input for the motor smaller than 10V.

4.4 Implementation of the Dead-Beat Controller

Open the Simulink model called *"q_position_pv.mdl"*. The model has two identical closed loops; one is connected similar to the simulation block of the previous section, and the other loop has the actual plant in it.

Before running the model, you must set your final for the sample time and the parameters of the dead-beat controller in the MATLAB Workspace (type it in MATLAB). You can now build the system using the *WinCon->Build* menu. You will see the model compile, and then you can use the WinCon Server to run the system (click on the start button).

Plot the *output_system* and the *output_simulation* as well as the *input_system* and the *input_simulation*. This is done by clicking on the scope button in WinCon and choosing *output_system)*. Now you must choose the other signals thru the *Scope->File->Variables* menu.

- How does your actual plant response compare to the simulated response?
- Is there a discrepancy in the results? If so, why?
- What happens if you choose a dead-beat controller with a smaller sample time?
- Compare the different controller types. Describe the advantages and disadvantages of each controller.