MS150 Modular Servo Workshop - Teaching Manual

33-008-4M5

Feedback

- Computer Assisted Learning
- Electricity & Electronics
- Control & Instrumentation
- Process Control
- Mechatronics
- Robotics
- Telecommunications
- Electrical Power & Machines
- Test & Measurement

Technology Training for tomorrow's world

MS150 Modular Servo Workshop

Teaching Manual

33-008-4M5

Feedback

Feedback Instruments Ltd, Park Road, Crowborough, E. Sussex, TN6 2QR, UK.
Telephone: +44 (0) 1892 653322, Fax: +44 (0) 1892 663719.
email: feedback@fdbk.co.uk World Wide Web Homepage: http://www.fbk.com

Manual: 33-008-4M5 Ed02 990228 *Printed in England by FI Ltd, Crowborough*

Feedback Part No. 1160–330084M5

Notes

THE HEALTH AND SAFETY AT WORK ACT 1974

We are required under the Health and Safety at Work Act 1974, to make available to users of this equipment certain information regarding its safe use.

The equipment, when used in normal or prescribed applications within the parameters set for its mechanical and electrical performance, should not cause any danger or hazard to health or safety if normal engineering practices are observed and they are used in accordance with the instructions supplied.

If, in specific cases, circumstances exist in which a potential hazard may be brought about by careless or improper use, these will be pointed out and the necessary precautions emphasised.

While we provide the fullest possible user information relating to the proper use of this equipment, if there is any doubt whatsoever about any aspect, the user should contact the Product Safety Officer at Feedback Instruments Limited, Crowborough.

This equipment should not be used by inexperienced users unless they are under supervision.

We are required by European Directives to indicate on our equipment panels certain areas and warnings that require attention by the user. These have been indicated in the specified way by yellow labels with black printing, the meaning of any labels that may be fixed to the instrument are shown below:

CAUTION - RISK OF DANGER

Refer to accompanying documents

CAUTION - RISK OF ELECTRIC SHOCK

CAUTION - ELECTROSTATIC SENSITIVE DEVICE

PRODUCT IMPROVEMENTS

We maintain a policy of continuous product improvement by incorporating the latest developments and components into our equipment, even up to the time of dispatch.

All major changes are incorporated into up-dated editions of our manuals and this manual was believed to be correct at the time of printing. However, some product changes which do not affect the instructional capability of the equipment, may not be included until it is necessary to incorporate other significant changes.

COMPONENT REPLACEMENT

Where components are of a 'Safety Critical' nature, i.e. all components involved with the supply or carrying of voltages at supply potential or higher, these must be replaced with components of equal international safety approval in order to maintain full equipment safety.

In order to maintain compliance with international directives, all replacement components should be identical to those originally supplied.

Any component may be ordered direct from Feedback or its agents by quoting the following information:

1. Equipment type
2. Component value
3. Component reference
4. Equipment serial number

Components can often be replaced by alternatives available locally, however we cannot therefore guarantee continued performance either to published specification or compliance with international standards.

CE DECLARATION CONCERNING ELECTROMAGNETIC COMPATIBILITY

Should this equipment be used outside the classroom, laboratory study area or similar such place for which it is designed and sold then Feedback Instruments Ltd hereby states that conformity with the protection requirements of the European Community Electromagnetic Compatibility Directive (89/336/EEC) may be invalidated and could lead to prosecution.

This equipment, when operated in accordance with the supplied documentation, does not cause electromagnetic disturbance outside its immediate electromagnetic environment.

COPYRIGHT NOTICE

© Feedback Instruments Limited

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior permission of Feedback Instruments Limited.

ACKNOWLEDGEMENTS

Feedback Instruments Ltd acknowledge all trademarks.

IBM, IBM - PC are registered trademarks of International Business Machines.

MICROSOFT, WINDOWS 95, WINDOWS 3.1 are registered trademarks of Microsoft Corporation.

MATLAB is a registered trademark of Mathworks Inc.

TABLE OF CONTENTS

Contents

1. INTRODUCTION

1.1. KNOWLEDGE LEVEL REQUIRED

It is assumed that the student has some experience of using MATLAB version 5.* It is also assumed that the basic features of the *Modular Servo MS150* are known. It is suggested that the experiments described in [4] should be studied first.

A recommended progression through this teaching manual is::

- read the general background of *Assignment 1,*
- familiarise with the environment: do the experiments of *Assignment 1,*
- perform the identification experiment from *Assignment 2,*
- it is desirable but not mandatory that the next experiments be done in numerical order because each one builds on the ideas of the previous ones,
- start a new project: design your own real-time identification/control algorithms using the features of the MATLAB environment and functions from the *Modular Servo Toolbox.*

The toolbox functions and MATLAB functions can be used at different application levels. This manual uses the convention given in the following table.

application level 1	RTK*) communication functions, visualisation functions, miscellaneous functions, MATLAB functions
application level 2	function ***es*** (menu driven experiments)
application level 3	External Interface software

*) RTK - real-time kernel

Application level 1 means a direct use of the *Modular Servo Toolbox* functions controlling the information flow between the process sensors and the RTK. The functions used at this level configure a real time kernel, typical virtual controllers and an excitation source (See MSW - Reference Guide – 33-008-2M5).

Application Level 2: The function ***es*** initialises a menu-driven part of the toolbox.

Application Level 3: The External Interface software allows user-defined algorithms to be developed.

1.2. COMPUTER-AIDED CONTROL ALGORITHM DESIGN

The design of a control system requires many computations supported by simulation studies. For a given process the design is performed in several steps, which are represented in Figure 1-1. For the first step (definition of requirements and definition of a device technique) a rough model of the process is sufficient.

Modelling presupposes the knowledge of the essential physical laws of the process. For a digital servomechanism three classes of models can be distinguished:

- non-linear models
- continuous-time linear models
- linear discrete models.

The models of a lower class can be developed by making some assumptions concerning the model performance and the control system structure. Parameters of the models are derived on the basis of measured input and output, and other physical variables. This step of the design procedure is known as experimental modelling or identification.

When designing a controller, it is necessary to iterate through the methods a number of times to meet the essential requirements. Control algorithms are developed using an interactive dialogue with simulation techniques used to model the system behaviour using the variable values produced at each stage of the iteration. In addition various design parameters have to be chosen such as state variable weighting factors, control variable limits etc.. After these steps have been taken the performance of the control system can be measured.

If the assumptions about the control system are justified (i.e. the model is a good one), then the real-time part of the algorithm can be activated (or modified) and control experiments can be performed on a real plant to validate the design.

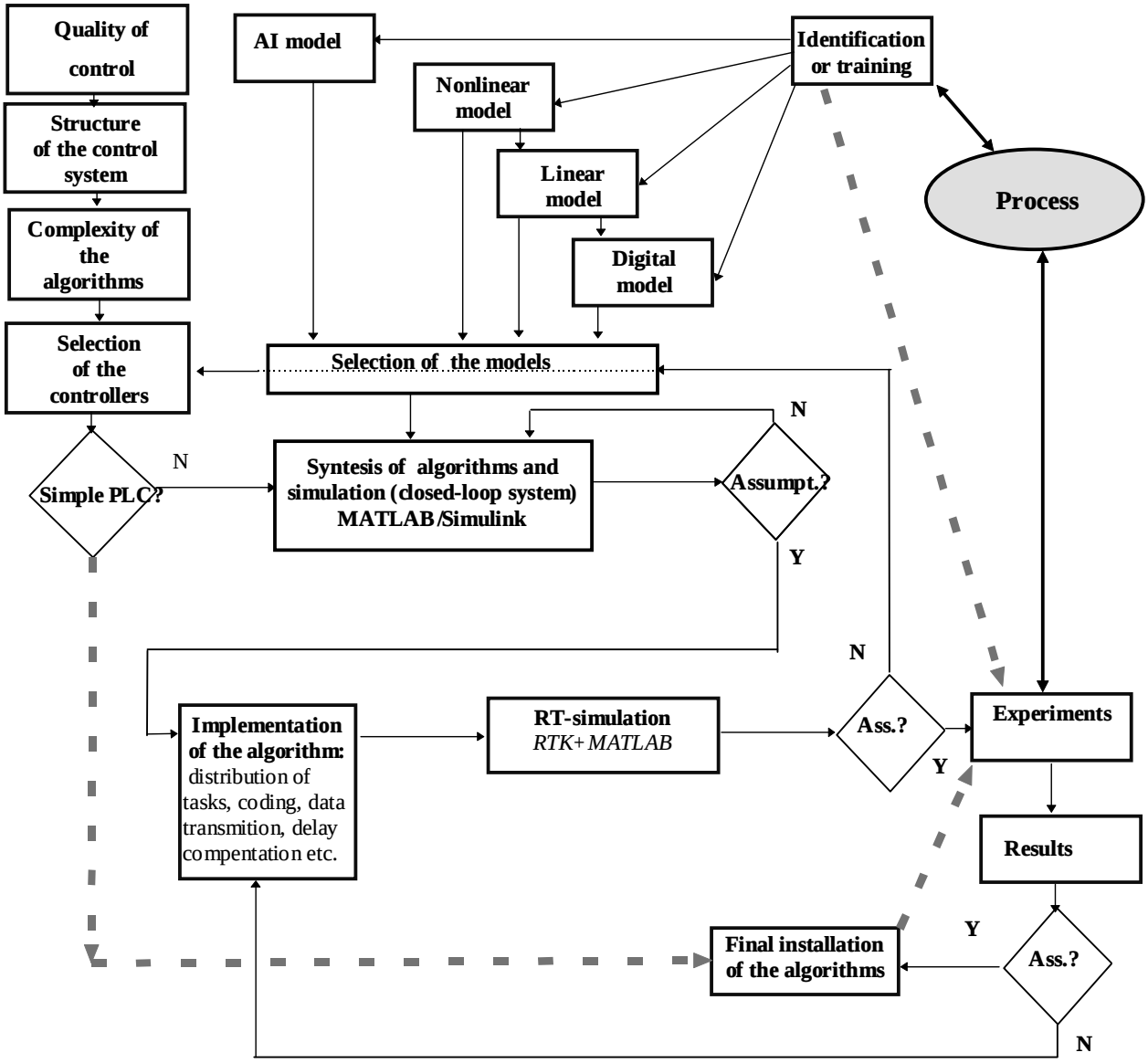


Figure 1-1: Design steps for a control system

2. ASSIGNMENT 1: REAL-TIME DIGITAL CONTROL

Content: The practicals in the Assignment 1 provide an introduction to the software and hardware environment [11], before more advanced control experiments are carried out.

2.1. COMPUTER CONTROLLED SYSTEM [1],[2]

A block diagram of a computer-controlled system is given in Figure 2-1. The system contains six blocks: the process, sensors (S), A/D and D/A converters, control algorithm, and a clock. The operation of the converters and of the control algorithm is controlled by the clock. The time between successive conversion of the signal to a digital form is called the sampling period (T_0). The clock supplies a pulse (or interrupt) every T_0 seconds, and the A/D converter sends a number to the computer each time the interrupt arrives. The control algorithm computes a control variable and sends a number to the D/A converter. It is assumed that the D/A converter holds the signal constant over the sampling period. Periodic sampling is normally used. It is possible to use different sampling periods for A/D and D/A converters.

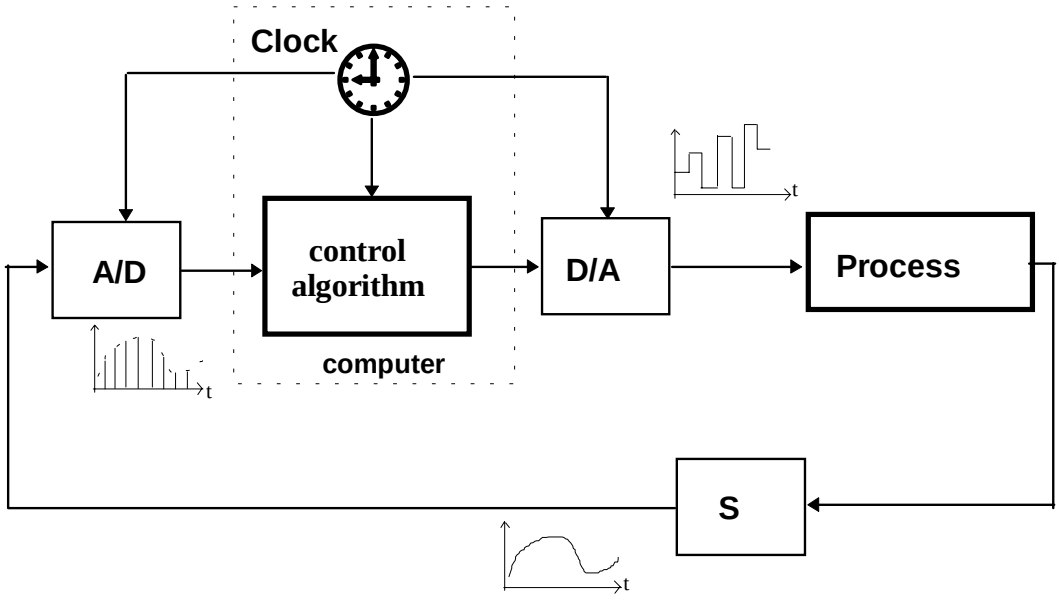


Figure 2-1: Computer controlled system

2.2. REAL-TIME CONTROL SYSTEMS [6]

The term "real-time" is often used but seldom defined. One possible definition is [6]:
The operating mode of a computer system in which the programs for the processing of data arriving from the outside are permanently ready, so that their results will be available within predetermined periods of time; the arrival times of the data can be randomly distributed or be already determined depending on the different applications.
A real-time software is usually structured around particular internal or external signals (events) into a set of tasks. Each task implements the processing required by a corresponding event. A task scheduler recognises the events and activates or suspends the tasks. In the simplest case, when all tasks require processing at the same frequency, a sequential organisation of the tasks can be implemented (Figure 2-2).

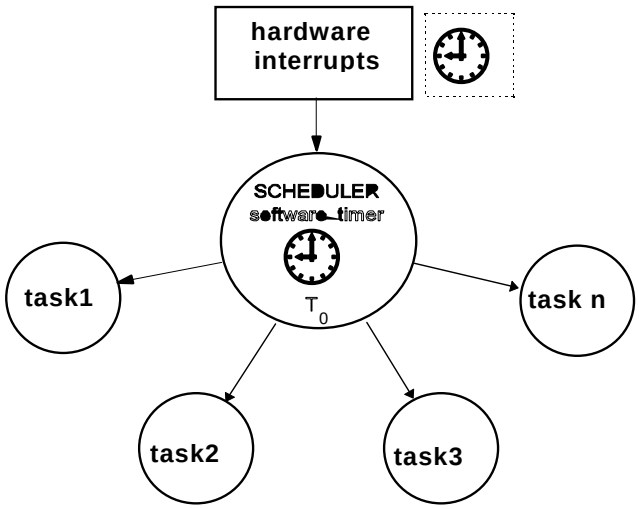


Figure 2-2: Sequential real-time system

A time frame of each task is fixed and it is assumed that the longest task job takes no longer then the period of time defined generated by software timer. A list of real-time tasks creating an environment suitable for developing basic digital controllers is given in Table 1.1.

For a low-cost system an *IBM-PC* compatible computers can be configured to run real-time tasks in the *Windows* environment. The real-time tasks are organised in the form of Real Time Kernel (RTK). Two levels of priority are defined:
highest priority for RTK tasks and lower priority for other *WINDOWS* applications (Figure 2-3).

The time available in between the processing of high priority tasks is used for processing lower priority tasks.

Table 1.1. Real-time tasks

data acquisition task	reads analogue and digital signals from all input channels.
filter task	performs filtration of the input data.
control task	calculates the control output
output task	transfers the controls to the output devices.
alarm task	starts special critical procedures
communication task	transmits the shared data between foreground and background (MATLAB and RTK)
self-diagnostic task	checks the operation of the scheduler .

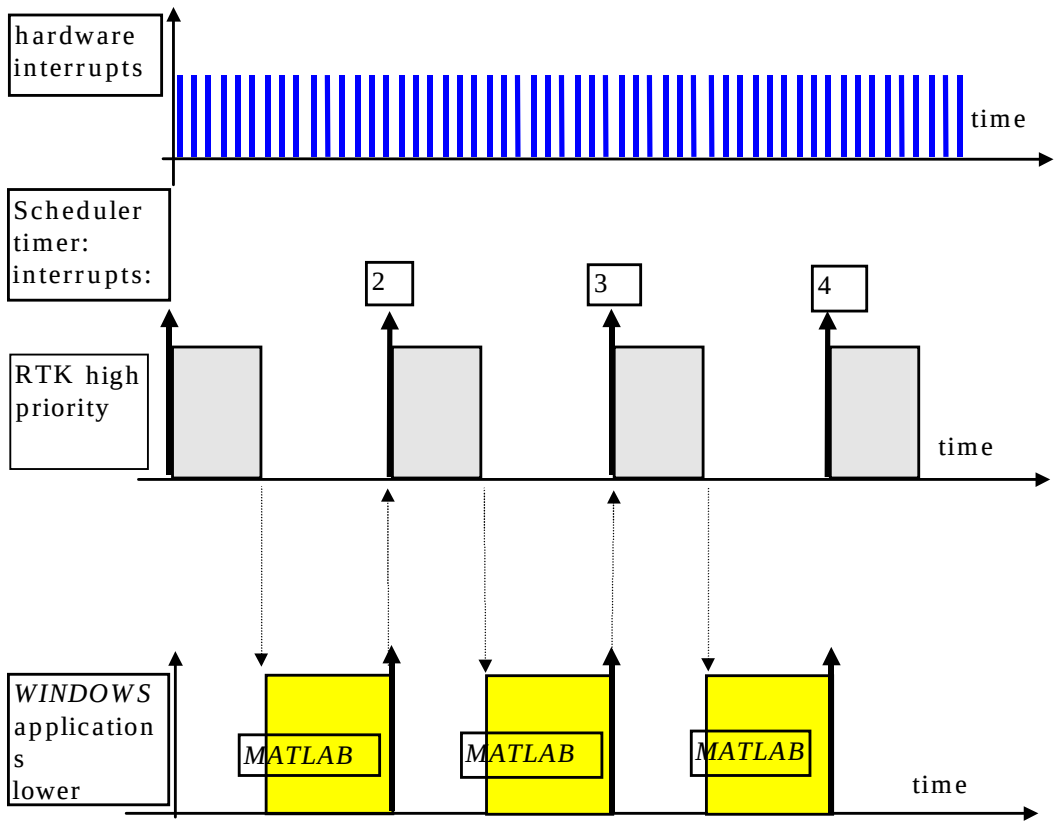


Figure 2-3: Multitasking in the *Windows* environment

2.3. SYSTEM CONFIGURATION [5]

An application of a general digital control system to the servomechanism is given in a block diagram form in Figure 2-4. Two process outputs are considered: the position **y1** and the speed **y2**. The process outputs can be measured as continuous signals (sensors **S_2)** digitised by analogue-to-digital converters (A/D), or a direct digital measurement technique can be used (sensors **S_1)**. The reference input (desired value) can be generated in a digital form using an internal excitation source or, alternatively, an external excitation generator (available from the Input Potentiometer IP150H) can be applied. Additionally, the *Simulink Generator* can be applied as a source of an excitation signal if a Simulink model is used. Two timers can be used to supply interrupts for the system: a basic clock which activates the periodic sampling of A/D converters, and an auxiliary clock which synchronises the computation of control outputs (**u**) and periodic digital - to - analogue **(D/A)** conversion. Details are given in Table 1.2.

Table 1.2. Control of the digital servomechanism

process	MS150 Actuator Unit
y1	position
y2	speed
S1	potentiometer for position, tachogenerator for speed (MS150 Actuator Unit)
S2	encoder for position
internal excitation source	RTK
external excitation source	Modular Servo MS150
A/D and D/A converters	data acquisition board
Simulink excitation source	From Mathsworks Inc. software

Figure 2-5 illustrates, how the blocks of the control system from Figure 2-4 are located in the hardware & software environment of the *Modular Servo Workshop.* Real time tasks and tasks scheduler are organised in the form of a real-time kernel (RTK). Real-Time Kernel supervises the set of real-time tasks creating a feedback control structure (Figure 2-5). Control tasks are embedded in the real-time kernel. Selection of the control algorithms and tuning of their parameters is made by means of the *communication interface* from the MATLAB environment. The communication interface is also used for configuration of real-time kernel (sampling period, excitation source, controller parameters etc.). A special memory buffer is created in order to enable process data flow between the real-time kernel and toolbox functions.

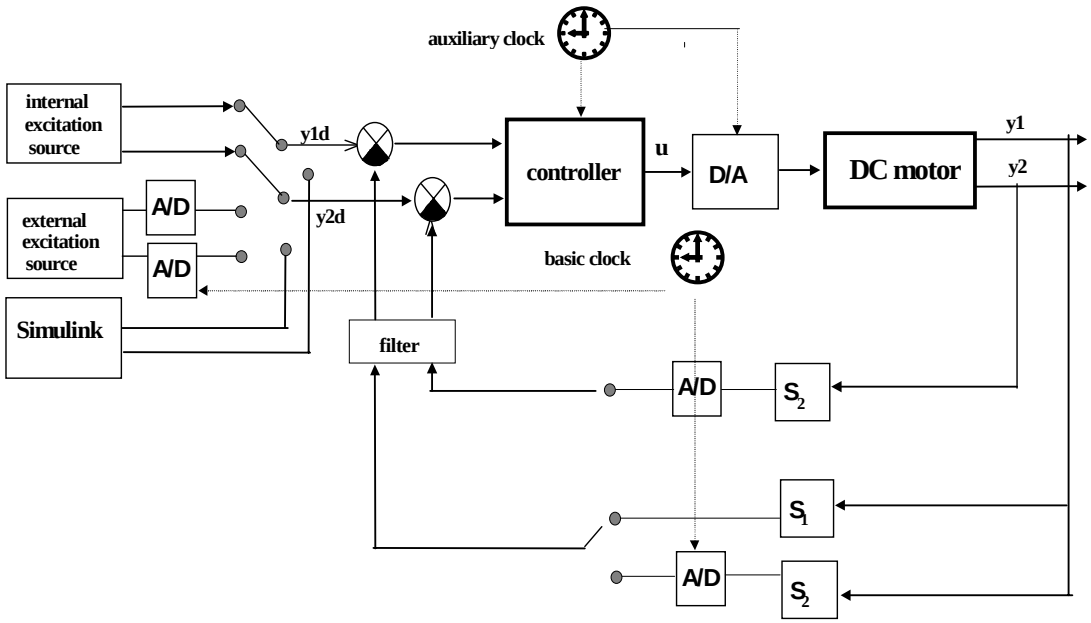


Figure 2-4: Digital control of the servomechanism (basic block diagram)

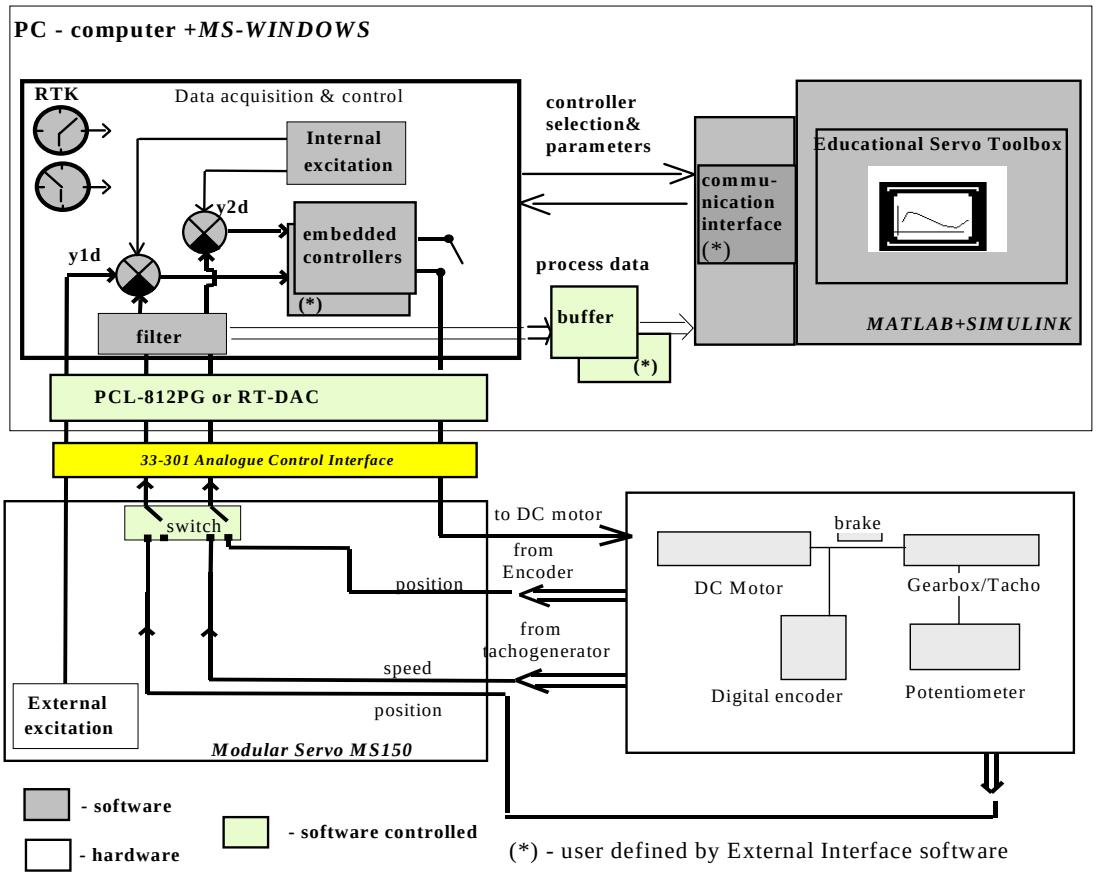


Figure 2-5: Digital control of the servomechanism (hardware and software configuration)

2.4. PRACTICAL 1.1: Data Transmission, Measurement and Filtration.

Objectives: In this practical the content of the buffer is read, data is displayed and the effects of data filtration are demonstrated.

Application level: 1.

The hardware and software configuration to realise this practical is given in Figure 2-6. The functions described in the MSW Reference Manual – 33-008-2M5 are used directly in this case.

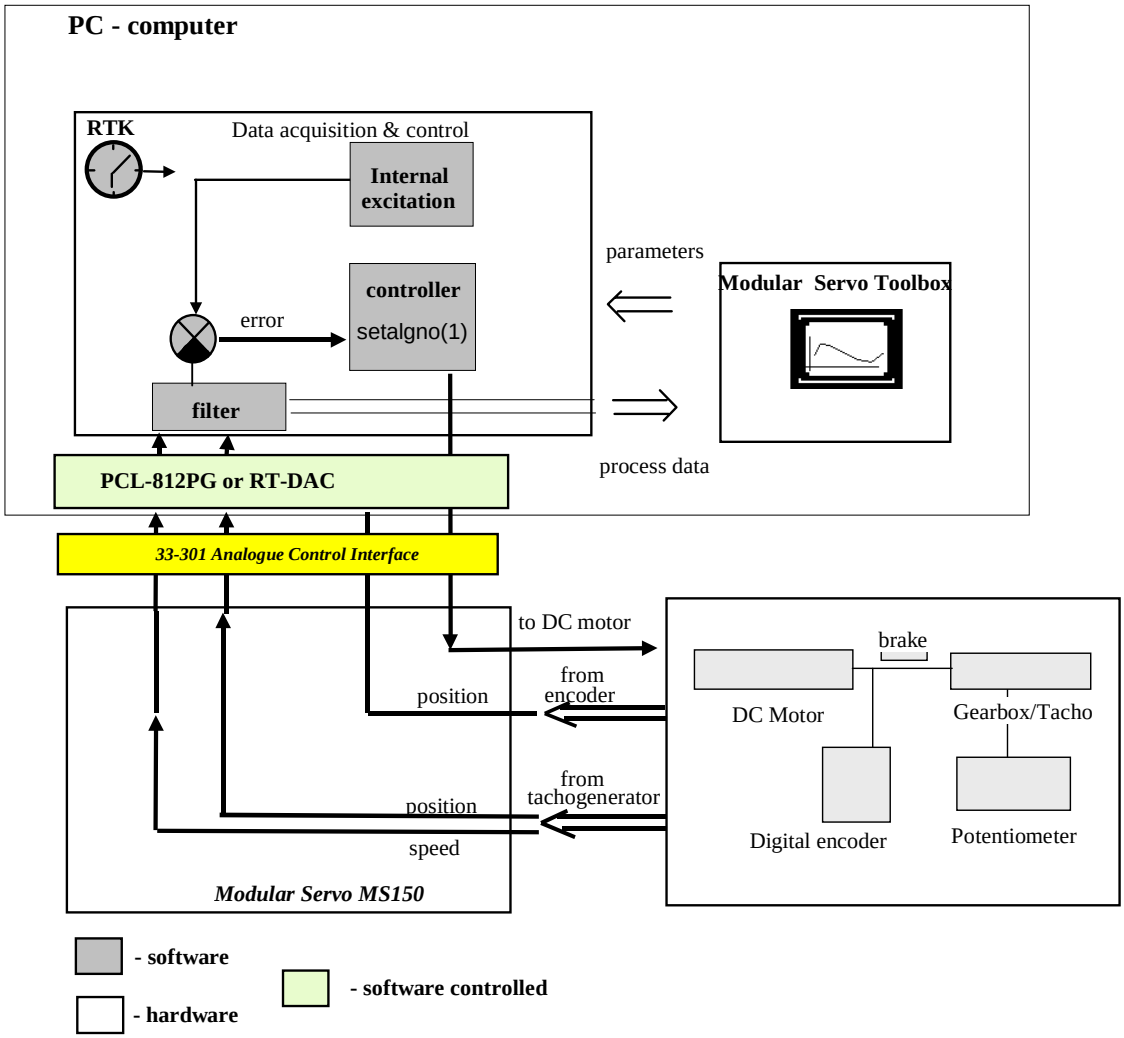


Figure 2-6: Hardware and software configuration for practical 1.1 (open-loop control)

When the MATLAB prompt appears type ***clear*** in order to remove items from the memory. Then invoke the following M file (see [11] for more details):

mservo2

> ***Note: check that the base address of the PCL 812 board, which is set in the code of this file, is the one your system is set to.***

This m-file causes the motor to rotate, forced by a sinusoidal function generated by the internal excitation source, in open-loop mode. The digital encoder is used to measure the position; the tachogenerator and A/D converter are used to measure the speed.

After this the data are in the workspace (in the *buffer* variable). The function *GetNoOfSamples* returns the number of samples in the buffer. Notice, that the function *GetHistory* cleans the buffer. It means that the number of samples in the buffer depends on the time elapsed from the last *GetHistory* **(**or *StartAcq***)** function call. This effect is illustrated in the Table 1.3.

The data are then plotted using a standard MATLAB ***plot*** function [7]:

Plots with the forms of Figure 2-7 will appear on the screen. Notice, that the length of the time axis depends on the number of samples available to the buffer, when *GetHistory* function was called.

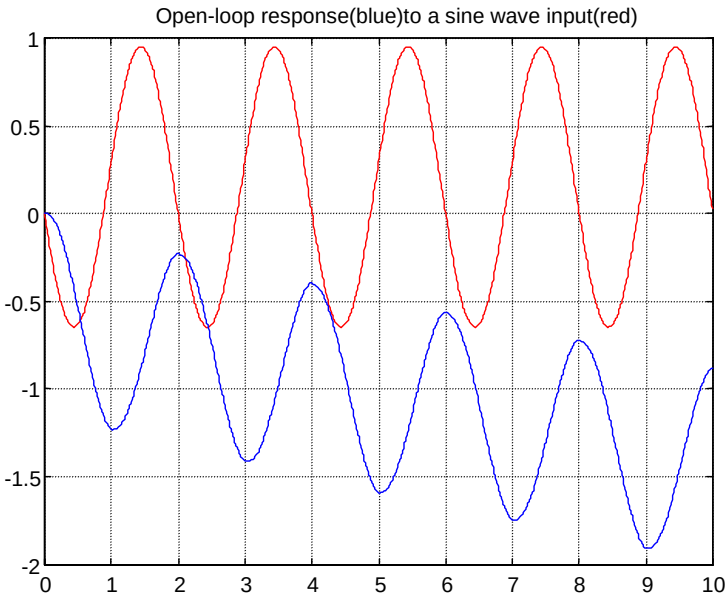


Figure 2-7: Open-loop experiment I - digital position measurement

Table 1.3. Example: relationships between MATLAB workspace and buffer contents.

		Number of samples in:	
time	**function**	**workspace**	**buffer**
t1	**StartAcq**	not defined	0 (cleared)
t2	**GetNoOfSamples**	not defined	n1=(time elapsed from t1)/ (sampling period)
t3	**GetHistory;**	n2=(time elapsed from t1)/ (sampling period)	0 (cleared)

Analogue position measurement

Close previous figure and type *mservo3* at the MATLAB prompt:

This m-file causes the motor to rotate, forced by a sinusoidal function generated by the internal excitation source, in open-loop mode.

This time the potentiometer is used to measure the position in this case instead of the digital encoder.

"Static" plots with the forms of Figure 2-7b will appear on the screen.

In this case some discontinuities of the position measurements from potentiometer can be observed. It is caused by limited range of potentiometer equal to 180 degrees. (Start with the 150K set at "12 o'clock".

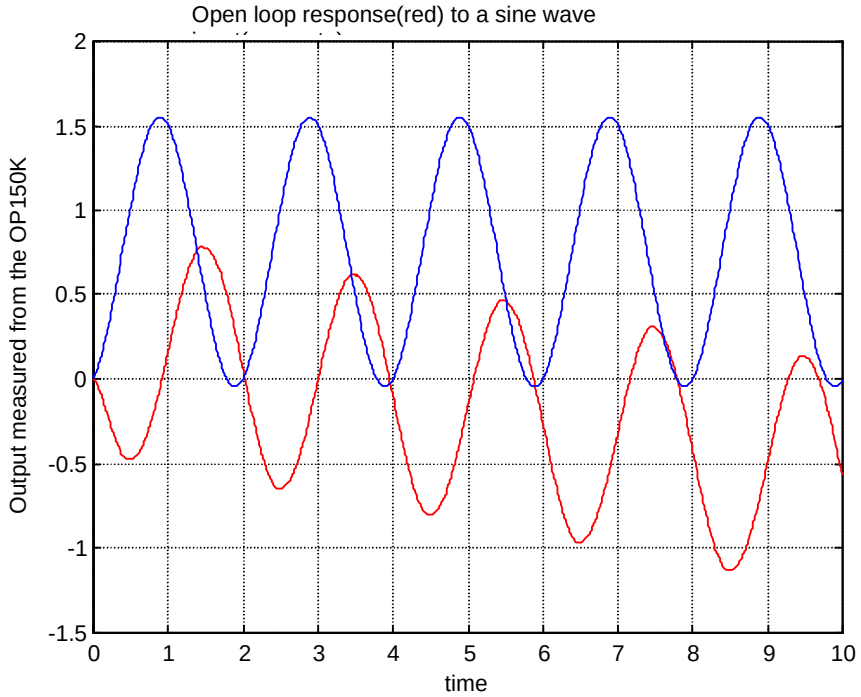


Figure 2-7b: Open-loop experiment II – analogue position measurement

2.5. PRACTICAL 1.2. Steady-State Characteristics of the DC Motor

Objectives: To investigate the steady-state characteristics of the DC-motor and to tune the power amplifier.

Application level: 2

In most cases small signal changes are presupposed for the design of control algorithms, so that the control system could be considered as being linear. As applications show, strong non-linearities in the control loop have to be taken into account when designing control systems. These include non-linear static characteristics such as hysteresis and saturation, which may occur in the following places: operational amplifiers, actuators, finite word length in A/D and D/A converters. Often the signal constraint first appears at the control variable:

$$u_{min} \leq u \leq u_{max}.$$

The purpose of this practical is:

- to estimate the control range within which the process behaves approximately linearly (u_{min},u_{max}).

The process is understood as a sequential connection of: D/A converter, power amplifier, DC motor, tachogenerator and A/D converter.

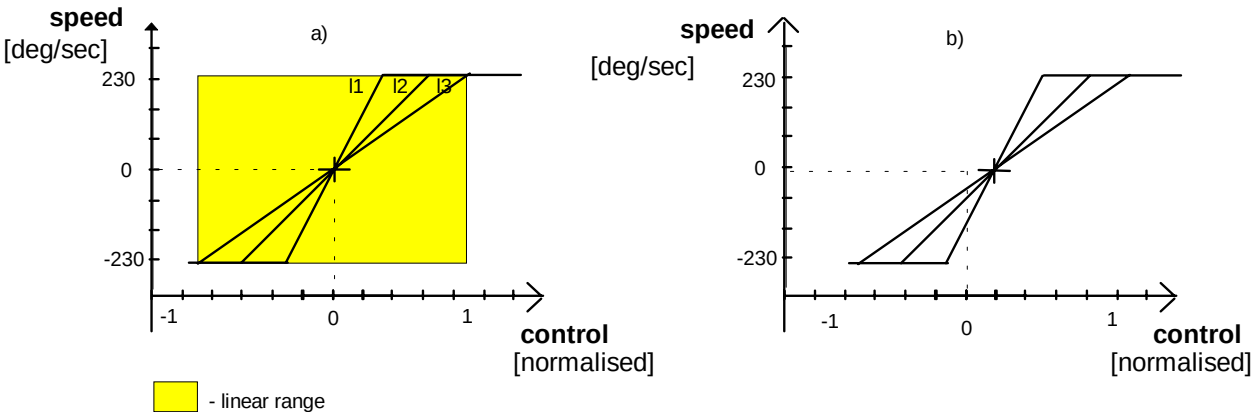


Figure 2-8: Steady-state characteristics of the process: a) symmetric, b) shifted.

l1,l2,l3 denotes the actual load

The hardware and software configuration for this practical is given in Figure 2-9. An internal excitation source is applied in order to generate desired control signals for the DC motor. The controller is in an open-loop mode (Algorithm no.1 in the RTK).

To start Practical 1.2 type ***es*** at the MATLAB prompt and select *Static characteristics of DC motor* in the *Main Control Window.* Then the number of branches of the characteristics, the number of test points per branch, and the ranges of the control signal must be set. Different branches of the characteristic will be generated then, if the load (magnetic brake position) is changed after each prompt.

Usually it is necessary to apply several positions of the magnetic brake in order to find the zero-point of the characteristics.

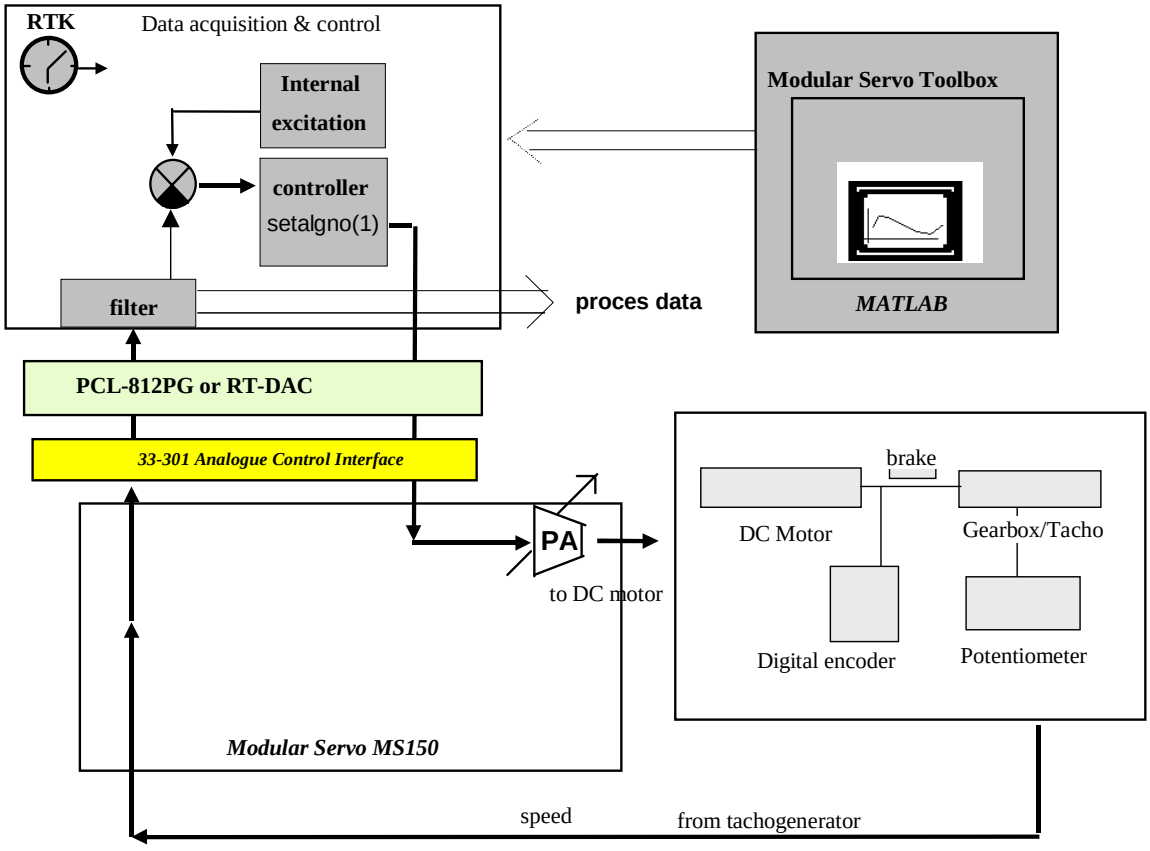


Figure 2-9: Hardware and software configuration for Practical 1.2 (open-loop control)

We know that the speed of an ideal motor is approximately proportional to the voltage applied to the armature in the linear portion of the motor characteristic curve. All methods of controlling the speed involve the use of this relationship.

$\dot{x} = kV$, where V is the voltage

2.6. PRACTICAL 1.3: Digital Control of the Servomechanism

Objectives In this practical a feedback loop is configured and simple tests of the servomechanism are performed.

Application level: 1

The hardware and software configuration to realise this practical is given in Figure 2-10. Proportional plus integral (PI) position controller and the external excitation source are applied.

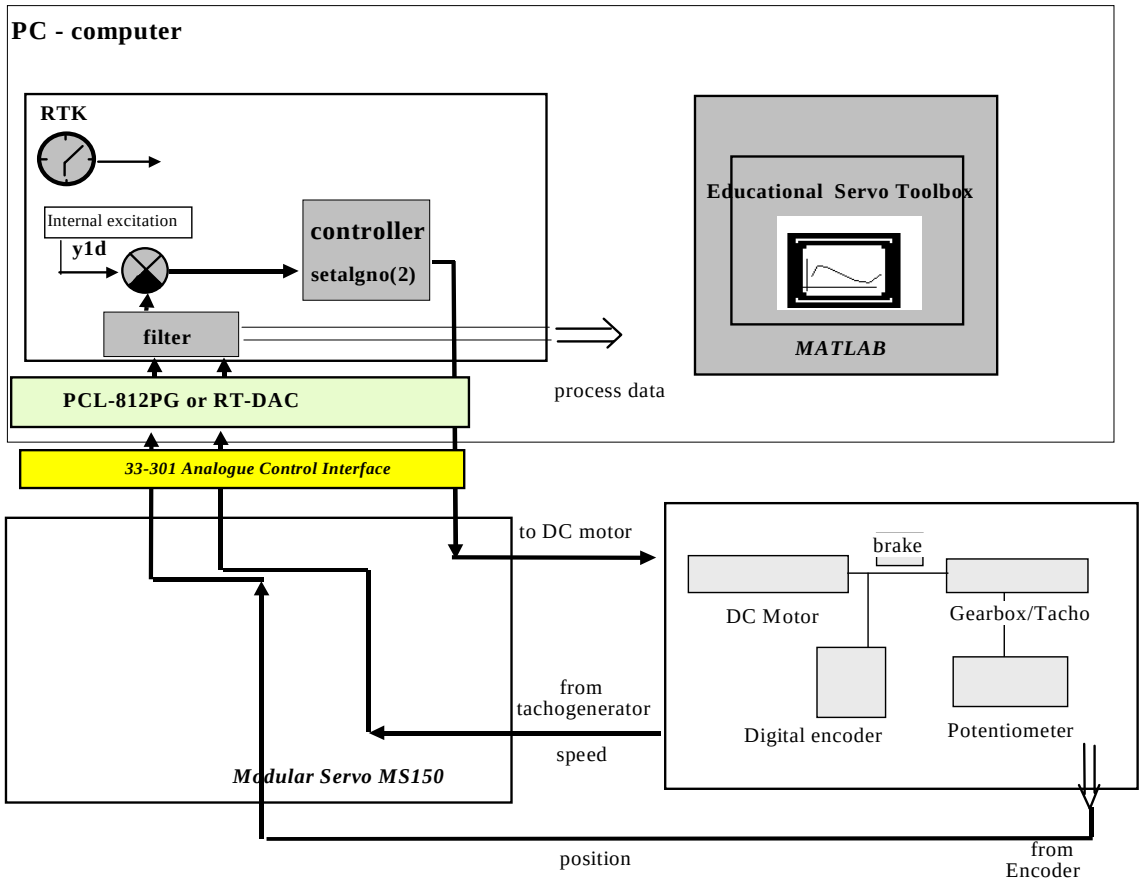


Figure 2-10: Hardware and software configuration for practical 1.3

Invoke the following command from the MATLAB command window

mservo5

Figure 2-11: Time response of the PI control system

3. ASSIGNMENT 2: Process Models - Open-Loop Characteristics

Content: The practicals in the Assignment 2 introduce models of the servomechanism and basic identification methods.

3.1. INTRODUCTION

The basic diagram of a servomechanism is shown in Figure 3-1. The action of servomechanism is to track a desired position despite the presence of disturbance input to the process and despite of errors in the sensors.

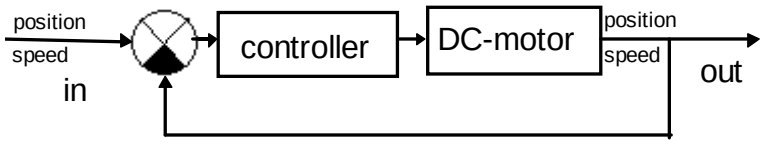


Figure 3-1: Basic block diagram of a servomechanism

If digital techniques are used to generate input and output signals, a computer-controlled servomechanism can be schematically described as in Figure 3-2.

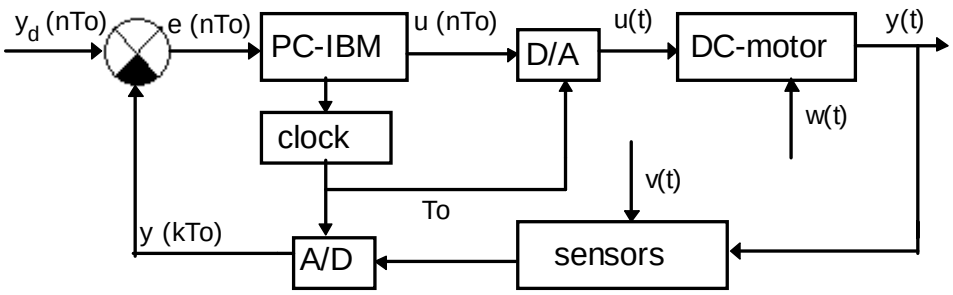


Figure 3-2: Standard configuration of a computer-controlled DC-motor

Notation of Figure 3-2			
y_d	reference inputs (desired values),	**w**	disturbance input to the process.,
u	control or actuator input signal,	**v**	disturbance or noise to the sensor,
y	controlled or output signal (e.g. position),	**To**	sampling time,
e	error,	**A/D**	analogue-to-digital converter,
		D/A	digital -to-analogue converter.

3.2. MODEL DEVELOPMENT

In a model, which is a formal description of a system, we wish to incorporate information necessary for the control of the process.

The construction of a model consists of two stages:
- choice of the class of models, and,
- selection of a particular model from this class (identification), see Figure 1-1.

First stage for our problem is simple as the linear model of a DC-motor is well known. Figure 3-3 shows a block diagram of a linear time-continuous model of the DC-motor.

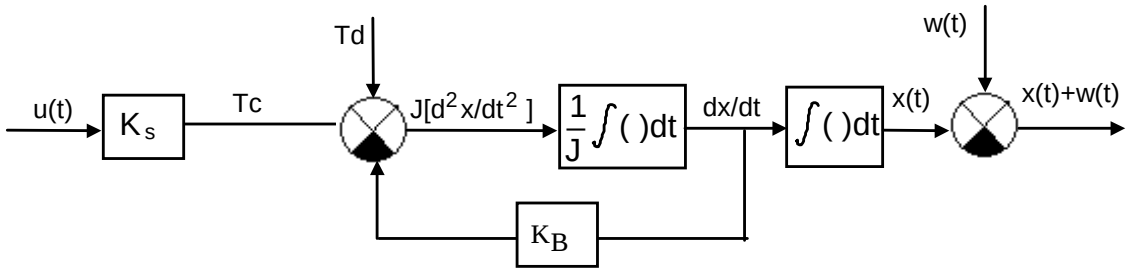


Figure 3-3: Block diagram of the DC-motor

The following definitions are used in Figure 3-3:

J	moment of inertia of the moving parts
K_B	damping coefficient consisting of mechanical friction and back EMF-effect
u	control voltage
x	position of shaft
Tc	drive-motor torque
Td	external torque
w	noise
Ks	parameter of the DC-motor

A DC-motor can be described by the second-order model with one integrator and one time constant. The time constant is due to the mechanical parts of the system and EMF-effect of the electrical parts. The following differential equation describes the dynamics of the DC-motor:

$$\frac{d^2x(t)}{dt^2} + \frac{1}{T_s}\frac{dx(t)}{dt} = \frac{K_S}{T_s}\,u(t)$$

where:

$$T_s = J/K_B$$

with initial conditions :

$$x(t=0) = x_0, \quad \frac{dx(t=0)}{dt} = \dot{x}_0$$

A typical step response of the DC-motor is given in Figure 3-4.

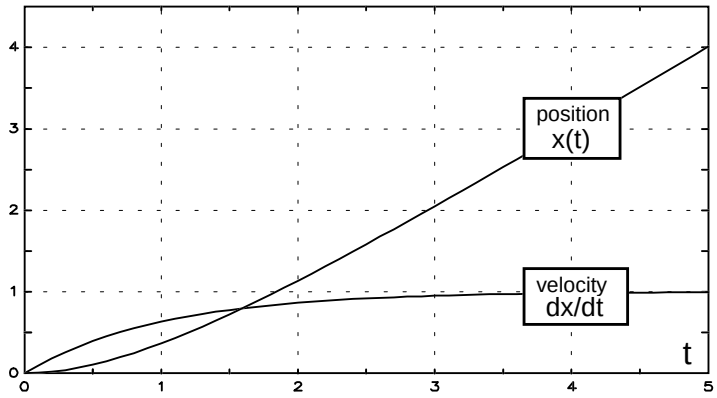


Figure 3-4: Step response of the DC-motor (time-domain)

3.2.1. State-space model [8]

Second-order differential equation of the DC-motor can be rewritten in the form two first-order differential equations or in the matrix form. This representation is called a state-space model:

$$\frac{dx}{dt} = Ax + Bu$$
$$y = Cx + Du$$
$$x(t=0) = x_0$$

A general form of the state space model is shown in Figure 3-5.

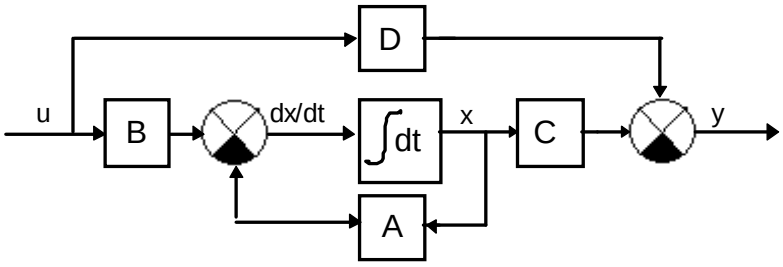


Figure 3-5: General state-space model

The general solution of the state-space equations is:

$$y(t) = C\left\{e^{At}x0 + \int_0^t e^{A(t-\tau)}Bu(\tau)d\tau\right\} + Du(t)$$

In the case of a DC-motor the velocity x_2 and the position x_1 of the motor shaft are introduced as the states variables. The state variables can be expressed in physical, A/D converter or electrical (voltage) units.

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad u = u, \quad y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

$$A = \begin{bmatrix} 0 & 1 \\ 0 & -1/T_s \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ \frac{K_s}{T_s} \end{bmatrix}, \quad C = \begin{bmatrix} c_1 & 0 \\ 0 & c_2 \end{bmatrix}, \quad D = 0$$

The system can be classified as a multivariable (SIMO) because it has two measurable states and one control variable. A typical step response of the DC-motor is given in Figure 3-4. Figure 3-7 shows a response of the DC-motor in the state-space domain. The parameters of the output matrix C depend on the type of sensors and transformations of the measurements in the RTK

In our case matrix C is identity matrix for measurements expressed in the physical units.

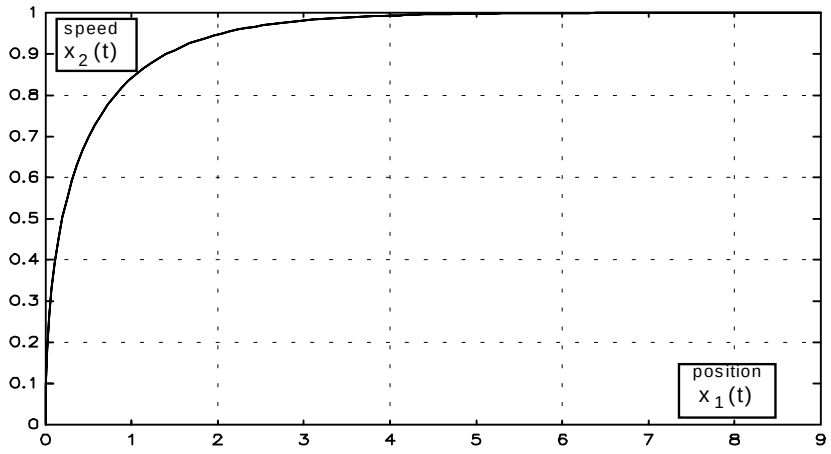


Figure 3-6: Step response of the DC-motor (phase plane)

For the DC-motor application described in this manual we have the situation in Figure 3-7:

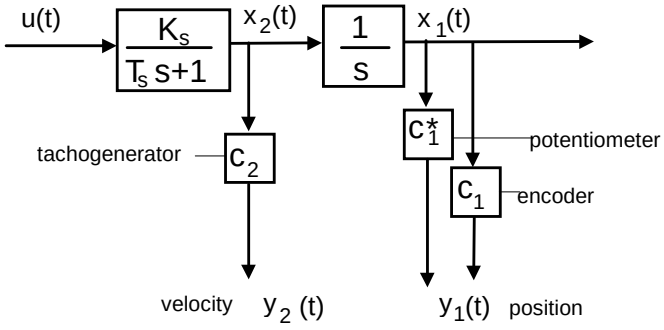


Figure 3-7: Model of the DC-motor and sensors

3.2.2. Transfer function model [8]

The transfer function of the DC-motor can be obtained from the differential equations of the process as:

$$G(s) = \frac{Y(s)}{U(s)} = \begin{bmatrix} G_1(s) \\ G_2(s) \end{bmatrix} = C\,(sI - A)^{-1} B = \begin{bmatrix} \frac{c_1 K_s}{s(T_s s+1)} \\ \frac{c_2 K_s}{T_s s+1} \end{bmatrix} = c_2 \begin{bmatrix} \frac{(c_1/c_2) K_s}{s(T_s s+1)} \\ \frac{K_s}{T_s s+1} \end{bmatrix}$$

The model has one pole at the origin and one pole on the negative real axis. The problem is to calculate the parameters K_S and T_S. This problem can be solved on the basis of measured inputs and outputs.

3.2.3. Discrete time models [1][2]

The use of digital computers to calculate a control action for a continuous, dynamic system introduces the fundamental operation of sampling. Samples are taken from the continuous, physical signals such as position or velocity, and these samples are used by the computer to calculate the control signals to be applied. Analog-to-digital converter (A/D) is a technical device which provides a discrete signals in process. Figure 3-8. shows an ideal sampler. Its role is to give a mathematical representation of the process of taking periodic samples r(kTo) from the continuous signal r(t). The ideal sampler is introduced for the purpose of process modelling: a sampling operation is represented as a impulse modulation.

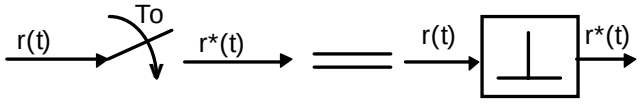


Figure 3-8: An ideal sampler

Thus, as in Figure 3-9, we model the output of the sampler as a string of Dirac's impulses:

$$r^*(t) = \sum_{k=-\infty}^{\infty} r(t)\,\delta\,(t - kTo), \quad \text{where } \delta(t - kT_0) = \begin{cases} 1 \text{ if } t = kT_0 \\ 0 \text{ otherwise} \end{cases}$$

The string of impulses r*(t) from the sampler depends only on discrete sample values r(kTo).

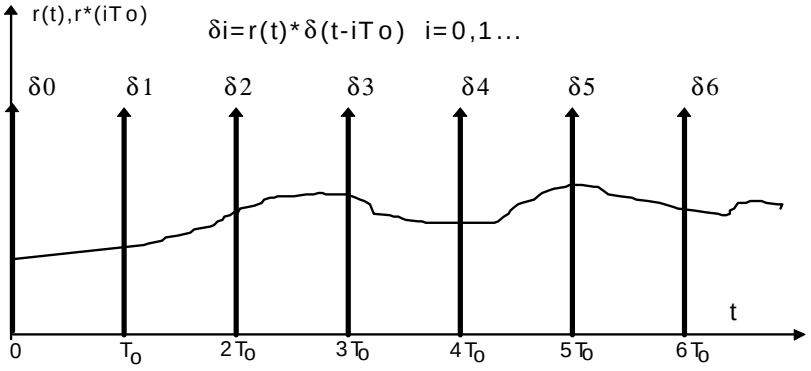


Figure 3-9: Ideal sampled signals.

The interface between discrete domain and continuous domain is modelled as a zero-order-hold. This device accepts samples r*(t) at t=kTo and holds its output at this value until next sample is sent (Figure 3-10 and Figure 3-11).

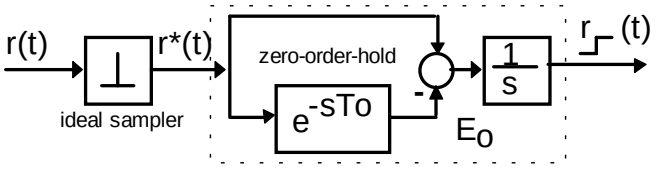


Figure 3-10: A real sampler (zero-order hold)

Figure 3-11: Real sampler output

If signals of the continuous plant are sampled, having a model of the sampling and hold operation a discrete model of the computer controlled DC-motor can be introduced (Figure 3-12). Each A/D converter is represented as an ideal sampler, and D/A converter is represented as a hold circuit, having transfer function from Figure 3-10.

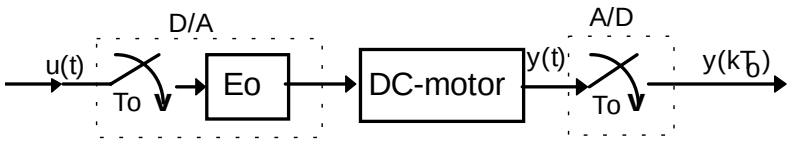


Figure 3-12: A discrete model of the continuous part of plant

Differential state-space equation of the continuous system:

$$\frac{dx}{dt} = Ax + Bu$$

$$y = Cx + Du$$

$$x(t = 0) = x_0$$

can be then substituted by an equivalent discrete state-space equation (assuming that interval between *n+1* and *n* time points is equal to T_0):

$$x[n+1] = A_d x[n] + B_d u[n]$$

$$y[n] = C_d x[n] + D_d u[n]$$

$$x[0] = x0$$

The flow diagram of the discrete state-space model is given in Figure 3-13.

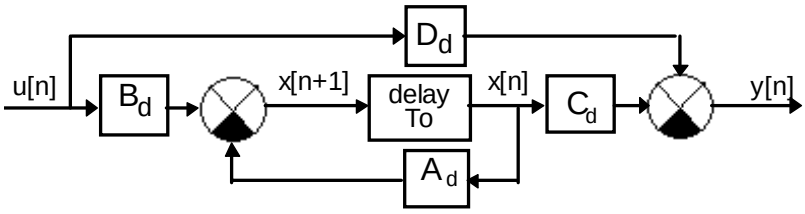


Figure 3-13: Flow diagram a discrete state-space system

The matrices A_d and B_d can be calculated from the continuous state-space model using the following relations [1]:

$$A_d = e^{A T_0} \qquad C_d = C$$

$$B_d = \left[\int_0^{T_0} e^{A t} dt \right] \qquad D_d = D$$

For the discrete model of the DC motor matrices A_d and B_d are in the form:

$$A_d = \begin{bmatrix} 1 & T_s\left(1 - e^{-\frac{T_0}{T_s}}\right) \\ 0 & e^{-\frac{T_0}{T_s}} \end{bmatrix} \qquad B_d = \begin{bmatrix} K_s\left[T_0 - T_s\left(1 - e^{-\frac{T_0}{T_s}}\right)\right] \\ K_s\left(1 - e^{-\frac{T_0}{T_s}}\right) \end{bmatrix}$$

The general solution of the discrete state-space equation is:

$$y[n] = C_d\left\{A_d^n x0 + \sum_{i=0}^{n-1} A_d^{n-i-1} B_d u[i]\right\} + D_d u[n]$$

A discrete time transfer function can be obtained from the relation:

$$G_d(z) = \frac{Y(z)}{U(z)} = C(zI - A_d)^{-1} B_d + D_d$$

where I is an identity matrix, and z is a shift operator.

For the DC-motor a discrete transfer function is given by (C_d = I, D_d = 0):

$$G_d(z) = K_S\begin{bmatrix} \frac{z[T_0 + T_s(a-1)] + [T_s - a(T_0 + T_s)]}{(z-1)(z-a)} \\ \frac{1-a}{z-a} \end{bmatrix} \quad \text{where:} \quad a = e^{-\frac{T_0}{T_s}}$$

3.2.4. Selection of the sampling period

Shannon's theorem assumes that the spectrum of the signal to be sampled is cut off vertically at ω_s. In practise, there are always frequencies greater than ω_s present in the signal. These differences from the ideal mean that a shorter sampling period needs to be used. One recommended choice is:

$T_0 \leq \frac{\Pi}{10\omega_b}$., where ω_b is a closed-loop bandwidth

Shannon's theorem applies both to open loop and closed loop systems .The sampling period should be chosen in relation to the desired behaviour of the closed-loop system. The sampling period can be related to

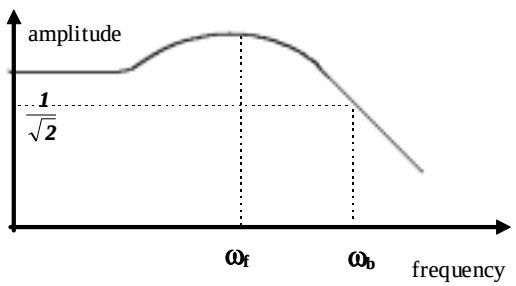
- eigenfrequency
- rise time
- damped frequency

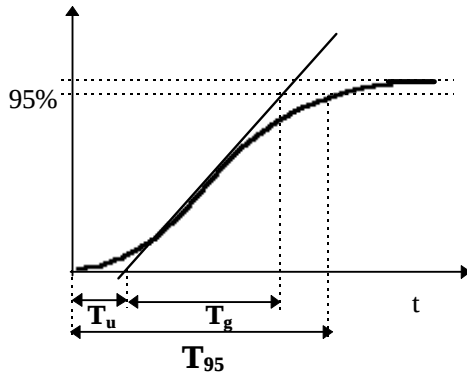
The methods of sampling period selection are given in Table 2.1.

For example, if we want the system to track the reference signal up to a certain closed-loop bandwidth ω_b, it follows that the signal will have spectral content up to that frequency. Notice the distinction between the closed-loop bandwidth, and the frequencies in the open-loop model because these frequencies can be quite different.

Table 2.1. Summary of rules for determining the sampling time.

Criteria to determine the sample time	Determination of the sample time	Remarks
Shannon	$T_0 \le \frac{\pi}{\omega_b}$	
eigenfrequency of a closed-loop system	$T_0 = (1/8 - 1/16)\frac{1}{\omega_f}$	Iserman [3]
95 % settling time	$T_0 = (1/6 - 1/12)T_{95}$	Iserman [3]
delay time	$T_0 = (1.2 - 0.35)T_u$ $T_0 = (0.35 - 0.22)T_u$	$0.1 \le T_u / T \le 1.0$ $1 \le T_u / T \le 10$
rise time (open and closed-loop)	$T_0 = (0.5 - 0.25)T_g$	Astrom [1]
oscillatory system	$T_0 = \frac{2\pi}{20\omega\sqrt{1-\xi^2}}$	Astrom [1]
PID derivative time T_D	$T_0 = (0.1 - 0.5)T_D$	Astrom [1]





3.2.5. Non-linear model (saturating components)

The models introduced above are all based on the assumption that the process can be described by linear model. All real physical plants are non-linear, in particular all outputs and inputs are limited (saturated). Figure 3-14 the shows main saturating components for a computer-controlled servomechanism.

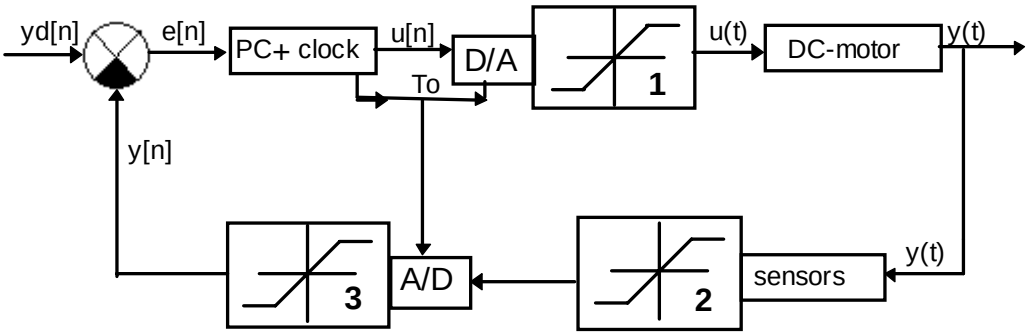


Figure 3-14: Saturating components in the PC -controlled DC-motor

Block 1 symbolises that the control signal u(t) is limited: output voltage from the power amplifier is limited.

Block 2 symbolises that outputs from the sensors are limited. Large measured values may saturate.

Block 3 symbolises that the converters D/A and A/D have limited length of digital register. The non-linearities shown in Figure 3-14 are important if large changes of control signals occur. A designer must take into account above facts, and understand why the outputs of the linear model and of the real plant differs for some ranges of the control signal.

3.3. FREQUENCY ANALYSIS

The most important characteristic of the process is its frequency response. Figure 3-15 shows the experiment desired to obtain a frequency characteristics of the process. If a sinusoidal signal of frequency ω is applied to a stable, linear, continuous system, then the steady-state response has amplitude $B(\omega)$ and phase $\varphi(\omega)$ in relation to the input signal.

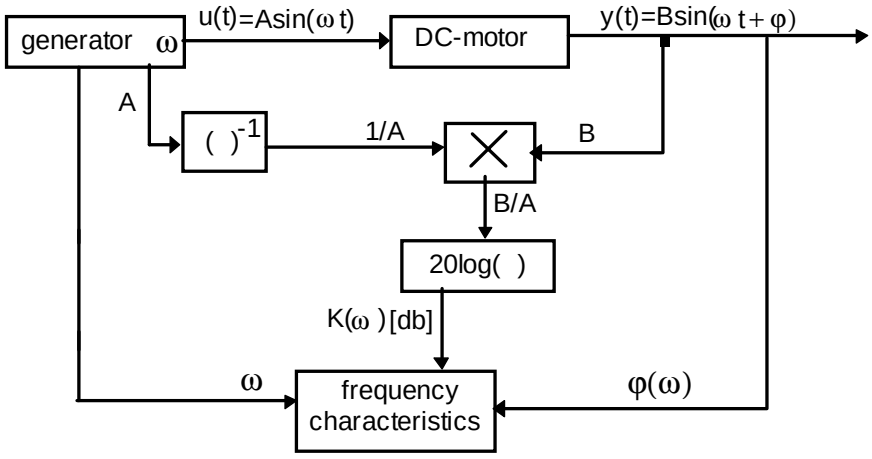


Figure 3-15: Experimental measurement of a frequency response

Figure 3-16 shows the frequency characteristics of the "standard" DC-motor:

- gain in dB
- phase in deg.

The "Standard" (K_s=1 and T_s=1) transfer function of the DC-motor is assumed as:

$$G_{stand} = \frac{1}{s\,(s+1)}$$

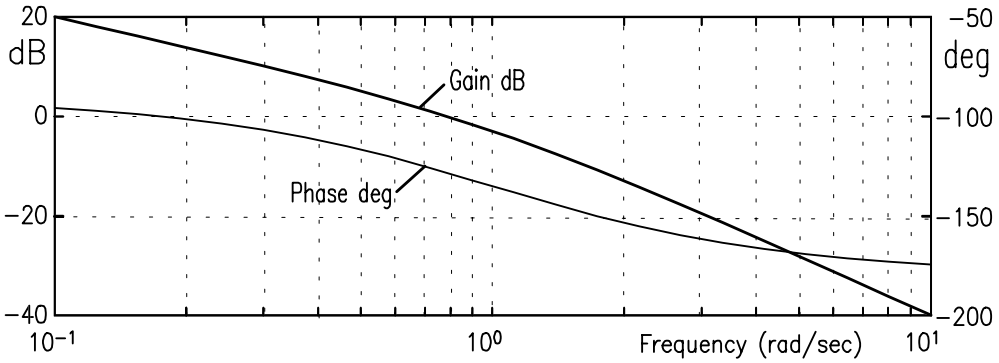


Figure 3-16: Standard frequency characteristic for the DC-motor

3.4. IDENTIFICATION METHODS

3.4.1. Introduction

This section represents a detailed explanation of the "Time Domain Identification" process, shown in the Main Control menu, and derives methods of tuning the parameters K_S and T_S of the model.

3.4.2. Adjustment parameters method

Figure 3-17 shows the basic block diagram of an identification problem. The problem is to tune the parameters of the model in such a way, that the outputs of the model fits the experimental data in sense of an assumed criterion. The basic criterion is the principle of least squares [1]:

$$Q = \int_0^\infty [e_1(t)]^2 dt \text{ , where } e_1(t) = y_1(t) - y_1^m(t)$$

where $y_1(t)$ represents the actual response of the DC motor to a step input, and y_1^m represents the response of the model to a step input.

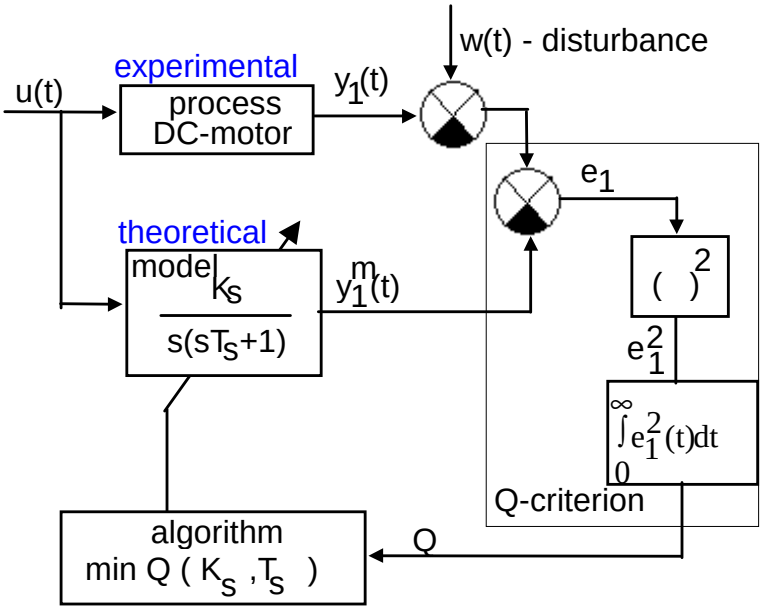


Figure 3-17: Block diagram of parameters adjustment identification method (Q denotes the quadratic integral criterion)

The adjustment algorithm, minimising the value of the criterion Q, operates in the two-dimensional space of the K_S and T_S parameters. In this case, the simple Gauss-Seidel algorithm is useful. Figure 3-18 shows the idea of the Gauss-Seidel algorithm. The parameters of the model are tuned independently, until minimum value of Q is obtained. For the Gauss-Seidel algorithm the step-excitation u(t) = 1(t) is applied.

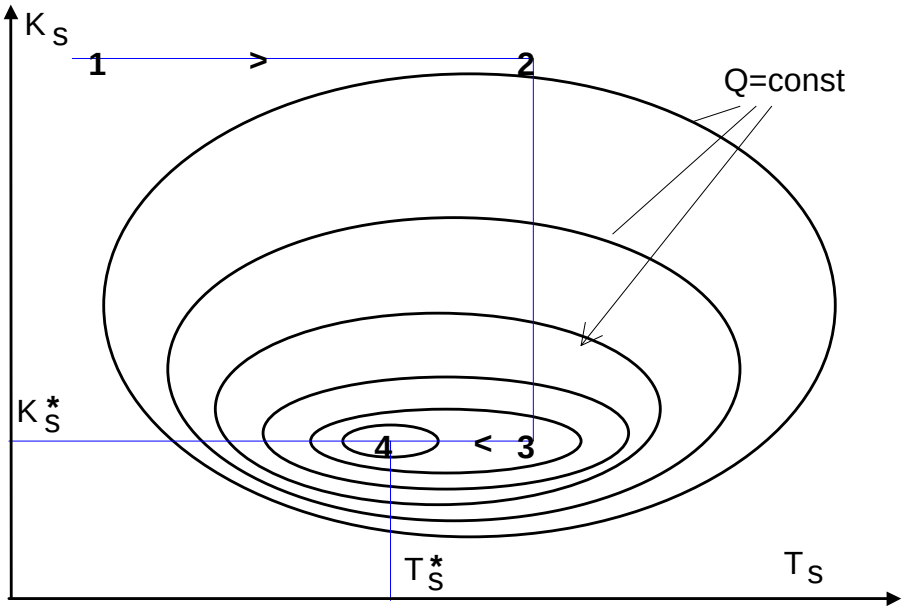


Figure 3-18: Gauss-Seidel procedure for parameter adjustment for the DC-motor model

3.4.3. Analytical solution of the identification problem by the surface method [9]

In the surface method a step input signal is applied: u(t)= 1(t) u_0. A velocity step response is denoted as $y_2(t)$. The problem is to calculate K_S and T_S for the assumed model $G_2(s)$ (Figure 3-19).

$$G_2(s) = \frac{y_2(s)}{y_1(s)} = \frac{K_s}{T_s\, s + 1} \qquad \text{where} \quad T_s = \frac{J}{K_B}$$

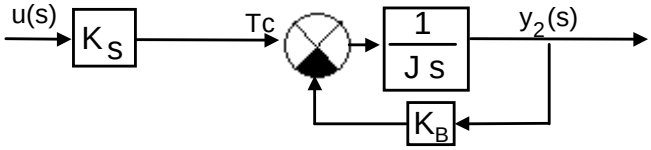


Figure 3-19: Segment of the DC-motor scheme used for the surface method

The required parameters K_S and T_S can be calculated using the following relations:

$$K_S = \lim y_2(t) \; ; \; t \longrightarrow \infty$$

$$T_S = K_1 / K_S$$

where:

$$K_1 = \lim \int_0^t [K_S - y_2(\lambda)] d\lambda \qquad t \longrightarrow \infty$$

The surface method is useful in the presence of disturbances. In this case the integral formula filtrates the measurement noises. Figure 3-20 shows a typical application of this method to a DC-motor identification problem.

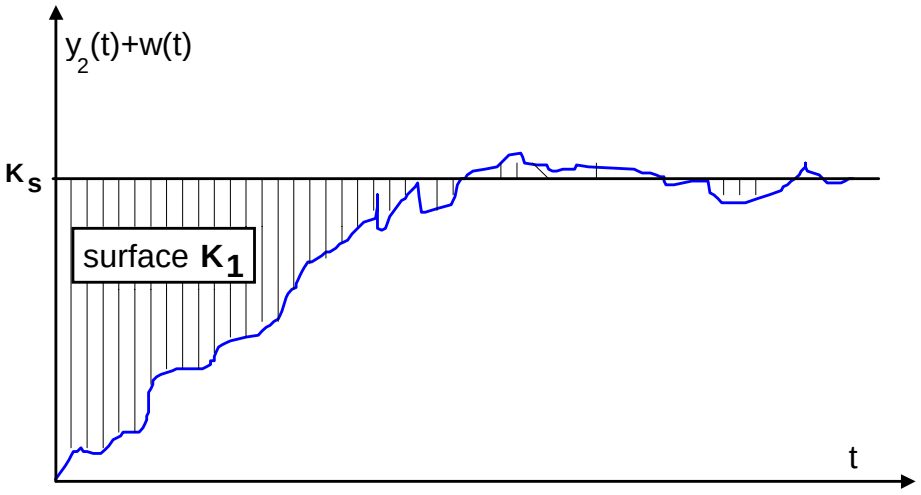


Figure 3-20: Surface method in the presence of disturbances

3.5. PRACTICAL 2.1. Dynamic Properties: Step Response Identification Method

Objectives In this practical, the parameters, K_S, T_S , of a mathematical model of the DC-motor are determined.

Application level: 2 - *Time domain identification* block in the *Main Control Window* .

The hardware and software configuration to realise this practical is given in Figure 3-21.

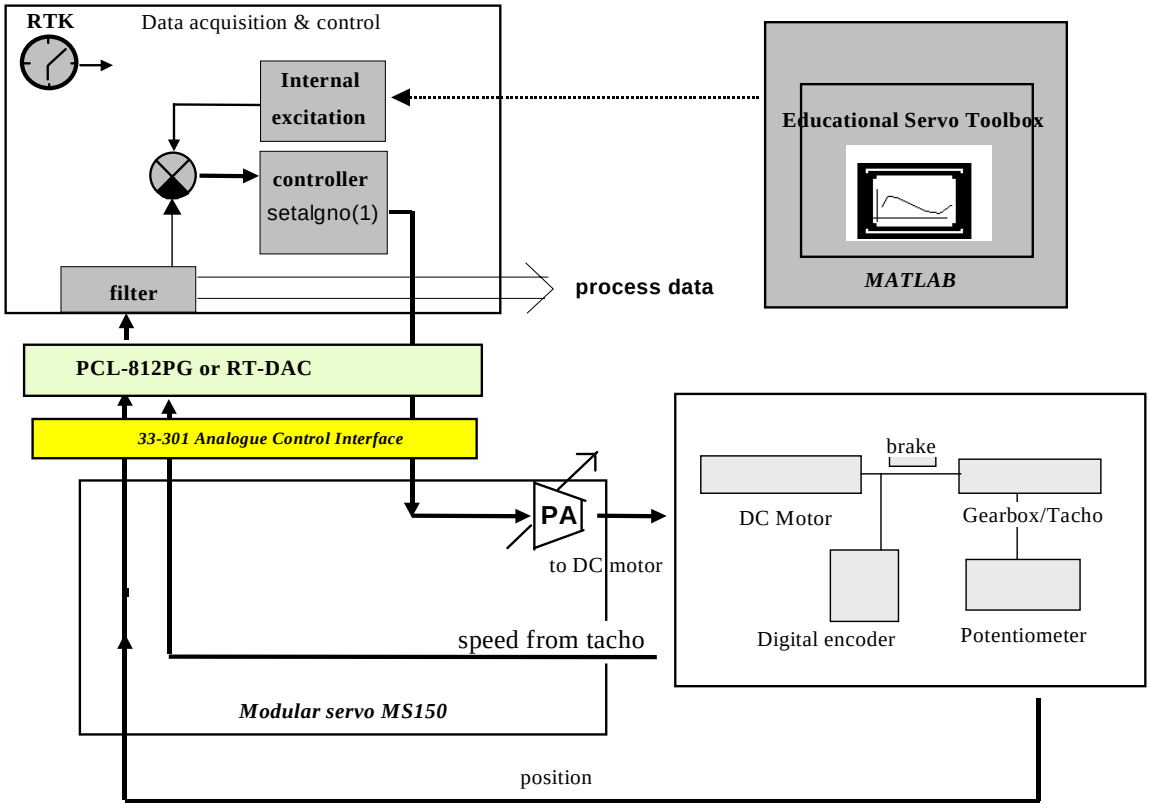


Figure 3-21: Hardware and software configuration for practicals 2.1 and 2.2

To start the Practical 2.1 type ***es*** from MATLAB prompt and *Main Control Window* appears on the screen. Now double click *Time domain identification* button**.** Figure 3-22 shows the opened window.

You can type new value of the step excitation (0 - 1 [normalised]) in the *Control* edit window. After you press *Data acquisition* button the identification experiment is performed five times**.** The average experimental step response appears on the screen.

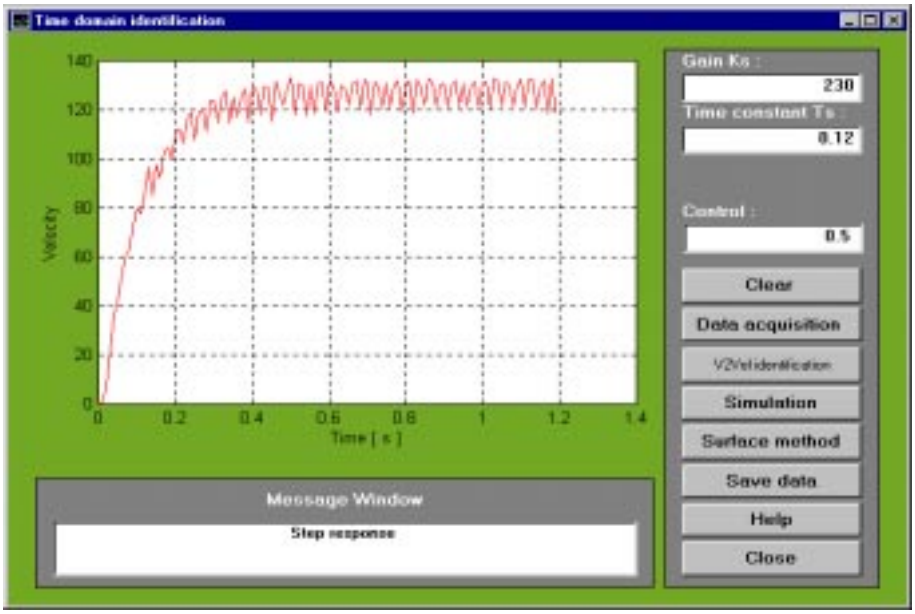


Figure 3-22: *Time domain identification* window

Click *Simulation* button and the step response of the model will be displayed. The model step response is calculated for coefficients which are displayed in the edit windows. On the top of the screen the value of integral square criterion of the model adjustment is displayed.

Now, tune the values of this parameters using the Gauss-Seidel procedure. After a number of steps of the procedure, if you decide that the value of the criterion is relative small click *Surface method* button.

Three curves are displayed on the screen (Figure 3-23):

- average output of the system,
- model output with parameters tuned by a user,
- model output with parameters obtained by the surface method.

In the Message Window coefficients K_s and T_s, calculated using the surface method, are displayed.

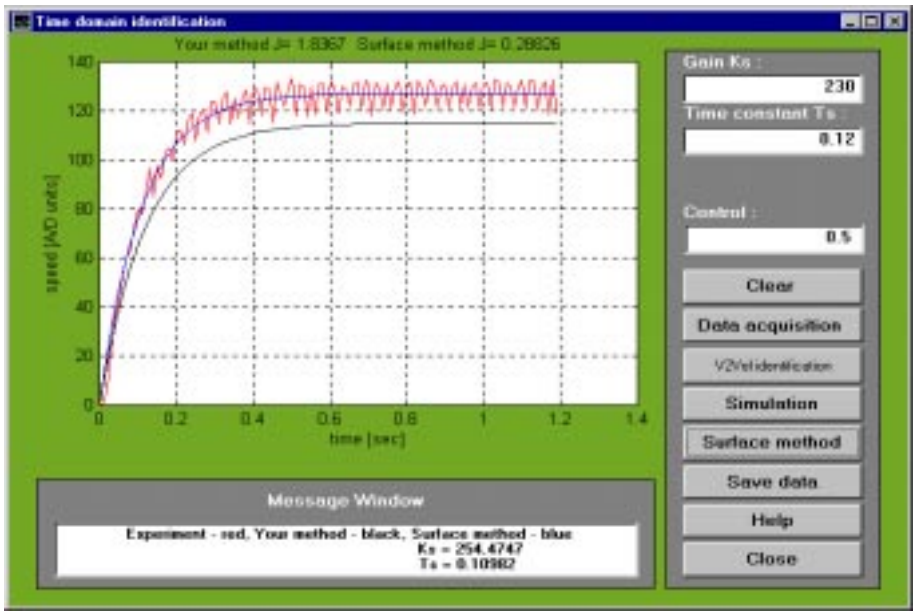


Figure 3-23: Results of time domain identification procedure

If the identification is completed click *V2Vel identification* button to start experiment for *V2Velocity* identification. *V2Velocity* parameter describes dependence of velocity [deg/s] versus voltage signal from the Gearbox/Tacho unit GT150X. Notice, that the *V2Velocity* parameter is necessary to obtain a precise measurements. The default value of this parameter is read from *es_par.ini* file and is equal to 1008. This value depends on the quality of the Reduction Gear Tacho Unit - GT150X.

For a given configuration of your hardware it is enough to perform *V2Velocity* identification only once.

After identification type the obtained value to e*s_par.ini* file. For this purpose open *es_par.ini* file using Windows *Notepad* and type in the appropriate value.

After five passes the plot similar to Figure 3-24 appears on the screen.

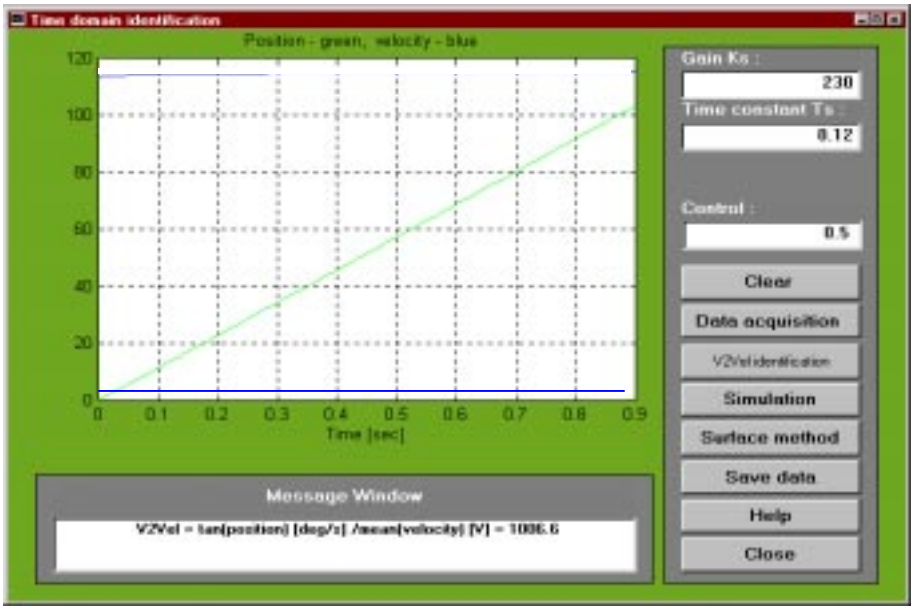


Figure 3-24: Identification of *V2Velocity* parameter

Now click Close button and return to *Main Control Window*.

3.6. PRACTICAL 2.2. Dynamic Properties: Frequency Analysis

Objectives In this practical frequency characteristics of the DC-motor are measured.

Application level: 3, *Frequency characteristics* in the *Main Control Window*.

A hardware and software configuration for this practical is given in Figure 3-21 (see practical 2.1).
To start practical 2.2 double click *Frequency characteristics* button**.** The window in Figure 3-25 opens. Select a number of points of characteristic. Lower bound of the frequency is equal to 0.4 [rad/sec], upper bound is assumed 7.94 [rad/sec]. After you click *Data acquisition* button the identification experiment is started. The sinusoidal input signal to DC-motor, for assumed different frequencies, is used. When data acquisition is completed, experimental characteristics of the DC-motor appear on the screen in blue (Figure 3-25).

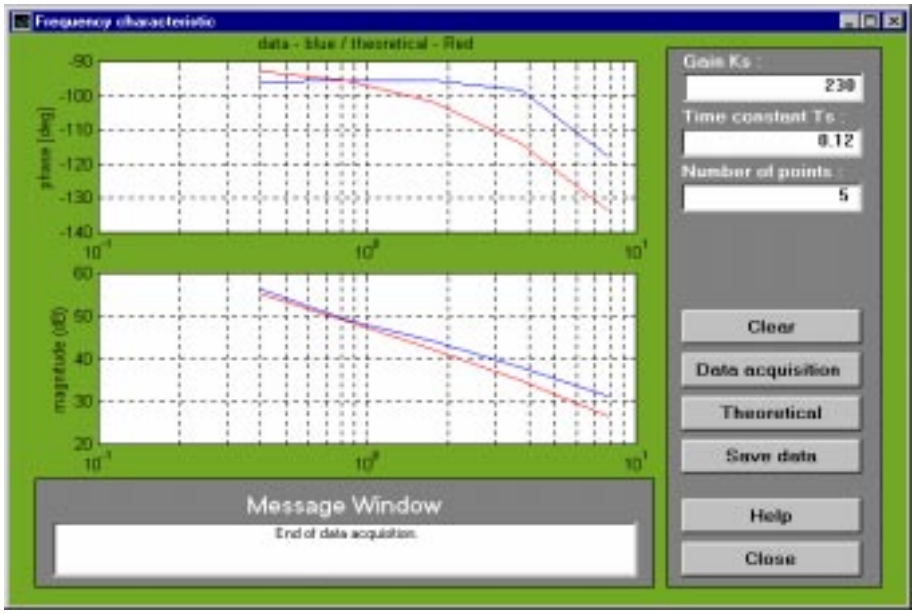


Figure 3-25: *Frequency characteristics* window

Click the *Theoretical* button and the frequency characteristics obtained for the DC-motor model are calculated and displayed in red for comparison on the same plot.

In Practical 2.2 the influence of the magnetic brake position on the dynamic properties of the DC motor may by tested. The position of the magnetic brake changes two parameters of the model: K_S and T_S.

4. ASSIGNMENT 3: Multivariable Control Design

Content: The practicals in Assignment 3 demonstrate how the properties of a closed-loop system are influenced by the design parameters: closed-loop roots and sample period. This assignment introduces two methods of closed-loop systems design. The first is based on a pole placement and is introduced for a continuous system. The second control method named "deadbeat" is used for discrete systems.

4.1. STATE SPACE DESIGN METHOD [1,8]

A closed-loop system with feedback gains from the states is analysed. The approach we wish to apply at this point is a pole placement, that is, having assumed a control law with enough parameters to influence all the closed-loop system roots. There are different ways of achieving this. One of the design methods is described below.

The continuous-time system is represented by the state equation:

$$\frac{dx}{dt} = Ax + Bu$$

$$y = Cx$$

The state controller follows a linear feedback control law in the form: u = -K (y-y_d), where K is a feedback gain matrix.

We request that the roots of the closed system be equal to λ_1, λ_2 (fixed). The design methods then, consist of finding K so that the roots of a closed-loop system are in the desired locations. That means, we assume dynamic properties of the closed system. It can be shown that there exists a linear feedback that gives a closed-loop system with roots specified if and only if the pair (A, B) is controllable. It is clear that the closed-loop system has to be stable and it is a 'sine qua non' condition of the design.

The state matrix of the closed system is:

$$A_c = (A - BKC)$$

For the case of the DC motor the matrix A_C is given as:

$$A_c = \begin{bmatrix} 0 & 1 \\ -\frac{K_s k_1}{T_s} & -\frac{1 + K_s k_2}{T_s} \end{bmatrix}$$

and the characteristic equation is:

$$\lambda^2 + \lambda\frac{1+K_s k_2}{T_s} + \frac{K_s k_1}{T_s} = 0$$

By means of the feedback gains, the roots of the characteristic equation may be changed.

From Vieta's formula we obtain :

$$\lambda_1 \cdot \lambda_2 = \frac{K_s k_1}{T_s} \qquad (\lambda_1 + \lambda_2) = -\frac{1+K_s k_2}{T_s}$$

and we can calculate K = ([k_1 k_2]) from:

$$k_1 = \frac{\lambda_1 \cdot \lambda_2 \cdot T_s}{K_s} \qquad k_2 = -\frac{(\lambda_1 + \lambda_2)T_s + 1}{K_s} \qquad (3.1)$$

This is clear that we can desire any behaviour of the closed-loop system but we have to keep the control between appropriate limits (e.g. -1 ≤ u ≤ +1 [normalised]). When the control variable saturates, it is necessary to make sure that the system still behaves properly.

4.2. DEADBEAT CONTROLLER

It is the control method unique to discrete systems, in which we calculate feedback gains in such a way that roots of the closed system are equal to zero. This control strategy has the property that it drives the states of a closed-loop system from an arbitrary values to zero in at most N steps (dim(A)=N).

In this method the sample time T_0 is the only one design parameter. The magnitude of the control variable *u* can be decreased by increasing the sample time T_0, or vice versa. For a given range of the reference variable step, a suitable sample time can be determined. The base problem in the design is the saturation of the system actuators.

4.2.1. Design method

We can design deadbeat controller using a simulation method. The following steps are necessary in this case:

1. choose sample time T_0,
2. create discrete model,
3. calculate feedback gains,
4. simulate closed-loop discrete system,
5. if the control overruns the maximum allowed limits, increase the sample time T_0 and repeat the steps from 1 to 5.

The system is described by a discrete equation:

$$x[(n+1)T_0]=A_d\, x[n\,T_0] + B_d\, u[n\,T_0]$$
$$y[n\,T_0] = C_d\, x[n\,T_0]$$

The matrices A_d and B_d are calculated from the continuous state-space model using the following relations [1]:

$$A_d = e^{AT_0} \qquad B_d = \left[\int_0^{T_0} e^{At}dt\right]B \qquad C_d = C \quad = I$$

For a discrete model of the DC motor matrices A_d and B_d are in the form:

$$A_d = \begin{bmatrix} 1 & T_s\left(1 - e^{-\frac{T_0}{T_s}}\right) \\ 0 & e^{-\frac{T_0}{T_s}} \end{bmatrix} \qquad B_d = \begin{bmatrix} K_s\left[T_0 - T_s\left(1 - e^{-\frac{T_0}{T_s}}\right)\right] \\ K_s\left(1 - e^{-\frac{T_0}{T_s}}\right) \end{bmatrix}$$

If we assume reachability of the pair (A_d,B_d) and a control law in the form:

$$u[n\,T_0]= -B_d\, K\, (y[n\,T_0]-y_d[n\,T_0]) \quad K=[k_1\ k_2];$$

the closed-loop system is described as:

$$x[(n+1)\,T_0] = (A_d - B_d\, K\, C_d\,)\, x[n\,T_0] + B_d\, K\, y_d[n\,T_0]$$

Figure 4-1 shows the block diagram of the closed-loop system.

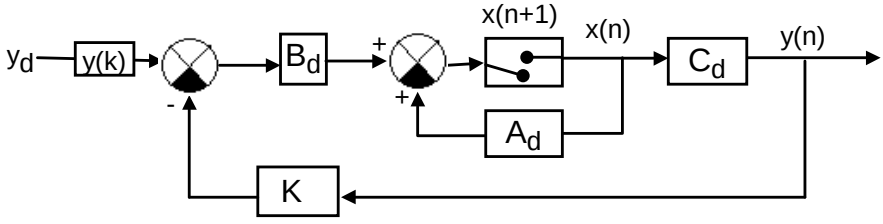


Figure 4-1: Closed-loop system with feedback gain

Feedback gains are calculated from the equation:

$$\text{eig}(A_d - B_d KC_d) = 0$$

and can be displayed after each change of the sample time T_0.

4.3. PRACTICAL 3.1 POLE PLACEMENT BY STATE FEEDBACK

Objectives: In this practical a design of the closed-loop system with a feedback gain is demonstrated and experimental verification with calculated gains is carried out

Application level: 2 *Closed-loop system design and simulation* block in the *Main Control Window*

The hardware and software configuration for this practical is given in Figure 4-2.
In this case setalgno(3) refers to an LQ controller.

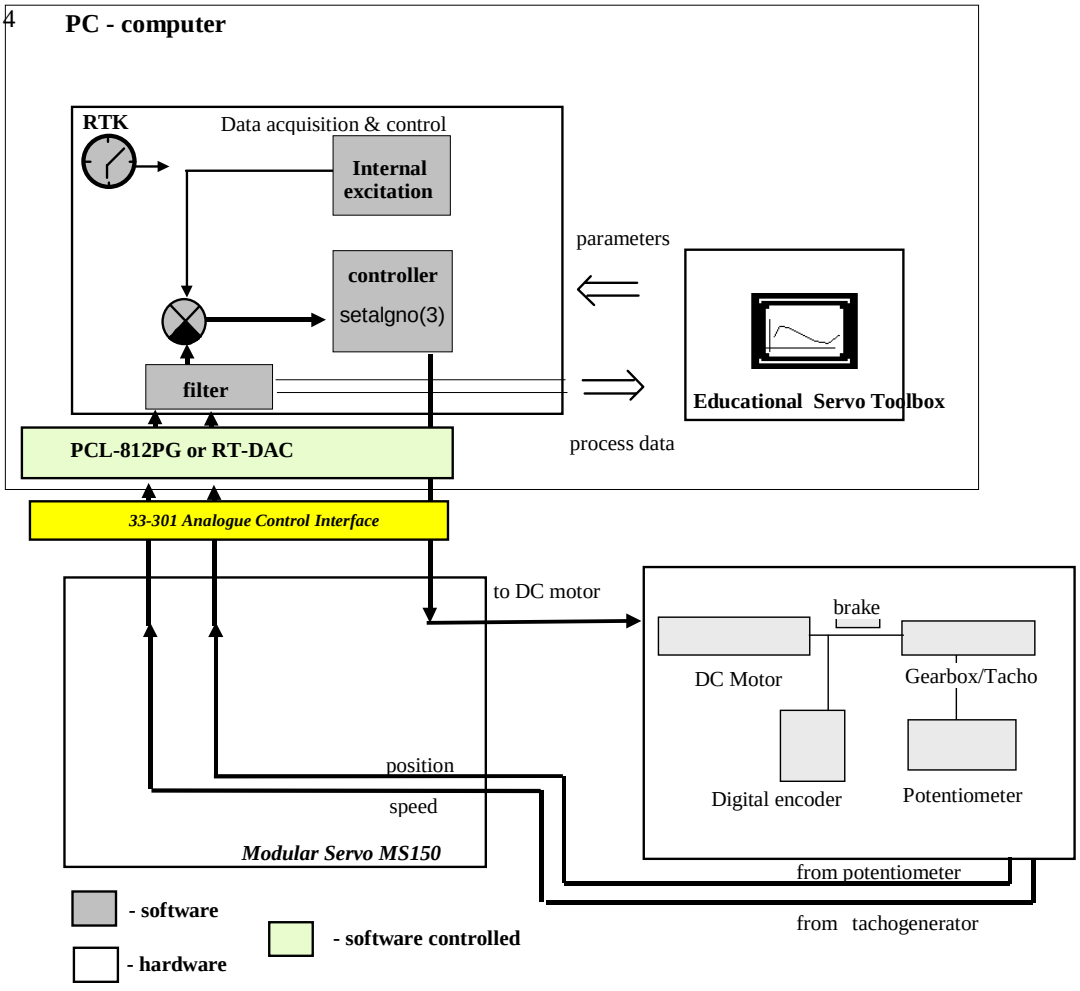


Figure 4-2: Hardware and software configuration for practicals 3.1 and 3.2

The design goal is for a closed-loop system having no oscillations. One possible selection of the roots is:

λ_1= -2 and λ_2= -3

for identified parameters (an example):

Ks=230; Ts=0.12;

from the formula (3.1) we calculate:

k_1 = 0.0031 and k_2 = -0.0017, which are then typed in

Then, we can simulate a closed-loop system with feedback gains k_1 , k_2:

Type ***es*** from the MATLAB prompt and then double click *Closed-loop system design and simulation* button.

Type in the edit windows the system parameters: Ks=230; Ts=0.12, which replace the default values present.

Set reference input equal to 40 (it is the level of the square wave signal), and sample time equal to 0.01.

Next, click *Continuous* button to start simulation. Results are shown in Figure 4-3.

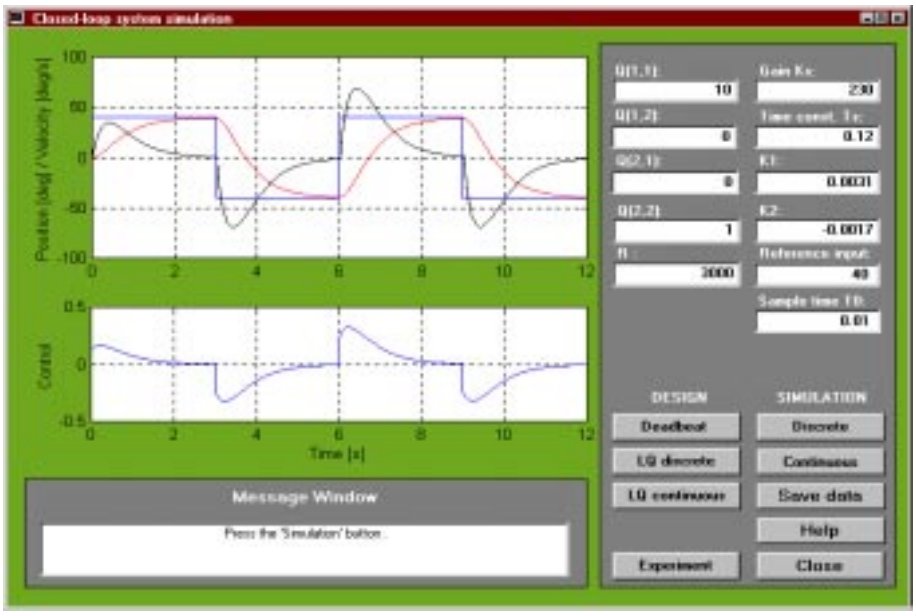


Figure 4-3: Simulation of the closed-loop system , λ_1= -2 and λ_2= -3

Notice, that the response of the system is slow (Figure 4-3). We can change the roots of the closed-loop system to make the system faster.

Assuming λ_1= -4 and λ_2= -5, we obtain: k_1 = 0.0104 and k_2 = 0.00034783.

Repeat the simulation using new values of k_1 and k_2. Example results are shown in Figure 4-4.

Notice that the response of the system is faster.

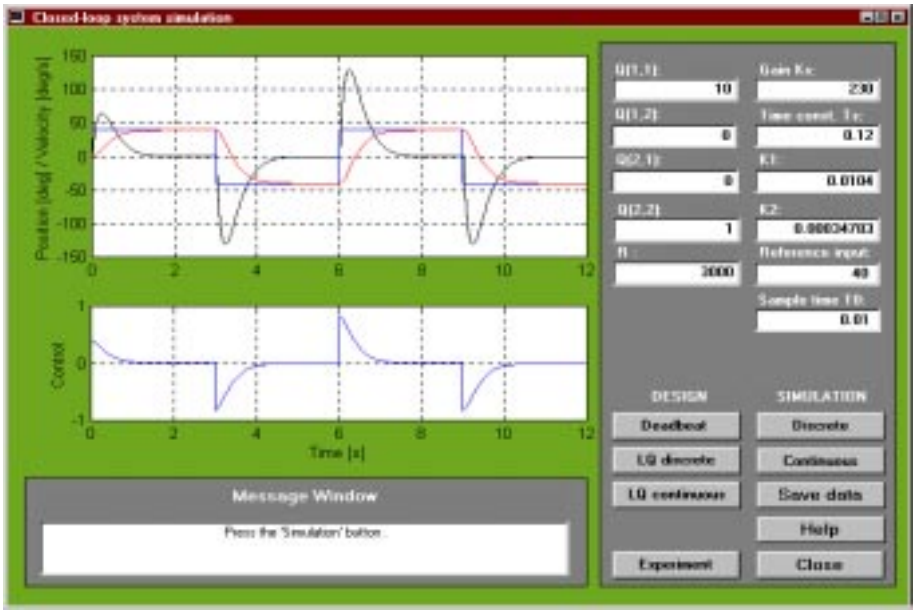


Figure 4-4: Simulation of the closed-loop system , λ_1= -4 and λ_2= -5

Now, the real-time experiment can be started. Click *Experiment* button and window shown in Figure 4-5 opens.

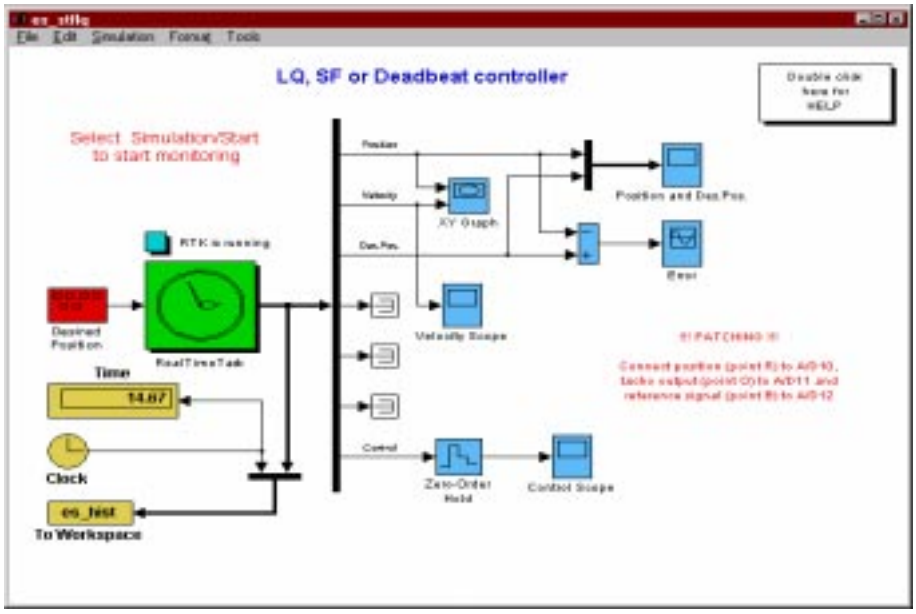


Figure 4-5: *LQ, SF or Deadbeat controller* window

Double click *Real Time Task* block and type the values of k_1 and k_2 parameters. Set sample time to 0.01 and choose Simulink Signal Generator as *Data Source*.

Close the block and then open the *Desired Position Generator* block.

Set *Amplitude* to 40, *Frequency* to 0.2, *Units* to Hertz, and *Wave form* to square.

Close the block and click *Simulation/Start* button.

The results of the real-time experiments are shown in Figure 4-6.

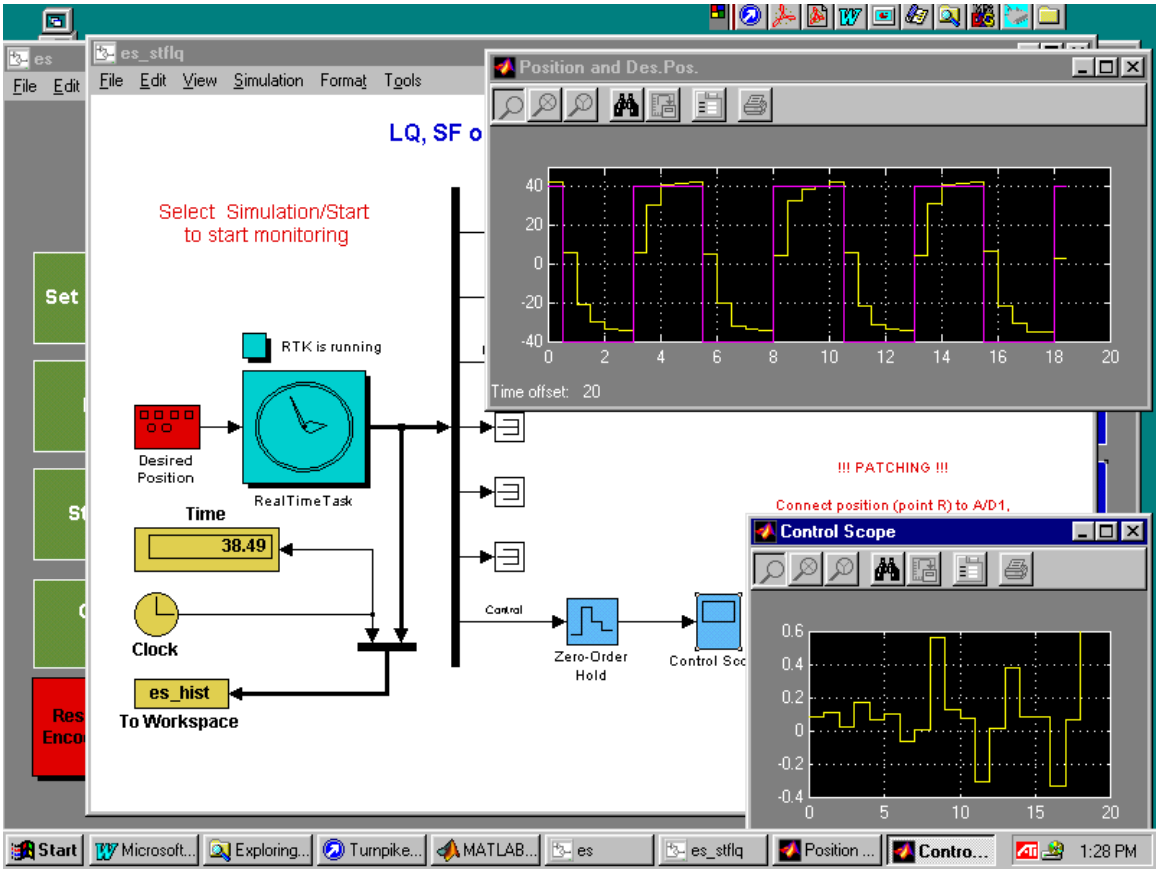


Figure 4-6: Real time experiment for the closed-loop system, λ_1= -4 and λ_2= -5

Compare the result of the real time experiment and the simulation.

4.4. PRACTICAL 3.2 DEADBEAT CONTROL

Objectives: This practical shows how to design the deadbeat controller.

Application level: 2 *Closed-loop system design and simulation* block in the *Main Control Window*.

The hardware and software configuration for this practical is given in Figure 4-2.

Type ***es*** from the MATLAB prompt and then double click *Closed-loop system design and simulation* block. Type in the proper edit windows system parameters: Ks=230; Ts=0.12. Set reference input equal to 40 (it is the level of the square wave signal), and sample time equal to 0.1.

Next click *Deadbeat* button. K1 and K2 edit windows display the calculated values of 'deadbeat' feedback gains. Next click *Discrete* button. Results of the simulation are shown in Figure 4-7.

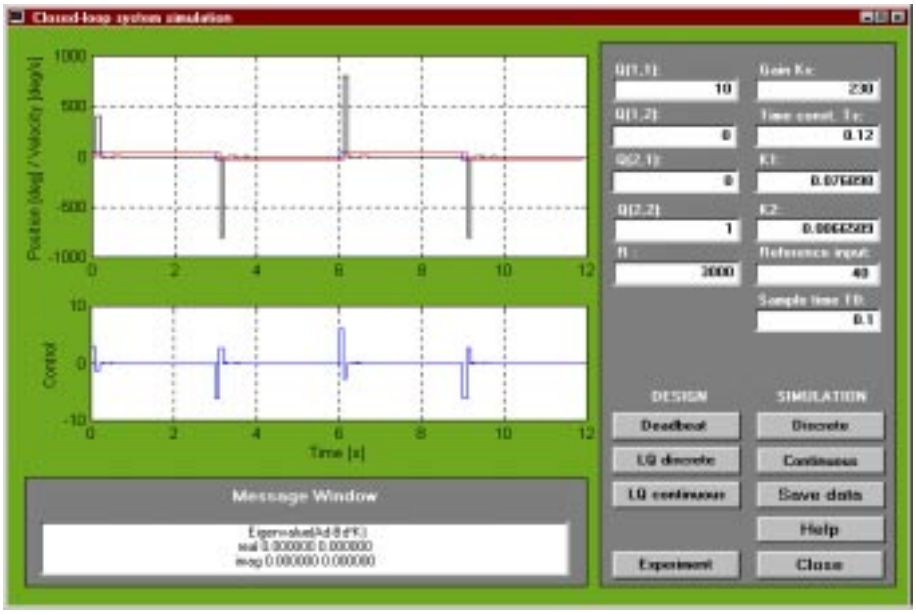


Figure 4-7: Results of the first "deadbeat" simulation experiment (T0 = 0.1)

Notice, that the control value overshoots the upper and lower limits (Normalised to ±1). It means that the controller is not well designed.

Type new value of the sample time : T0 = 0.28 and then click *Deadbeat* and *Discrete* buttons in sequence.

Notice, in this case control values are inside the admissible limits. Feedback gains in this case are equal: $k_1 = 0.013133$, $k_2 = 0.0015617$ for T0=0.35.

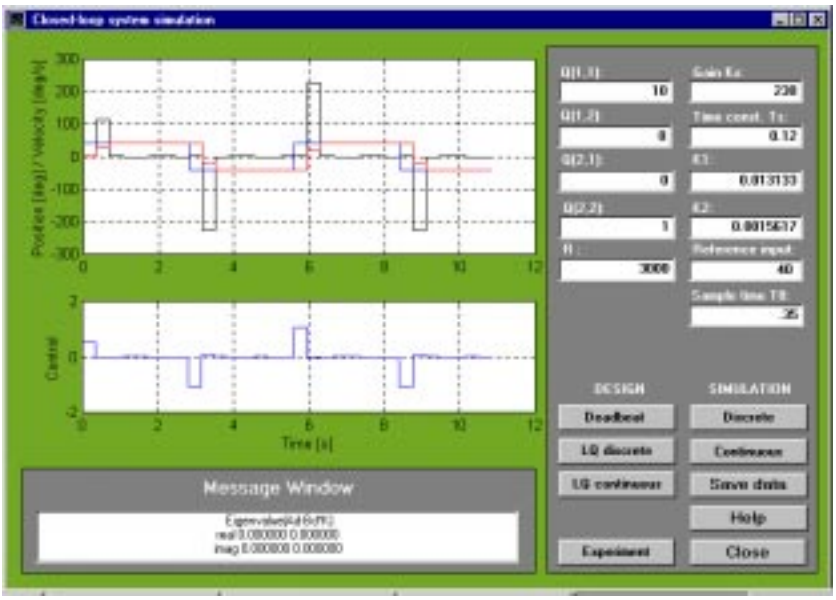


Figure 4-8: Results of the second "deadbeat" simulation experiment (T0 = 0.35)

Now, we can start the the real-time experiment. Double click the *Experiment* button. The window shown in Figure 4-5 opens. Double click *Real Time Task* block and type the last values of k_1 and k_2 parameters. Set sample time to 0.35 and choose Simulink Signal Generator as *Ref. position*. Close the block and then open the *Desired Position Generator* block. Set *Amplitude* to 40, *Frequency* to 0.167, *Units* to Hertz, *Wave form* to square and *Downsampling ratio* to 1. Close the block and click *Simulation/Start* button. The results of the real-time experiments are shown in Figure 4-9.

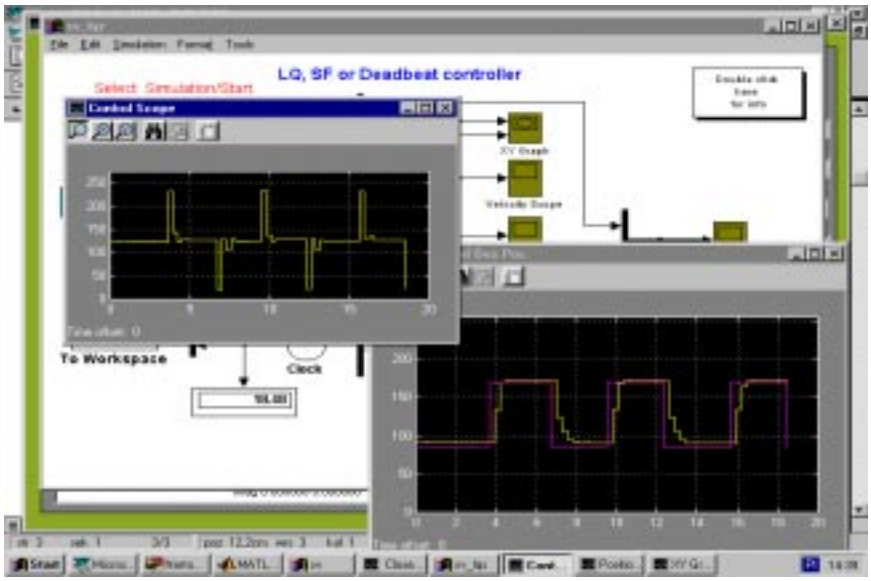


Figure 4-9: Real-time experiment with deadbeat controller

Repeat the same procedure with a sample time T_0 = 0.1 s. Compare the results.

Notes

5. ASSIGNMENT 4: Pid Controller Design

Content: The practicals of the Assignment 4 show how digital PID controllers can be obtained by translating and modifying analogue design.

Note: **(3)** contains details of the derivation of the formulae for PID continuous and discrete parameters

5.1. CONTINUOUS PID CONTROLLER

A block diagram of the continuous PID control system of the servomechanism is shown in Figure 5-1. Note, that only position feedback is applied in this case.

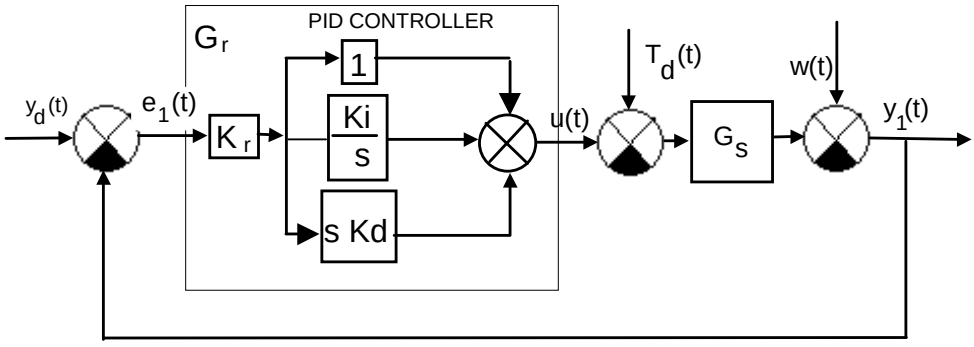


Figure 5-1: Block diagram of the PID control system

Notation used in Figure 5-1	
$y_d(t)$	desired value of position
e(t)	error
y(t)	measured output position
Td(t)	torque external disturbances
w(t)	disturbances measured by sensors

The transfer function of a PID controller is assumed in the form:

$$G_r(s) = \frac{U(s)}{E_1(s)} = K_r\left(1 + \frac{Ki}{s} + s\,K_d\right)$$

The transfer function of the DC-motor is:

$$G_s(s) = \frac{Y_1(s)}{U(s)} = \frac{K_s}{s(s\,T_s + 1)},$$

The transfer function of the servomechanism can be calculated from:

$$G(s) = \frac{Y_1(s)}{Y_d(s)} = \frac{G_r(s)\,G_s(s)}{1 + G_r(s)\,G_s(s)}$$

The error transfer function for the servomechanism is:

$$G_e(s) = \frac{E_1(s)}{Y_d(s)} = \frac{1}{1 + G_s(s)\,G_r(s)}$$

In the formula for the transfer function $G_r(s)$ the following parameters can be tuned by the user:

K_r - gain coefficient,
K_i - integral coefficient,
K_d - derivation coefficient

The PID controller available in the MSW is supplied with a conditional integration option. The integral part of the controller is used only when the error is below some specified limits. See MSW Reference Manual - 33-008-2M5 for details.

5.1.1. Ziegler -Nichols method [3]

Figure 5-2 shows a step response $y_1(t)$ (with disturbances) of the DC- motor.

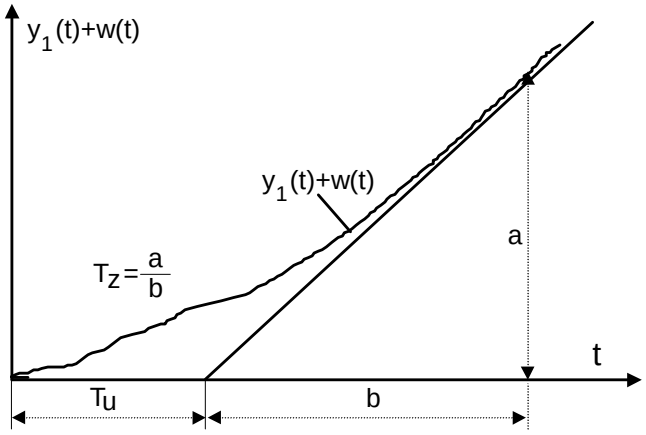


Figure 5-2: Step response of the DC-motor and Ziegler-Nichols model parameters

The Ziegler-Nichols approach is a method for tuning the parameters of a PID controller based on simple process experiments. There are two versions of the method: the ultimate-sensitivity method and transient-response method. The last one is described below. For the use of the Ziegler-Nichols method a simplified model of the DC-motor is introduced:

$$G_z(s) = \frac{T_z\, e^{-sT_u}}{s}$$ where: T_u - time delay, T_z – integration constant (see Figure 5-2)

For this model (astatism with time delay) an application of the Ziegler -Nichols method gives the following parameters:

for assumed overshoot of 2% to 5%:

P controller : $K_r = \frac{0.5}{T_z\ T_u}$

PI controller **:** $K_r = \frac{0.5}{T_z\ T_u}$ $K_i = \frac{1}{5\ T_u}$

PID controller **:** $K_r = \frac{0.65}{T_z\ T_u}$ $K_i = \frac{1}{5\ T_u}$ $K_d = 0.23\ T_u$

for assumed overshoot of 20%:

P controller **:** $K_r = \frac{0.7}{T_z\ T_u}$

PI controller **:** $K_r = \frac{0.7}{T_z\ T_u}$ $K_i = \frac{1}{3\ T_u}$

PID controller **:** $K_r = \frac{1.1}{T_z\ T_u}$ $K_i = \frac{1}{2T_u}$ $K_d = 0.37\ T_u$

T_Z and T_U can be measured from a step response of the DC-motor (Figure 5-2).

5.1.2. Integral square error method

Figure 5-3 shows a general idea of integral square method. The tuning of the parameters is based on the experiment shown in Figure 5-2. The simplified Ziegler-Nichols model of the DC-motor is applied in this case.

For the criterion:

$$J(K_r, K_i, K_d) = \int_0^\infty e_1^2 dt$$

on analytical solution for the optimisation problem:

$$\min \int_0^\infty e_1^2 dt \longrightarrow K_r, K_i, K_d \text{ can be obtained.}$$

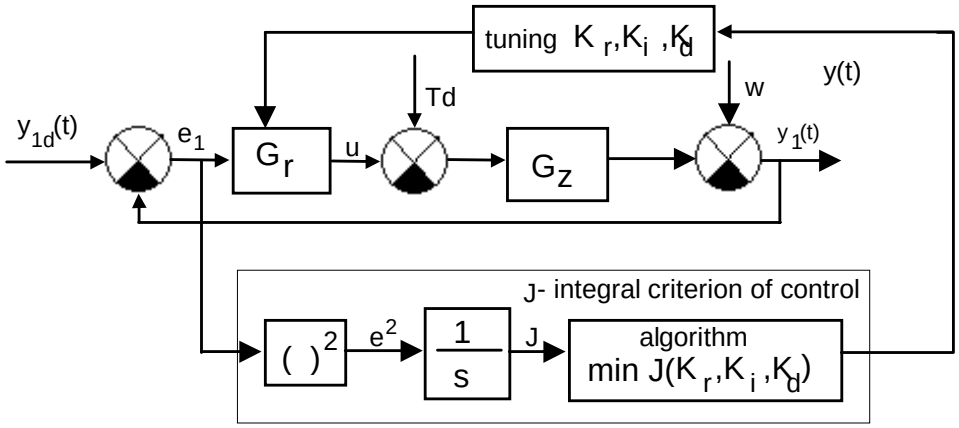


Figure 5-3: Integral square error method for tuning parameters of PID controllers.

In this case, the optimal values of controller parameters are:

For **P** controller: $K_r = \frac{1}{T_z T_u}$

For **PI** controller: $K_r = \frac{1}{T_z T_u}$ $K_i = \frac{1}{4.3\, T_u}$

For **PID** controller: $K_r = 1.3\frac{1}{T_z T_u}$ $K_i = \frac{1}{1.6\, T_u}$ $K_d = 0.5\, T_u$

If a complete model of the of the DC-motor is used: $G_S = \frac{K_S}{s(sT_S + 1)}$, then the optimisation criterion can be calculated as:

$$J(K_r, K_i, K_d) = \int_0^\infty e(t)^2 dt = \frac{(1 - K_s K_r T_s)}{2K_i K_s K_r [T_s + T_i (K_r K_s K_d - 1)]}$$

To obtain the parameters of the controller, the following set of equations must be solved:

$$\frac{\partial J}{\partial K_r} = 0 \ , \quad \frac{\partial J}{\partial K_i} = 0 \ , \quad \frac{\partial J}{\partial K_d} = 0 \ ;$$

Figure 5-4 shows the influence of a proportional (P) controller gain on the output $y_1(t)$. A standard DC-motor transfer function was used for this simulation experiment (T_s=1, K_s=1):

Figure 5-5 shows the error function e(t) for different gains.

Figure 5-6 shows how the integral criterion value is influenced by the controller gain.

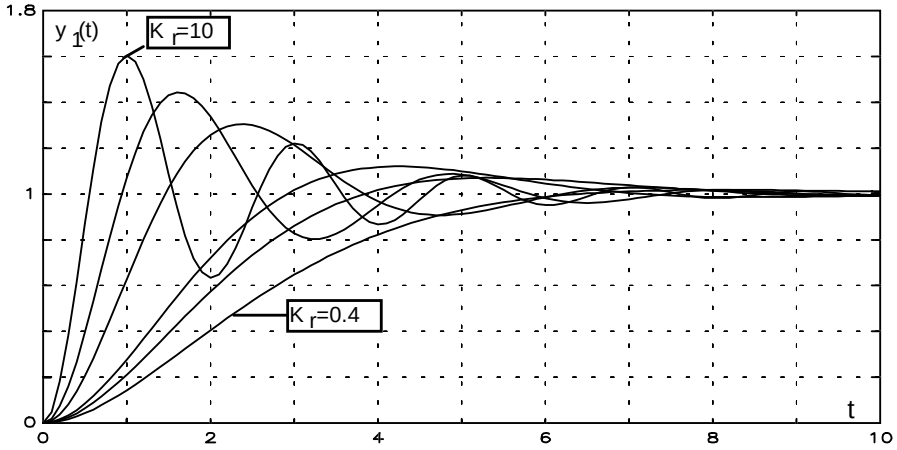


Figure 5-4: Outputs of the servomechanism with proportional (P) controller and standard DC-motor model

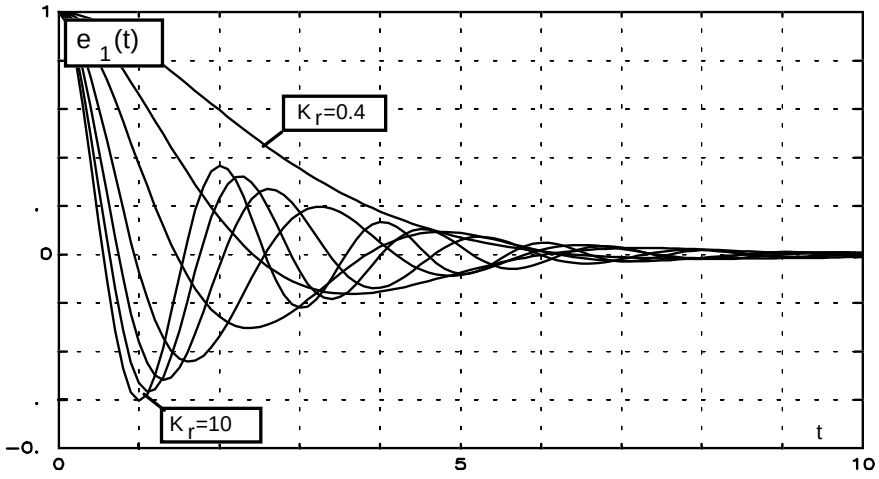


Figure 5-5: Error $e_1(t)$ as a function of gain K_r (proportional (P) control).

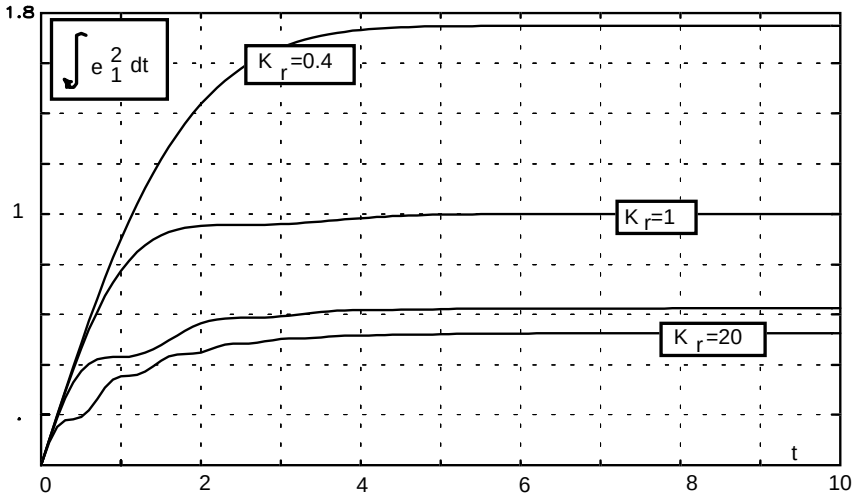


Figure 5-6: Values of the criterion J (proportional (P) control)

5.2. DIGITAL PID CONTROL OF THE SERVOMECHANISM [2][8]

The transfer function of a PID controller represents a differential-integral equation. It is natural to obtain a discrete representation by approximating the derivative with a forward difference and integral with the sum of previous errors. This gives [3]:

$$G_r(z) = K_r\left(1 + K_i \frac{z}{z-1} + K_d \frac{z-1}{z}\right)$$

$G_r(z)$ denotes the transfer function of the digital PID controller.

The model of the digital control closed-loop system is given in Figure 5-7.

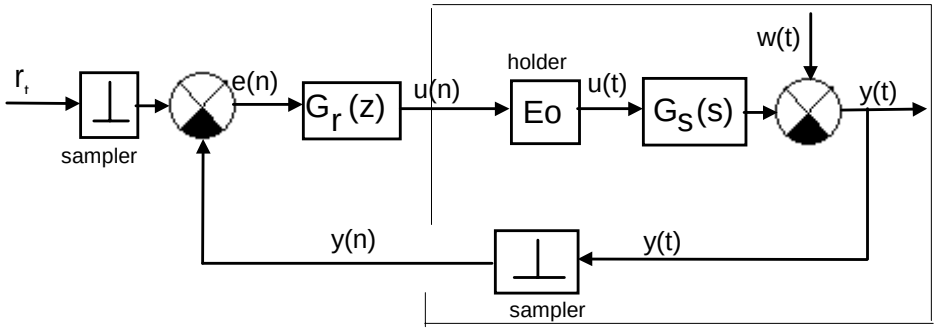


Figure 5-7: DC-motor with digital controller.

The rules of Ziegler-Nichols can be used for tuning of the digital PID controller. The rules are based on the experiments from Figure 5-2 (calculated under the assumption that $T_U \geq 0.5\, T_0$).
For a proportional (**P)** digital controller the Ziegler-Nichols method gives:

$$u_t = K_r \left(r_t - y_t \right)$$

$$K_r = \frac{1}{T_z (T_u + T_o)}$$

For **PI** digital controller:

$$u_t - u_{t-1} = K_r [\left(y_{t-1} - y_t \right) + K_i \left(r_t - y_t \right)]$$

$$K_r = \frac{0.9}{T_z (T_u + 0.5 T_0)} - 0.5\, K_i$$

$$K_i = \frac{0.27\, T_0}{T_z \left(T_u + 0.5\, T_0 \right)^2}$$

For **PID** digital controller:

$$u_t - u_{t-1} = K_r [\left(y_{t-1} - y_t \right) + K_i \left(r_t - y_t \right) + K_d \left(2y_{t-1} - y_t - y_{t-2} \right)]$$

$$K_r = \frac{1.2}{T_z (T_u + T_z)} - 0.5\, K_i$$

$$K_i = \frac{0.6\, T_0}{T_z \left(T_u + 0.5\, T_0 \right)^2}$$

$$K_d = \frac{0.5}{T_z\, T_0} \quad \text{or} \quad K_d = \frac{0.6}{T_z\, T_0} \quad \text{if } (T_u * T_z) \text{ is an integer.}$$

Notice, the parameters of PID digital controller depends on the sampling time T_0.

5.3. PRACTICAL 4.1. ZIEGLER-NICHOLS METHOD (ANALOG AND DIGITAL DESIGN)

Objectives In this practical parameters of PID controller (K_r, K_i and K_d) are determined. An application of Ziegler-Nichols method is demonstrated.

Application level: 2 *PID controller* block in *Main Control Window*.

The hardware and software configuration for this practical is given in Figure 1-1.

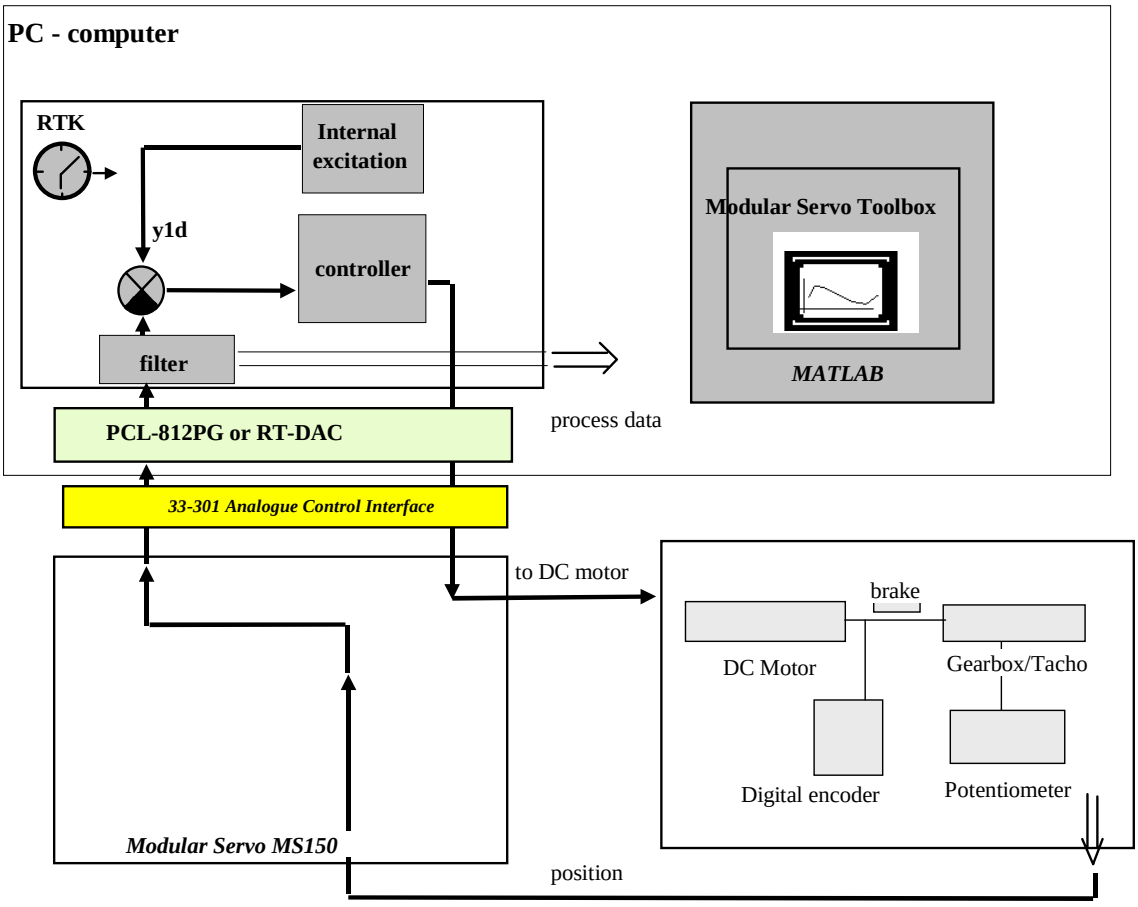


Figure 5-8: Hardware and software configuration for practical 4.1

To start the practical 4.1, type **es** from MATLAB prompt and the *Main Control Window* appears on the screen. Now double click *Ziegler Nichols method* block**.** Figure 5-9 shows the opened window.

Assume, that the model parameters are correctly identified and you want design discrete controller for sampling time T0 = 0.2 s. Click *Data acquisition* button and this starts the experiment. Step input excites the open system and output position is measured. The results are shown in Figure 5-9.

You can see initial part of the step response of the system and straight line which is the Ziegler-Nichols model step response. Two constants are calculated from the plot: T_z - tangent of model response and T_u - delay of the model. According to Ziegler-Nichols coefficients of the continuous and discrete PID controllers are calculated. You can see them in the *Message Window*.

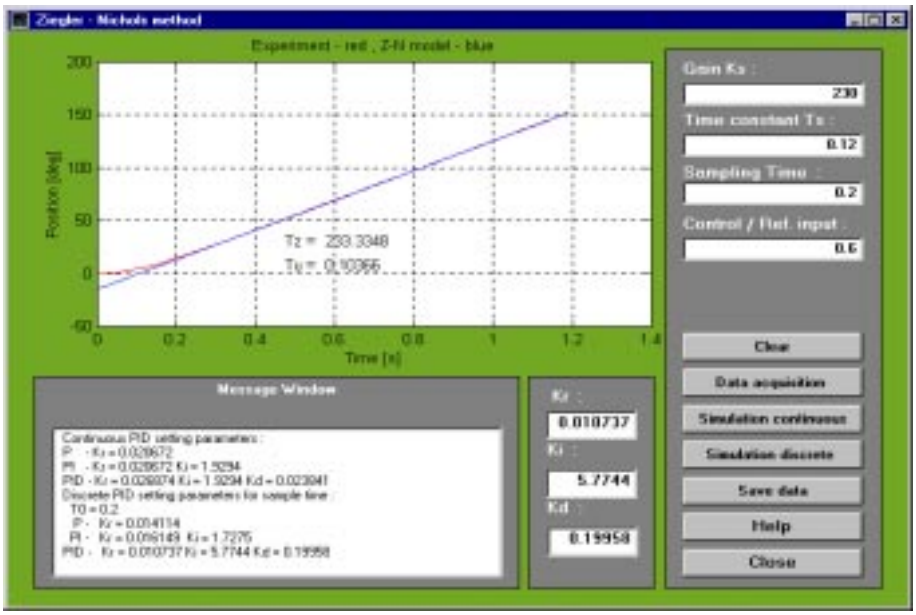


Figure 5-9: *Ziegler-Nichols method* window after the experiment is completed

Now type the reference value equal to 40 in the upper edit window. Then, type the calculated coefficients of the continuous PID controller in the edit windows and click *Simulation continuous* button. Figure 5-10 shows (at the upper window) the response of the closed-loop system for a square wave as reference signal. In the lower window you can see the control which is produced by PID controller. Notice, that the control exceed lower and upper limits (-1 and +1 [normalized]). The damping of the step response is very good and overshoot is less than 30%.

We conclude that this PID controller can properly control the servo system.

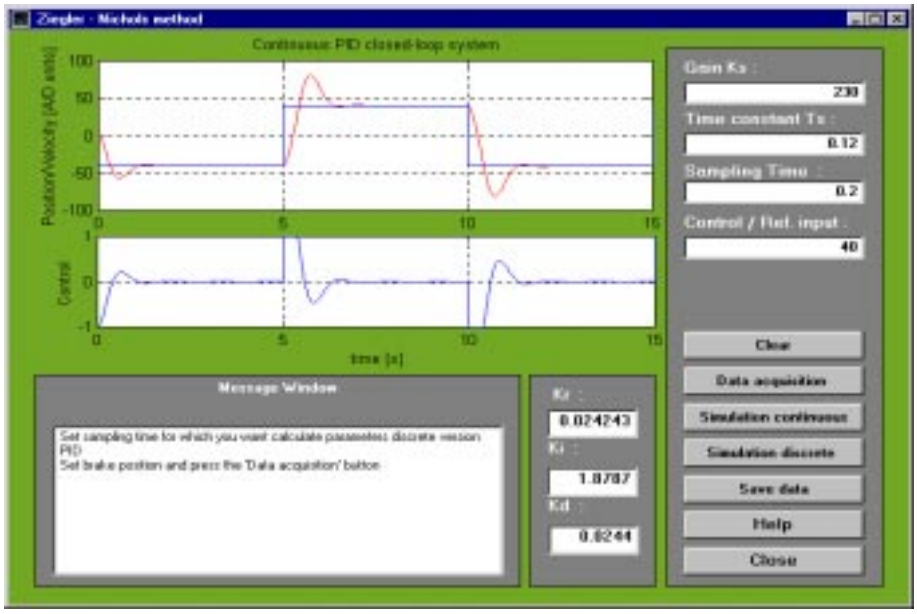


Figure 5-10: Simulation results (continuous version)

Type the calculated coefficients of the discrete PID controller in the proper edit windows and click *Simulation discrete* button. Figure 5-11 shows response of the closed-loop system for square wave as reference signal and the control signal. In this case you can see that closed-loop system works properly as well.

Figure 5-11: Simulation results (discrete version)

In all cases of simulation and practical experiments, arbitrary values of the PID controller settings may be introduced by the user. It is important to remember that PID settings calculated in the continuous case are correct if the sample time T_O is short ($T_O << T_S$ - time constant in model). If the sample time is long relative to T_S, only the PID settings calculated for a discrete case are correct.

A rational choice of the sampling time in a closed-loop control system should be based on an understanding of its influence of the performance of the control system. The selection of sampling time rates can be based on Table 2.1.

After the design the real-time experiment can be started.

Go to the *Main Control Window* and double click *PID controller* button and window shown in Figure 5-12 opens.

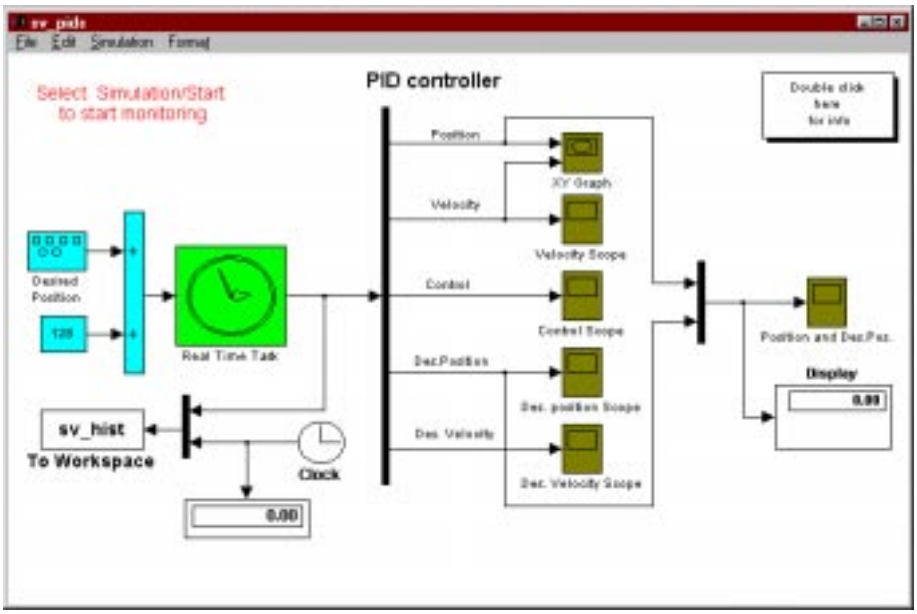


Figure 5-12: *PID controller* window

Double click *Real Time Task* block and type the values of Kr, Ki and Kd parameters calculated in the previous section. Set the sample time to 0.01 and select Simulink Signal Generator as *Ref. source*. Close the block and then open the *Desired Position g*enerator block. Set *Amplitude* to 40, *Frequency* to 0.167, *Units* to Hertz, and *Wave form* to square. Close the block and click *Simulation/Start* button.

Results of the real-time experiments are shown in Figure 5-13.

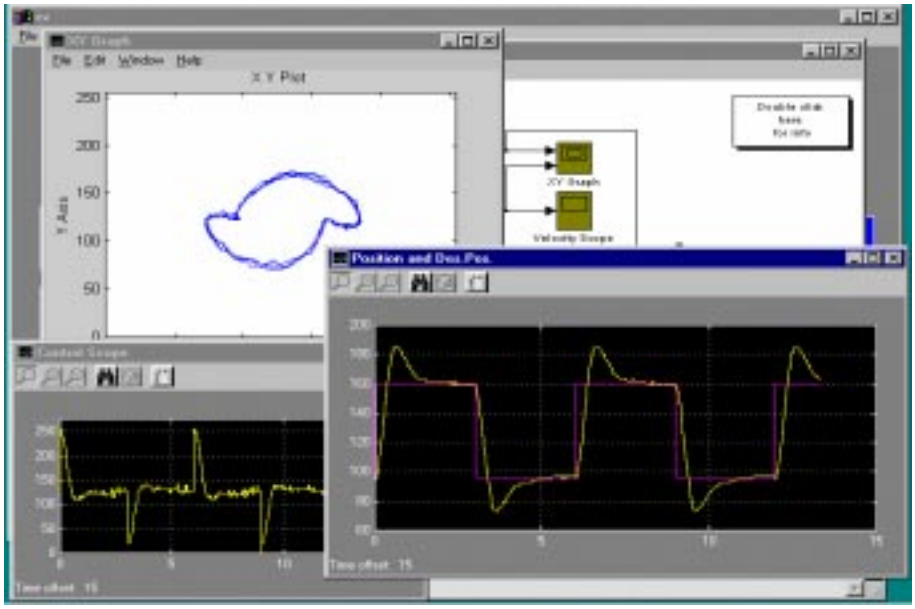


Figure 5-13: Results of PID control experiment (continuous)

To perform the discrete PID control experiment double click *Real Time Task* block and set sample time to 0.2 and type the values of Kr, Ki and Kd calculated for the discrete controller. Select Simulink Signal Generator as *Ref. position*. Close the block and click *Simulation/Start* button.

Results of the real-time experiments are shown in Figure 5-14.

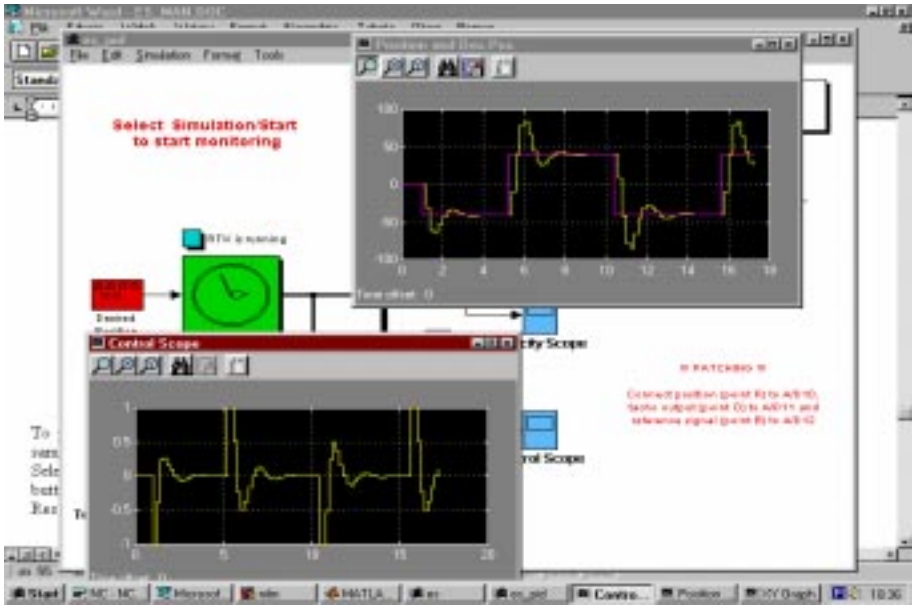


Figure 5-14: Results of PID control experiment (discrete)

6. ASSIGNMENT 5: Optimal Design Method: LQ Controller

Contents: Continuous and discrete linear-quadratic controller design is presented. The difference between continuous and discrete approach to the controller design is discussed. The influence of weights and sampling time is illustrated.

The linear-quadratic problem (LQ problem) is a central one in the theory and applications of optimal control. There are two versions of the LQ problem: the open-loop and the closed-loop optimal control problem. Either the optimal control is given as an explicit function of time for fixed initial conditions, or the optimal controller is synthesised. Only the second case is considered here. The main result of the finite-dimensional linear-quadratic theory is that, under suitable assumptions, the optimal feedback controller is linear with respect to the state, and constant with respect to time.

6.1. LINEAR-QUADRATIC CONTROL PRINCIPLES [2]

The synthesis of the discrete and continuous LQ controller is shown below. For a very small value of the sampling time the response of the discrete system converges to the response of the corresponding continuous system. The most important question for the designer of a control system, as far as LQ regulator problem is concerned, is how to select the weighting factors in the cost function.

Let us examine separately the continuous and discrete LQ problems.

6.1.1. The continuous case

The dynamical model of the DC-motor is described by the linear differential equations:

$$\frac{dx}{dt} = Ax + Bu \qquad (5.1)$$

$$y = Cx,$$

where the matrices A, B, C have the form:

$$A = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{1}{T_s} \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ \frac{K_s}{T_s} \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad (5.2)$$

The desired input time history of state vector is given by yd=[y1d, y2d]. Hence, the error vector **e** is defined:

$$e = yd - y \qquad (5.3)$$

A typical quadratic cost function has the form:

$$J(u) = \frac{1}{2}\int_0^{T_k} [x^T(t)Qx(t) + u^T(t)Ru(t)]dt \qquad (5.4)$$

where:

- Matrix Q≥0, Q is nonnegative definite,
- Matrix R>0,R is positive definite
- The (A,B) pair is controllable.

The weighting matrices Q and R are selected by a designer but they must satisfy the above conditions. It is most easily accomplished by picking the Q's to be diagonal with all diagonal elements positive or zero. Some positive weight (|R|≠0) must be selected for the control, otherwise the solution will include infinite control gains.

The values of elements of Q and R matrices (5.4) are weakly connected to the performance specification. A certain amount of trial and error is required with a simulation program to achieve satisfactory result. A few guidelines can be recommended. For example, all states are to be kept under close regulation and Q is diagonal with entries so selected that a fixed percentage change of each variable makes an equal contribution to the cost. The matrix R is also diagonal.

If the maximum deviations of the servomechanism outputs are: y_{1max}, y_{2max} and the maximum deviation of control is: u_{max} , then the cost is:

$$J = Q(1,1)y_1^2 + Q(2,2)y_2^2 + Ru^2$$

The rule is:

$$Q(1,1)=1/(y_{1max})^2,\ Q(2,2)=1/(y_{2max})^2 \text{ and } R=1/(u_{max})^2 \qquad (5.5)$$

This rule can be modified to satisfy desired root locations and transient response for selected values of weights. One must avoid saturation effects both of outputs and control. Because of the differences in methods of analysis, problem formulation and the form of results, we strongly distinguish the linear-quadratic problem with finite settling time from that with infinite settling time. However in applications we frequently encounter the situation when the termination moment of the control process is so far away that it does not affect the current control actions. The infinite-time optimal control problem is then posed. The cost function (5.4) is replaced by the formula:

$$J(u) = \int_0^{\infty} [x^T(t)Qx(t) + u^T(t)Ru(t)]dt \qquad (5.6)$$

Then the optimal scalar control u^* and the optimal trajectory vector y^* are given by:

$$u^* = - K(yd - y^*) \qquad (5.7)$$

where K is the feedback matrix.

The optimal control problem is now defined as follows: find the gain K such that the feedback law (5.7) minimises the cost function (5.6) subject to the state equation (5.1). The calculation of the control variable which minimises the criterion (5.6) is a dynamic optimisation problem. This problem can be solved by variation calculus applying the maximum principle due to the Bellman optimisation principle. The procedure returns the optimal feedback matrix K**,** the matrix **S**, the unique positive definite solution to the associated matrix Riccati equation:

$$SA + A^TS - SBR^{-1}B^TS + Q = 0 \qquad (5.8)$$

Because of the quadratic appearance of S**,** there is more than one solution, and S must be positive definite to select the correct one. The procedure returns also the matrix E, the closed-loop roots:

$$E = eig(A - B*K*C) \qquad (5.9)$$

The vector K can be calculated by a numeric iterative formula on the basis of the Riccati equation (5.8). The associated closed-loop system:

$$de/dt = (A - B*K*C)e \qquad (5.10)$$

is asymptotically stable.

To solve the LQ controller problem the *lqry* function can be used from the *Control System Toolbox*. The synopsis of *lqry* is: $[K,S,E] = lqry(A_d,B_d,C_d,D_d,Q,R)$.

In this case the matrix Q weights the outputs y instead of the state x. For the servomechanism D_d is the row matrix with two zero elements. The function *lqry* computes the equivalent Q, R and calls *lqr*, the other function from the same toolbox.

The control u is not constrained. Such an assumption cannot be satisfied for a real physical system. One must remember that if the control u is saturated during the practical, then the obtained control is no more of the LQ type.

To return to the LQ problem the amplitude of the u signal should be diminished. In such a case the designer tunes the relative weights between state and control variables in (5.6). Simulation tools are recommended to be applied in this case

6.1.2. The discrete case

If we introduce the sampling period T_0 then the model can be discretized. The discrete model of the DC motor has the form:

$$x[(n+1)T_0] = A_d \cdot x[n\,T_0] + B_d \cdot u[n\,T_0] \qquad (5.11)$$

$$y[n\,T_0] = C_d \cdot x[n\,T_0]$$

where the matrices: A_d, B_d and C_d are in the form:

$$A_d = e^{AT_0} = \begin{bmatrix} 1 & T_S(1-e^{-\frac{T_0}{T_s}}) \\ 0 & e^{-\frac{T_0}{T_s}} \end{bmatrix}, \quad B_d = \int_0^{T_0} e^{A\tau} B d\tau = \begin{bmatrix} K_S(T_0 - T_S(1- e^{-\frac{T_0}{T_s}})) \\ K_S(1-e^{-\frac{T_0}{T_s}}) \end{bmatrix}, \quad C_d = C \qquad (5.12)$$

The matrix A_d is the fundamental solution of the differential equation (5.2) calculated for the sampling period T_0. The explicit values in formula (5.12) of the servomechanism system can be obtained numerically by the use of *c2d* program - conversion from continuous to discrete time -from the *Control System Toolbox.* One must simply type the command:

[Ad,Bd]=c2d(A,B, T_0)

The optimal feedback law:

$$u[n] = - Ke[n] \qquad (5.13)$$

minimises the cost function:

$$J(u) = \sum_{n=0}^{N} [x^T(n)Qx(n) + u^T(n)Ru(n)] \qquad (5.14)$$

subject to the state equation (5.11). The *dlqry* function from the *Control System Toolbox* is used to solve the discrete-time linear-quadratic regulator problem. The synopsis of the *dlqry* and *lqry* programs are identical. The *dlqr* also solves and returns matrix S, the unique positive definite solution to the associated discrete iterative matrix Riccati equation:

$$S_{i+1} = A_D^T S_i A_D - A_D^T S_i B_D (R + B_D^T S_i B_D)^{-1} B_D^T S_i A_D + Q \qquad (5.15)$$

and

$$E = \text{eig}\,(A_d - B_d \cdot K \cdot C_d) \qquad \text{(the closed-loop roots)} \qquad (5.16)$$

The feedback matrix K is derived from S by

$$K = (R + B_D^T S B_D)^{-1} B_D^T S A_D \qquad (5.17)$$

6.2. PRACTICAL 5.1. DISCRETE AND CONTINUOUS LQ CONTROLLERS

Objectives: The practical consists of several examples devoted to LQ controller design.

Application level: 2

The hardware and software configuration for this practical is given in Figure 4-2.

6.2.1. Example 1

Type ***es*** from the MATLAB prompt and then double click *Closed-loop system design and simulation* block. Type in the proper edit windows the system parameters. (K_s=230; T_s=0.12 in this example).

Set reference input equal to 40 (it is the level of the square wave signal), and sample time equal to 0.01. Type in to the edit windows the following assumed weights: Q(1,1)=1, Q(2,2)=1, R=3000. The other parameters have default values.

Next, click *LQ continuous* button. In the K_1 and K_2 edit window will show the calculated values of LQ feedback gains. In our case there are : K_1 = 0.018257 and K_2 = 0.014921. Now, click *Continuous* button. Results of simulation are shown in Figure 6-1.

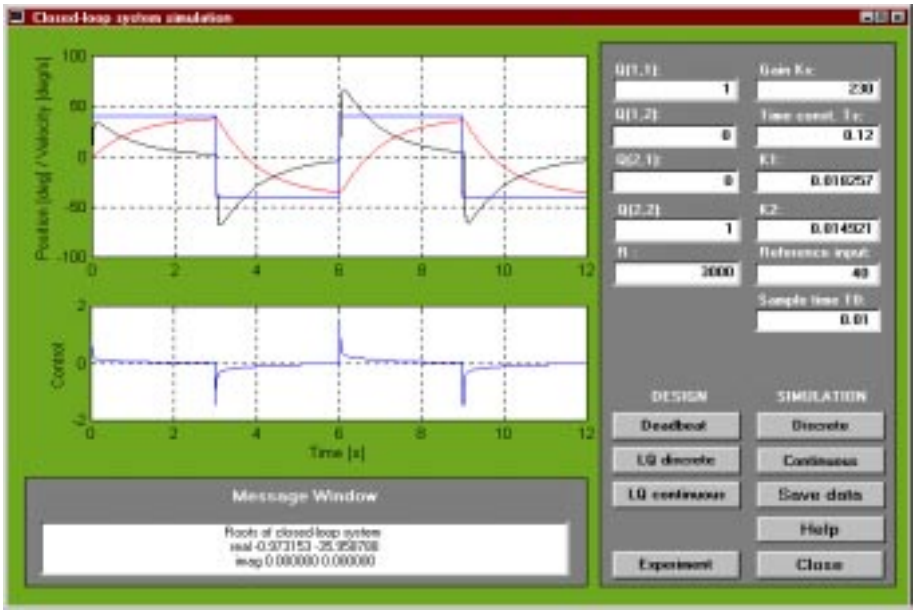


Figure 6-1: Simulation of LQ controller for Q(1,1)=1, Q(2,2)=1 and R=3000

Now we can start the real-time experiment. Double click *Experiment* button. The window shown in Figure 4-5 opens.

Double click *Real Time Task* block and type the values of K_1 and K_2 parameters. Set sample time to 0.01 and as Ref. position choose Simulink Signal Generator. Close the block and then open the *Desired Position Generator* block. Set *Amplitude* to 40, *Frequency* to 0.167, *Units* to Hertz, and *Wave form* to square. Close the block and click *Simulation/Start* button.

Results of the real-time experiments are shown in Figure 6-2.

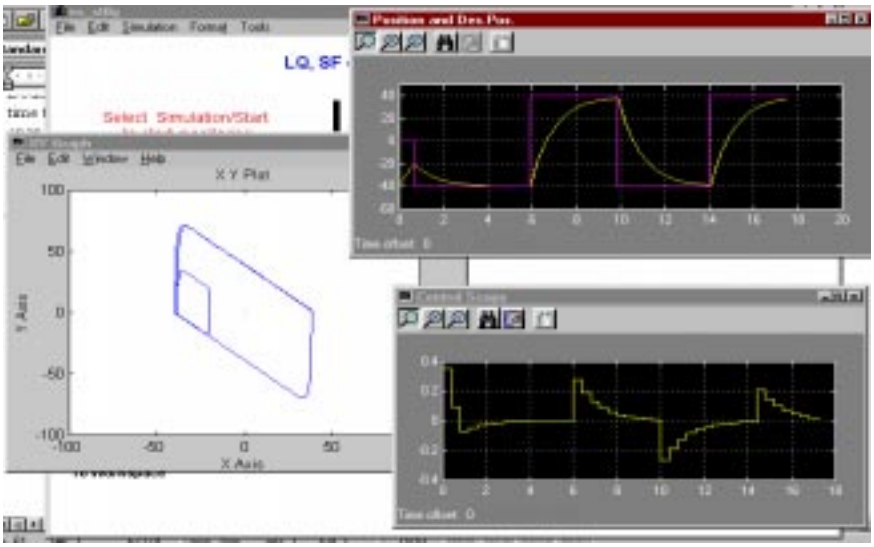


Figure 6-2: Practical of LQ controller for Q(1,1)=1, Q(2,2)=1 and R=3000

6.2.2. Example 2

Invoke the design and simulation for the discrete case. The only new parameter is the sampling period T_0 = 0.1 which type to edit window. Next click *LQ discrete* design button and then *Discrete* simulation button. In this case K_1 = 0.0066784 and K_2 = 0.0031829.

The response of the discrete LQ-controlled mechanism in simulation for the sampling period T_0=0.1 is given in Figure 6-3.

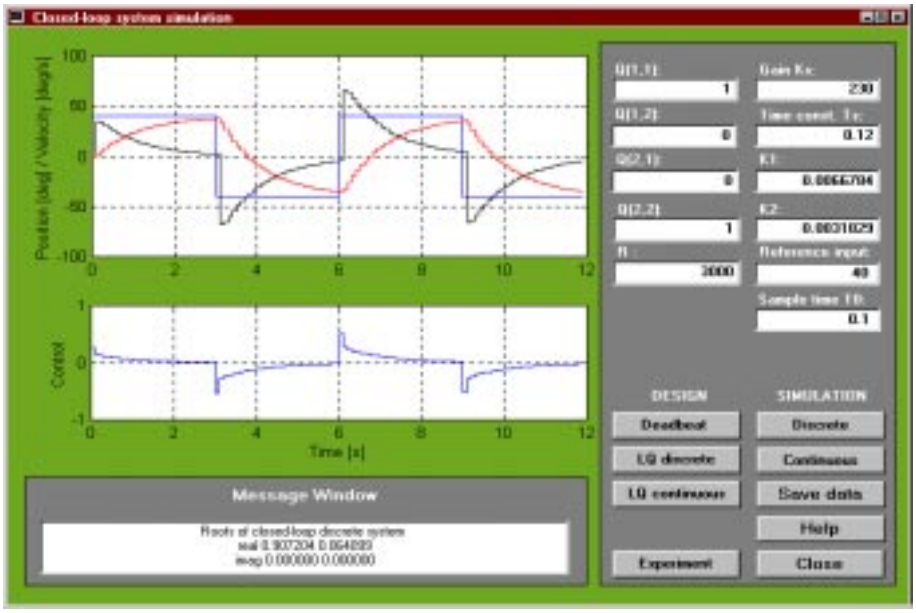


Figure 6-3: Results of LQ control, the discrete case, Q(1,1)=1, Q(2,2)=1, R=3000 and T_0=0.1

6.2.3. Example 3

If the servo positioning task is more important than the speed control task, one must assume a greater weight for Q(1,1). For example: Q(1,1)=10 is ten times greater then Q(2,2)=1. Simulation and experiments give the time responses shown in Figure 6-4 and Figure 6-5.

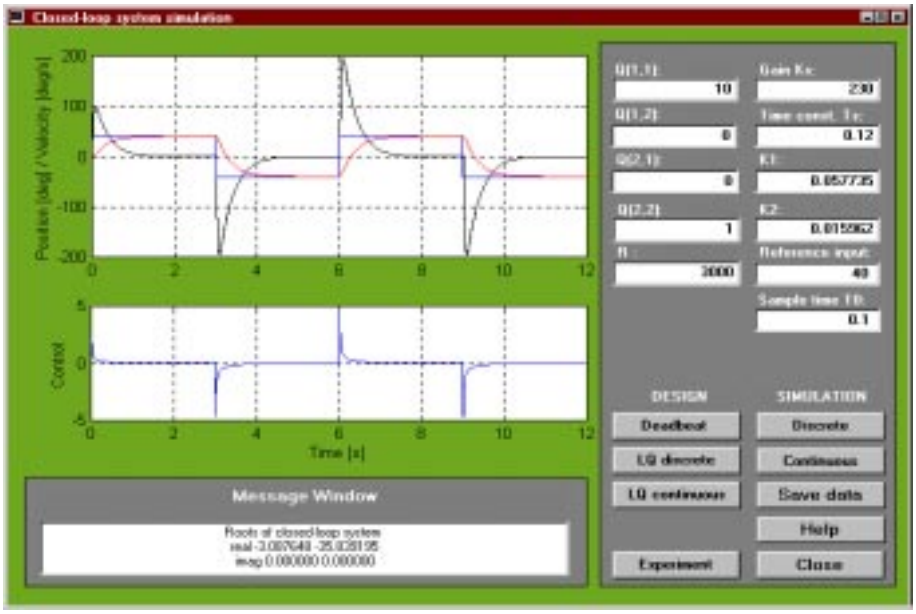


Figure 6-4: Simulation of the LQ controller -the continuous case Q(1,1)=10, Q(2,2) =1, R=3000

In the *Real Time Task* block set sampling time equal to 0.01 (continuous case) and downsampling ratio equal to 5.

Unfortunately, when the experiment is started the control signal is saturated (Figure 6-5).

To obtain a better response of the servomechanism you can proceed in many different ways. For example, increase the control weight R. This parameter is changed in the next example.

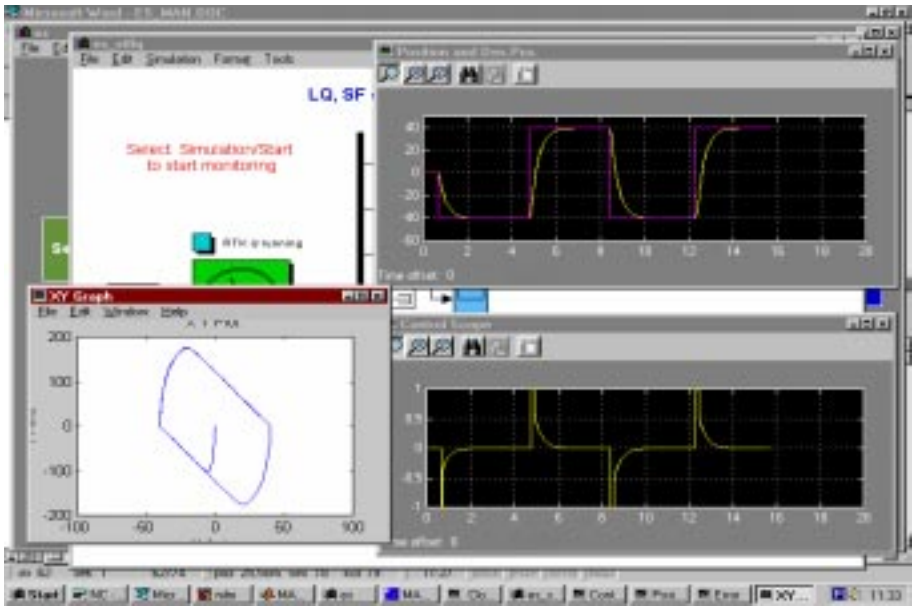


Figure 6-5: Practical of the LQ controller -the continuous case Q(1,1)=10, Q(2,2) = 1,R=3000.

6.2.4. Example 4

Set the new control weight R=6000. The following numerical values for the linear feedback are obtained:

$$K = [K1\ K2] = [0.01\ \ 0.0019242]$$

Verify the simulated response of the servomechanism. The results of the practical for the new parameters of the cost function is given in Figure 6-6. The control saturations have disappeared.

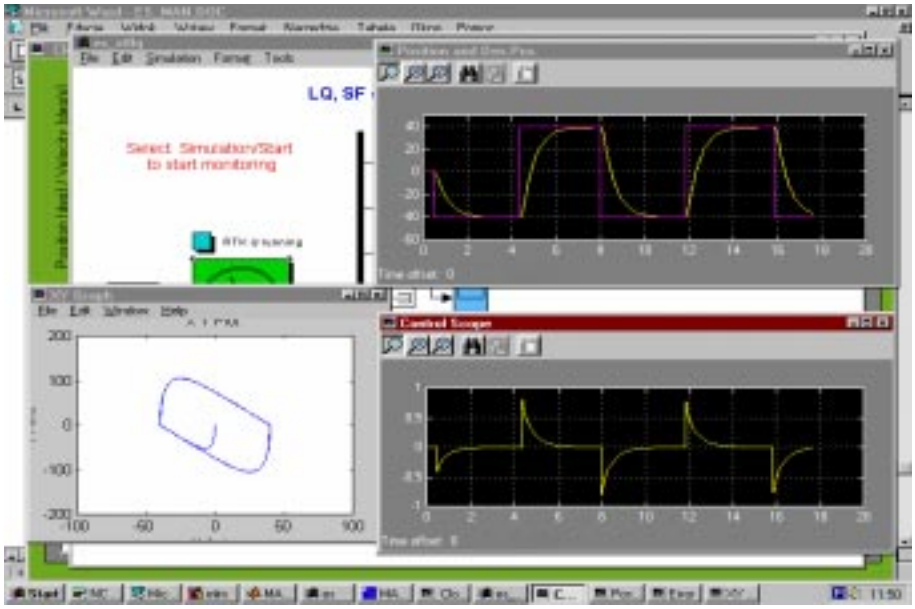


Figure 6-6: Practical of the LQ controller -the continuous case Q(1,1)=10, Q(2,2)=1, R=6000

6.2.5. Example 5

Now, change the sampling time in the *Real Time Task* block to 0.5, downsampling ratio equal to 1 and repeat the experiment. The results are shown in Figure 6-7. It is obvious that the controller does not work properly. It was designed for continuous version. This example shows the significant difference between discrete and continuous approach to the design of controllers.

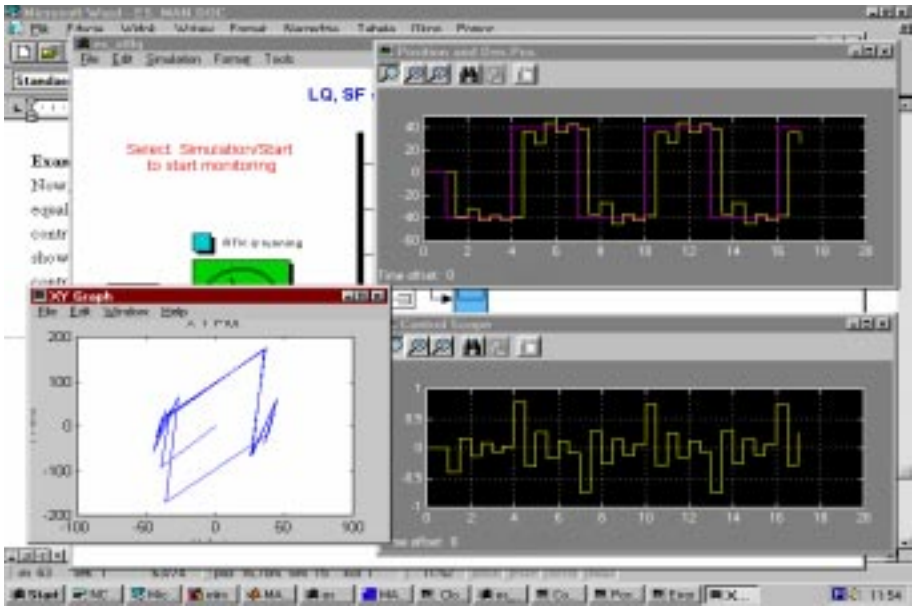


Figure 6-7: Practical of the LQ control - controller calculated for the discrete case Q(1,1)=10, Q(2,2)=1, R=6000, T_0=0.01 and implemented for T_0=0.5

6.2.6. Example 6

To achieve the proper work of the controller the discrete controller will be redesigned for the sampling time T_0 = 0.5. The feedback gains have the following numerical values:

$$K = [0.0054382 \quad 0.00065799]$$

Results of the experiment using the new values of the feedback gains are shown in Figure 6-8.

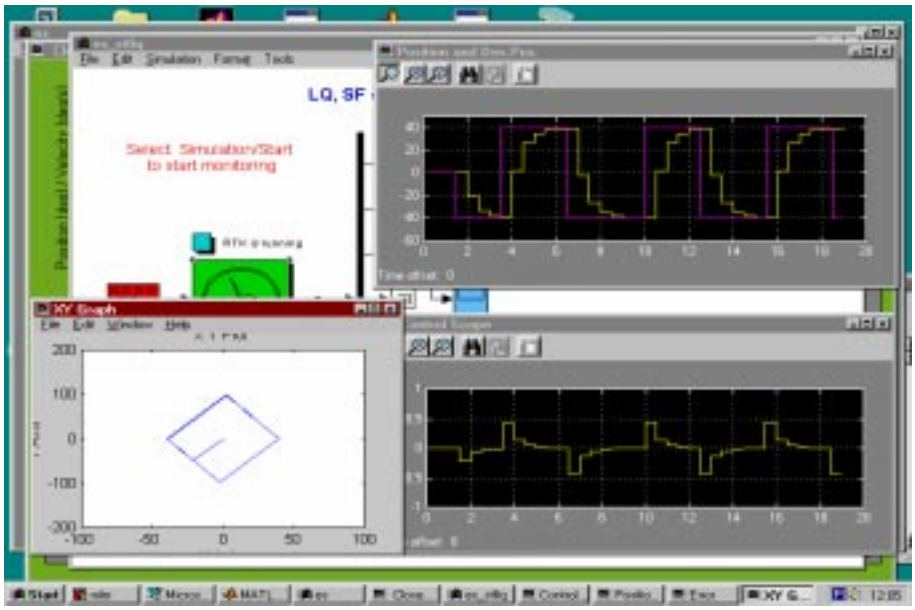


Figure 6-8: Practical of the LQ control - controller calculated for discrete case Q(1,1)=10, Q(2,2) =1, R=6000, T_0=0.5

7. ASSIGNMENT 6: Time-Optimal Control

Contents: Time-optimal controller design is presented. In the practical the time sub-optimal algorithm is implemented to avoid chattering of control.

The important mode of operation for a servo motor is to move quickly from one state to the next. An example of this might be the positioning servo for a robot arm. The synthesis of the time-optimal controller leads to **non-linear** systems. Usually, time-optimal control preserves the "bang-bang" character. Many practices and techniques achieve however near time-optimal solutions. The time-optimal control law may cause difficulties. One can observe that even for laboratory plants, process or measurement noise will introduce the control "chattering" between the maximal and minimal values.

7.1. PRINCIPLES OF DESIGN [2]

Let us review first the time-optimal law and later we will focused on sub-optimal or near time-optimal control strategies.

7.1.1. Transformation to the canonical Jordan form

The state-space model:

$$\frac{dx}{dt} = Ax + Bu \qquad (6.1)$$
$$y = Cx$$

with matrices A, B, C of the servomechanism:

$$A = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{1}{T_s} \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ \frac{K_s}{T_s} \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad (6.2)$$

is transformed to the canonical form:

$$J(A) = P^{-1}AP = \begin{bmatrix} 0 & 0 \\ 0 & -a \end{bmatrix}, \text{ where } a = \frac{1}{T_s} \qquad (6.3)$$

where: P is the non-singular matrix in the form:

$$P = \begin{bmatrix} 1 & 1 \\ 0 & -a \end{bmatrix}; \quad P^{-1} = \begin{bmatrix} 1 & \frac{1}{a} \\ 0 & -\frac{1}{a} \end{bmatrix} \qquad (6.4)$$

The new state variable is defined:

$$z = P^{-1}x \qquad (6.5)$$

The equation (6.1) is transformed then to the canonical Jordan form:

$$dz/dt = J(A)z + P^{-1}Bu \qquad (6.6)$$

where:

$$P^{-1}B = \begin{bmatrix} K_S \\ -K_S \end{bmatrix}$$

For simplicity of the coefficients the following linear transformation can be applied:

$$v = (K_S)^{-1} z \qquad (6.7)$$

Finally, the servomechanism model can be written as:

$$dv/dt = \Lambda v + B_\lambda u \qquad (6.8)$$
$$y = C_\lambda v$$

where:

$$\Lambda = \begin{bmatrix} 0 & 0 \\ 0 & -\frac{1}{T_s} \end{bmatrix}, \qquad B_\lambda = K_s^{-1}P^{-1}B = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \qquad (6.9)$$

$$C_\lambda = CPK_s = \begin{bmatrix} K_s & K_s \\ 0 & -\frac{K_s}{T_s} \end{bmatrix}; \quad C_\lambda^{-1} = \frac{1}{K_s}\begin{bmatrix} 1 & T_s \\ 0 & -T_s \end{bmatrix};$$

We obtain the system of two independent variables: v_1 and v_2. It means that the differential equations (6.8) are de-coupled and the derivatives $\frac{dv_1}{dt}$ and $\frac{dv_2}{dt}$ are linear functions of: v_1, u and v_2, u respectively.

7.1.2. Solution of the time-optimal problem

The equation (6.8) for:

$$u = \pm u_{max} \tag{6.10}$$

can be solved explicitly. The solution is:

$$v_1(t) = v_1(0) + t \cdot u \tag{6.11}$$

$$v_2(t) = \exp(-t / T_S)\, v_2(0) - T_S\, u[1 - \exp(-t / T_S)] \tag{6.12}$$

From the above equations eliminating t we get the formula for the system trajectories. The trajectories start from an arbitrary initial point $v_1(0)$ and $v_2(0)$ under the excitation u :

$$v_2(t) = \exp\{[v_1(0)-v_1(t)] / (uT_S)\}\, [v_2(0) + T_S\, u] - T_S\, u \tag{6.13}$$

From the above formula we find the two trajectories (for $u = \pm u_{max}$) which are crossing the origin of the system i.e. the point (0, 0). Simply we put $v_1(t)=v_2(t)=0$. It results in:

$$v_2(0) = T_S\, u\, \{\exp[-v_1(0) / (uT_S)] - 1\} \tag{6.14}$$

This equation represents two curves. We are interested only in this part of each trajectory which leads to the origin. The "switching curve" consists of two half-curves. We can write it down in the compact form:

$$0 = v_2(0) + T_S\, u\, \{\exp[v_1(0) / (uT_S)] - 1\} \tag{6.15}$$

where: $u = u_{max} \cdot \text{sign}[v_1(0)]$.

The "switching curve" (6.15) and a number of the system trajectories (6.13) starting under the control $u = u_{max}$ (6.10) from an arbitrary initial conditions: $v_1(0)$, $v_2(0)$ are given in Figure 7-1.

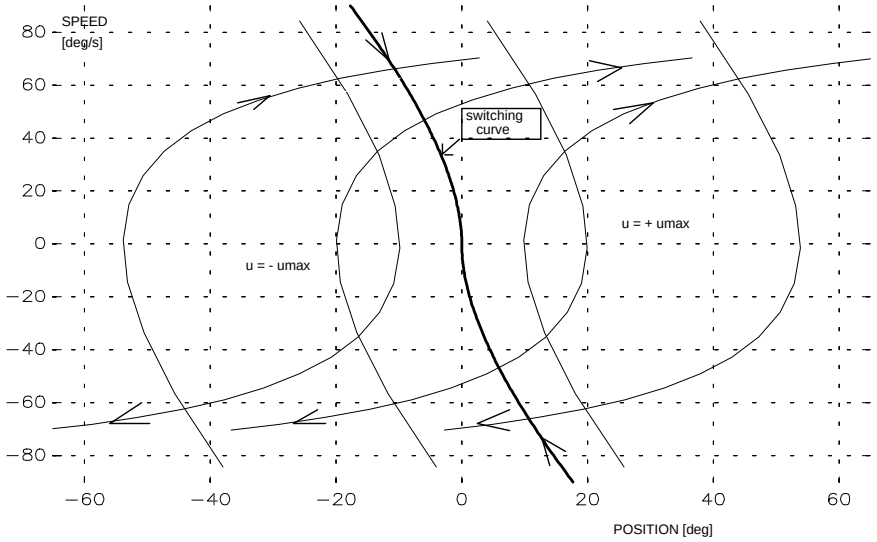


Figure 7-1: The "switching curve" and system trajectories for $u = \pm u_{max}$

It is well known from the maximum principle, that for a second order linear dynamic system described by two ordinary differential equations, under the assumption on the control u:

$$|u| \leq u_{max}, \qquad (6.16)$$

the control u belongs to the boundary i.e. $u = \pm u_{max}$ and its value is switched when the system trajectory is crossing the switching curve. The appropriate sign of the control must be applied at the beginning of the process and must be switched to the opposite value when the switching curve is achieved. The controller is testing the sign of the expression (6.16) in every step of system evolution. The value of this expression differs from zero if the point lies outside the switching curve.

The basic control rule is:

if : the right side of the equation (6.15) > 0 then $u = -u_{max}$ else $u = u_{max}$ (6.17)

The schematic diagram of the time-optimal controller is shown in Figure 7-2. If the simulation of the system is concerned the DC-motor and the non-linear controller are modelled. On the contrary to the simulation, the non-linear controller is the only part which is modelled for the practical. In this case the DC-motor model is replaced by the real DC-motor.

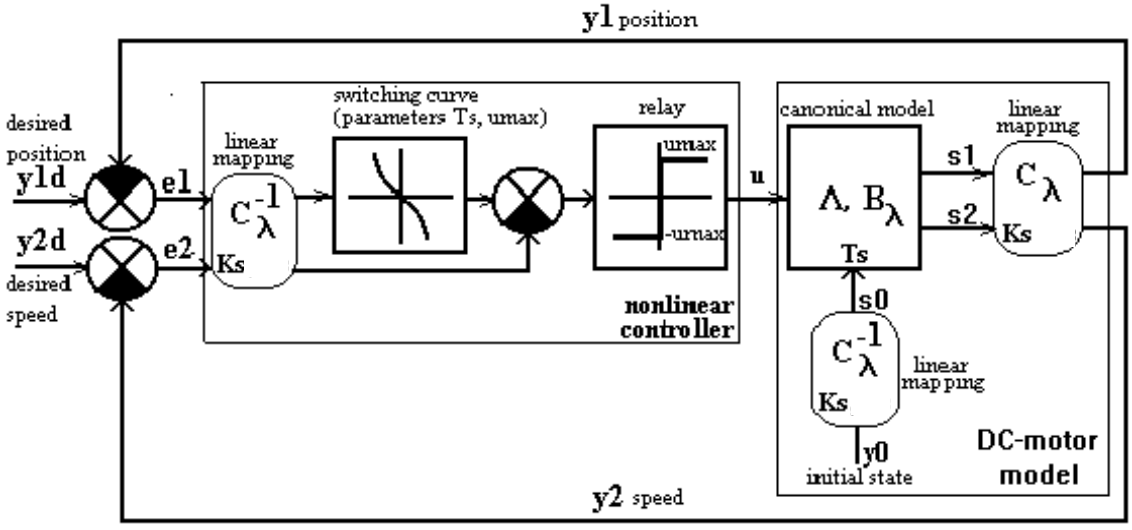


Figure 7-2: Time optimal control of the DC-motor model.

The time-optimal servomechanism is very sensitive to parameter variations. Therefore the model coefficients: T_S, K_S should be identified properly. It is a well known effect that the time-optimal rule (6.17) results in chattering of the servo. Near the origin the motor is excited by the control sequence: $+u_{max}$, $-u_{max}$, $+u_{max}$ ... switched with a high frequency. The sliding mode due to measuring noise is observed.

To avoid these effects one can change the time-optimal algorithm in the neighbourhood of the origin. For example: a decaying factor can reduce the amplitude of the control proportionally.
Now the non-linear controller from Figure 7-2. is equipped with the origin zone "sensor". This is illustrated in the Figure 7-3. The origin zone is only detected for position error. It is sufficient to achieve the goal of control. Notice, that a great level of noise is added to the measurements of speed.

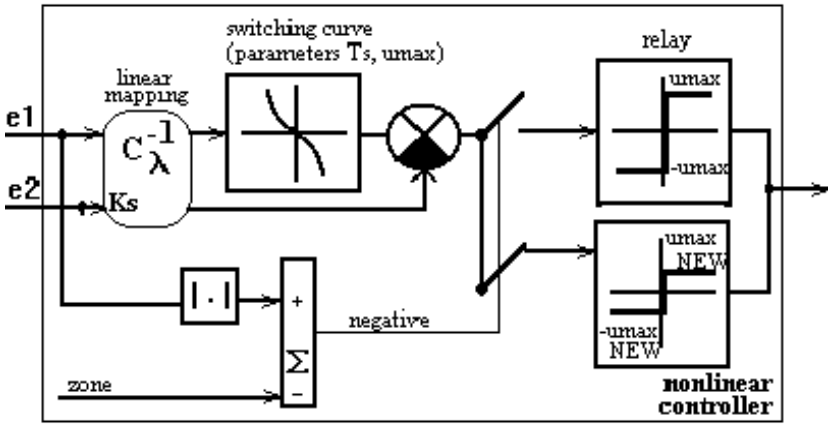


Figure 7-3: The origin zone "sensor".

The new time-suboptimal control algorithm is :

$$u_{maxOLD} = u_{max}$$

If $|e_1| < zone$, then $u_{max\ NEW} = u_{max\ OLD}\ |e1| / zone$,

else: apply rule (6.17) (6.18)

However, in practice switching moments are delayed. The servomechanism control is not switched when the trajectory achieves the switching curve but one step later. According to this observation one can modify the time-optimal rule. The upper ($v_2(t)>0$, $v_1(t)<0$) and lower ($v_2(t)<0$, $v_1(t)>0$) branches of the switching curve are moved respectively to the left and right from the origin by the interval tune. The new switching curve has the form:

$$0 = v_2(0) + T_s\, u\, \{\exp[v_1(0)+tune\cdot sign[v_1(0)] / (uT_s)] - 1\} \quad (6.19)$$

where: $u = u_{max} \cdot sign[v_1(0)]$

It results in an earlier - compared to the original switching curve - switching moment. The new slightly modified switching curve is obtained. This curve drawn by the bold line is shown in Figure 7-4. As far as the servo-motor time-optimal control is concerned, only the practical allows to select the best values of the parameters: tune and zone.

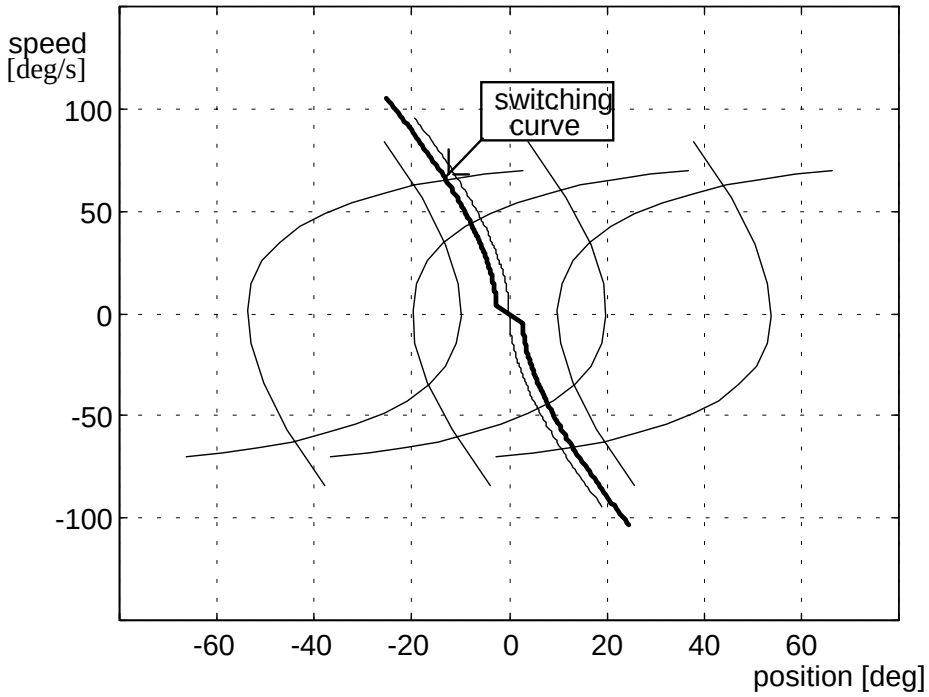


Figure 7-4: The "switching curve" modified by the parameter tune

7.2. PRACTICAL 6.1. OPTIMAL AND SUBOPTIMAL MINIMUM-TIME CONTROLLER

Objectives: The practical consists of several examples devoted to time-sub-optimal controller design.

Application level : 2

The hardware and software configuration for this practical is given in Figure 7-5. Note that the external excitation source is used in this case.

Make all necessary changes in patching of the digital unit: connect the square-wave generator, then set the frequency of the generator to 0.2Hz.

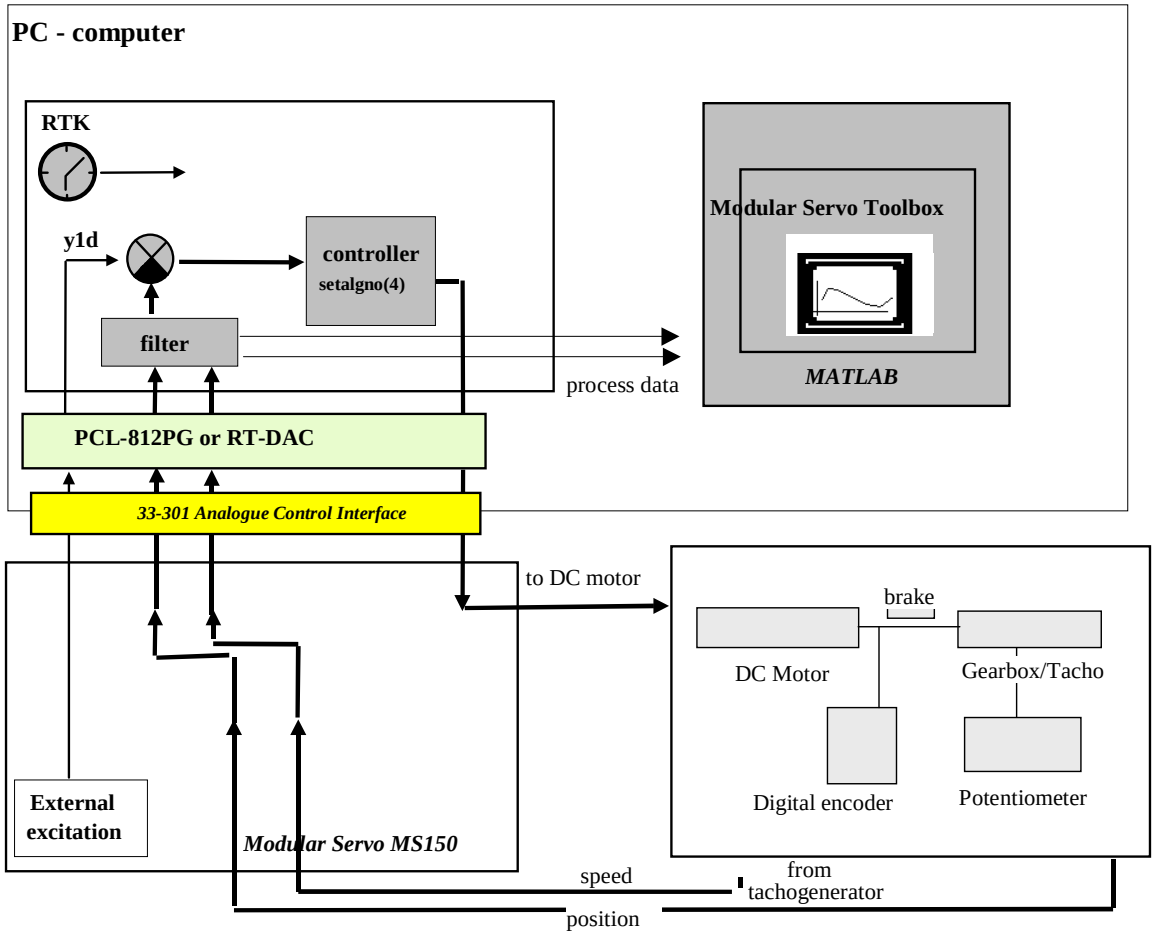


Figure 7-5: Hardware and software configuration for practical 6.1

Example 1

To invoke the time-optimal experiment double click the *Time-optimal controller* block. The window shown in Figure 7-6 opens.

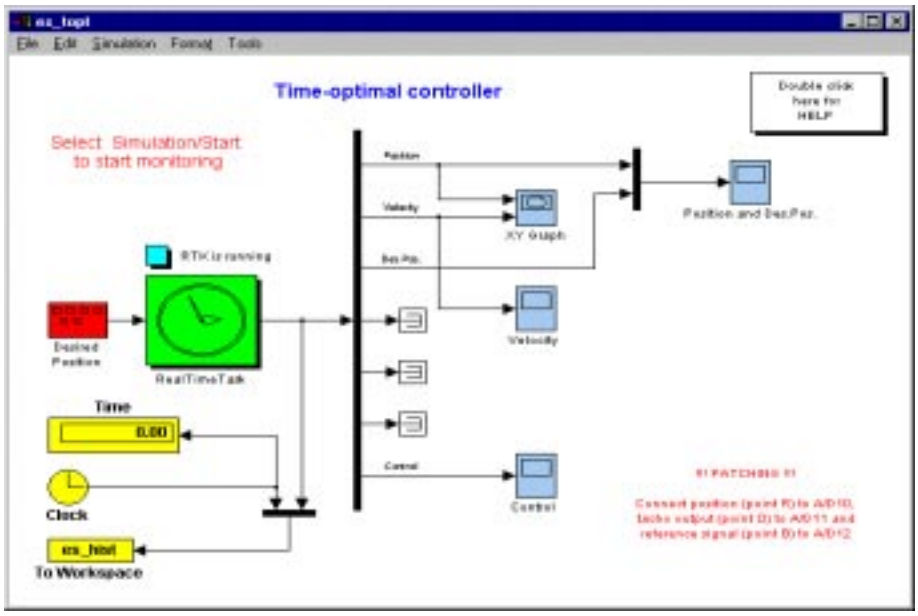


Figure 7-6: *Time optimal controller* window

To start the time-optimal simulation click *Simulation* button. Introduce *Max. Excitation* equal to 1 and start simulation. In the second run set *Max. Excitation* equal to 0.5 and follow by instruction on the screen. The simulation will be carried out and the plot similar to Figure 7-7. will be displayed. Both results are illustrated in Figure 7-7.

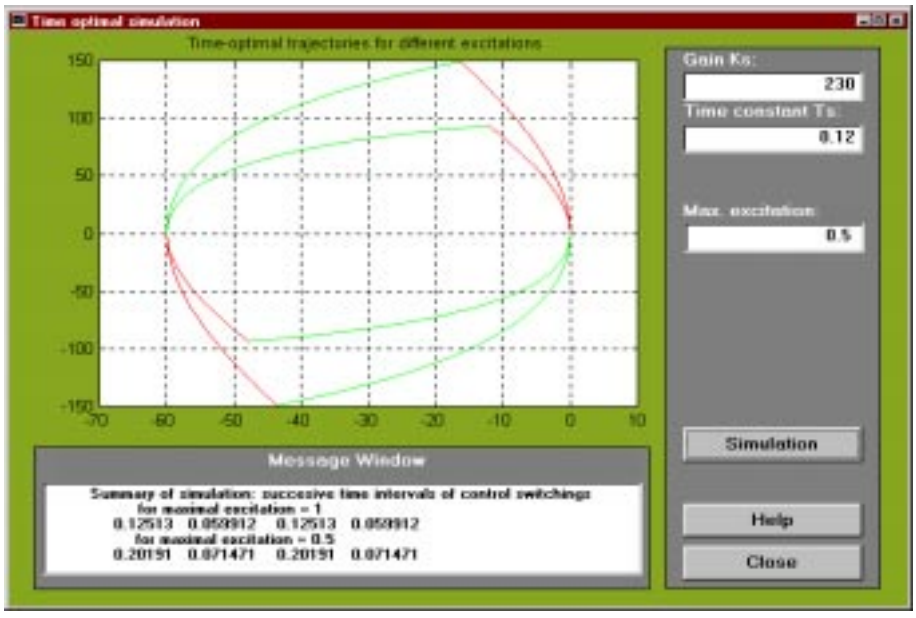


Figure 7-7: Results of time optimal simulation (phase plane)

The servomechanism has two equilibrium points: y_{1d}=0 and y_{1d}=-60. The shorter time interval of motion from one to another equilibrium point corresponds to the greater excitation value.

Comparison of the switching times in the both cases are displayed in the *Message Window*.

Now, close *Simulation* window and start real-time experiments.

In the beginning you can examine the "ideal" time-optimal control. Open the *Real Time Task* block and introduce the following parameters: Umax = 0.8, zone = 0, tune = 0, sampling time = 0.05, downsampling = 1. Next close block and click *Simulation/Start* button.

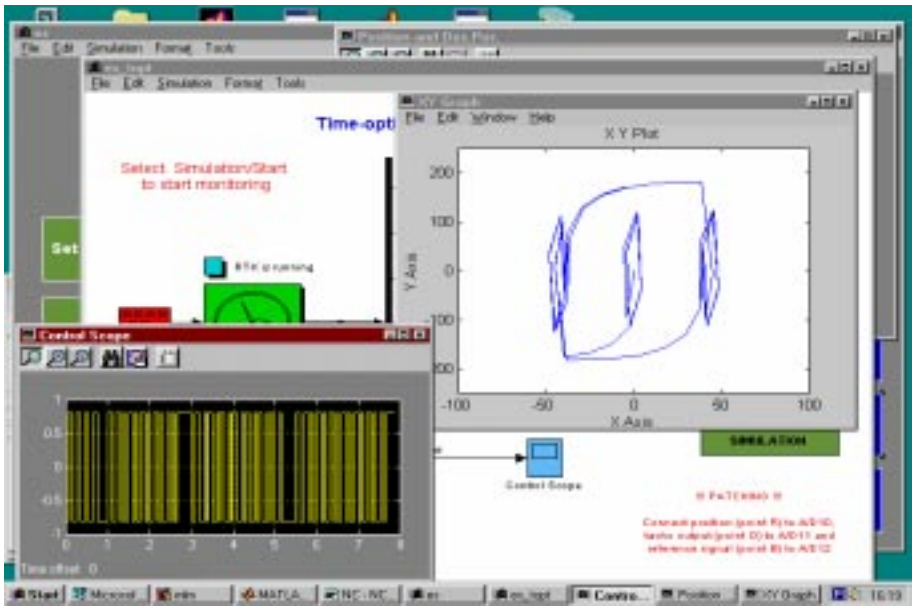


Figure 7-8: The experiment results with the zone = 0, tune =0

You should obtain results similar to Figure 7-8. The servomechanism starts to chatter.

Example 2

Introduce a non-zero parameter zone. For example: zone =15 and Umax = 0.8.

The result is shown in Figure 7-9. The chattering character of time-optimal control (Figure 7-8) has disappeared after introducing the non-zero value of the parameter zone. The value of this parameter can be selected experimentally. Near time-optimal character of control is preserved for a small value of the parameter zone.

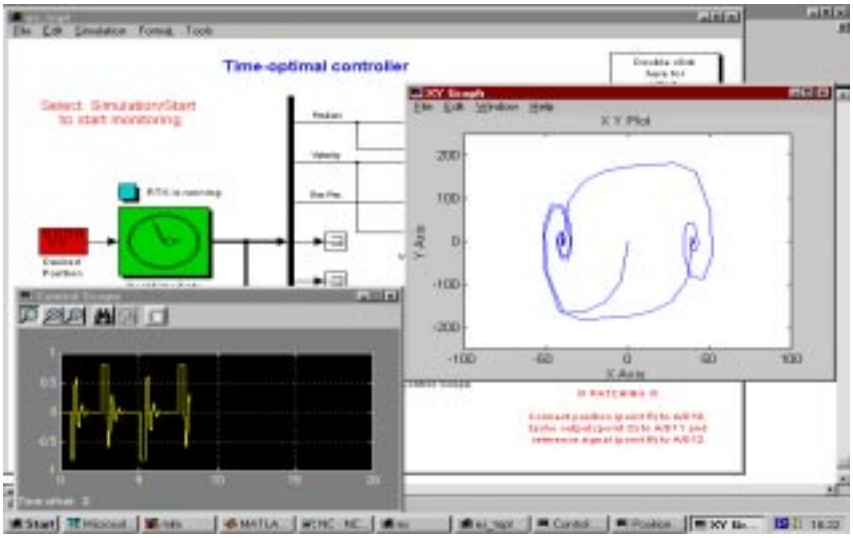


Figure 7-9: The experiment with the zone **=** 15, tune = 0

Example 3

The delays of switching can be seen in Figure 7-9. Introduce a non-zero parameter tune. For example type tune = 6.
The new modified switching curve (given by the formula 6.19) will be obtained. Set the parameters: zone and excitation as in the previous example. You obtain the result similar to that which is given in Figure 7-10.
The time-optimal control is very sensitive to variations of model parameters. To achieve the best performance without chattering of control and with a properly assumed switching curve one must first identify the parameters of DC-motor model: K_S, T_S, C_S and then start to experiment with non-zero values of parameters: *zone* and *tune*.

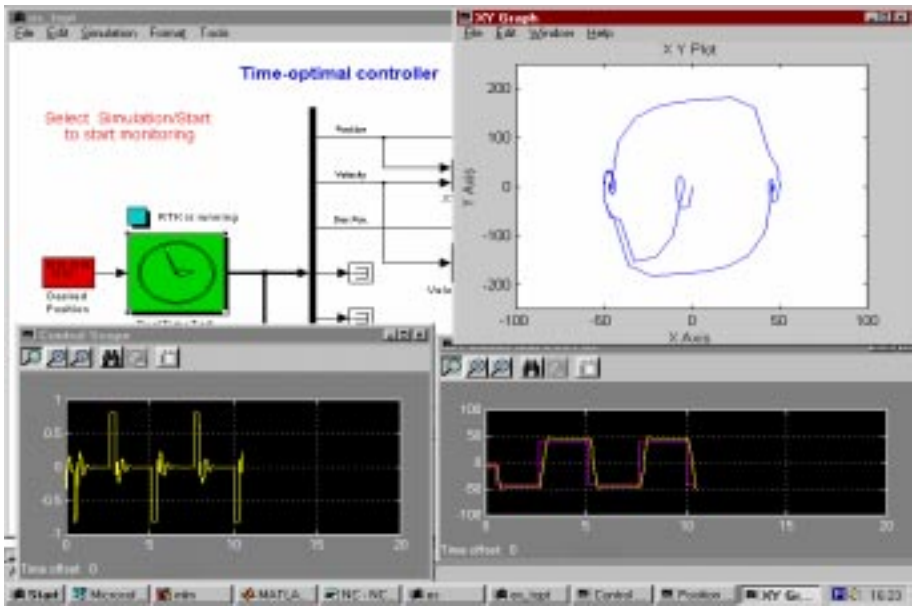


Figure 7-10: The results of the time-optimal experiment with *zone* = 15 and *tune* = 6

8. ASSIGNMENT 7. Model Reference Adaptive Control (MRAC) of the Servomotor

One mode of operation for a servo motor is to trace the desired value. Standard controllers use fixed settings and if the conditions of its operation has been changed the new parameters must be set - usually by operator. The advanced idea is to tune parameters of the controller automatically.

8.1. A SHORT VIEW OF MRAC THEORY

Model reference adaptive control [12], an explicit adaptive technique, has been very attractive from the very beginning of the adaptive control approach. The basic idea of the method is illustrated in Figure 7.1, where a reference model is used to specify the desired performances of the basic loop, consisting of the. controlled process and a controller.

The reference model consists of the servo model and the same type of the controller as used in the basic loop. In Figure 7.1 the classical feedback closed-loop controller is used in the basic loop, but in some cases feedback controllers and pre-filters can be used as well.

The current output of the controlled process is compared with the reference model output and the parameters of the controller are modified in such a way that the response of the controlled process follows that of the reference model.

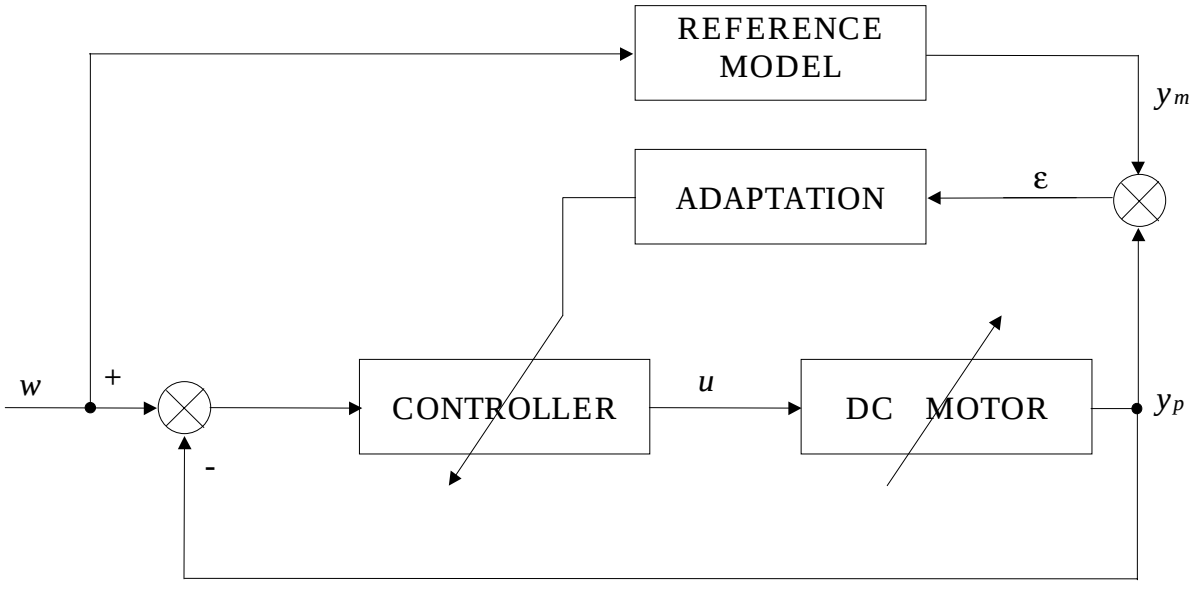


Figure 8-1: Block diagram of a model reference adaptive system (MRAC)

The criterion to be minimised is:

$$I = \frac{1}{2}\int_{t}^{t+\Delta t} \varepsilon^2(\tau)d\tau$$

where:

$$\varepsilon(\tau) = y_p(t) - y_m(t)$$

is the difference of the controlled process outputs and reference model outputs. The adjustment of the classical controller parameters can be made using the steepest descent technique.

8.2. ADAPTIVE CONTROLLER DESIGN

The rewritten transfer function of the DC motor is given by:

$$G(s) = \frac{Y(s)}{U(s)} = \frac{\frac{K_s}{T_s}}{s^2 + \frac{1}{T_s}s}$$

To control the DC motor we use the controller operating according to law:

$$u = fw - q_0\dot{y} - q_1 y$$

There are three parameters:
f proportional to the desired value
q_0 proportional to the DC motor velocity
q_1 proportional to the DC motor position

Figure 8-2: shows the basic control loop with the implemented controller.

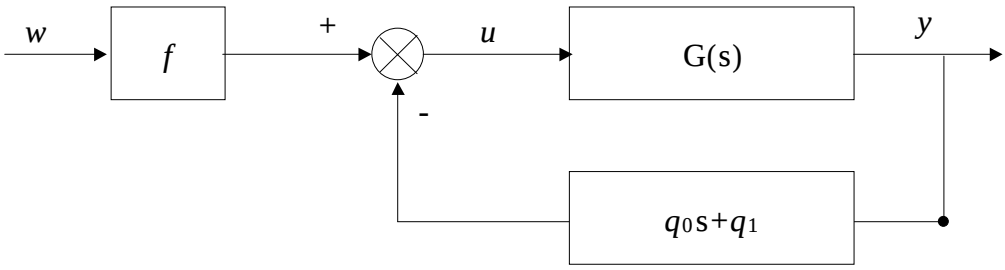


Figure 8-2: The applied control loops

where :

w - desired value
u - control,
y - output (position and velocity),
f, q_0, q_1 - parameters of the controller.

This kind of the controller is easy to implement while designing the adaptive controller. In this case the equation of the controller in the reference model is given by:

$$u_m = f_m w - q_{0m}\dot{y}_m - q_{1m} y_m$$

and the equation of the controller in the basic control loop (the real DC motor is controlled) is given by:

$$u = f_p w - q_{0p}\dot{y}_p - q_{1p} y_p$$

The transfer function of the reference model is given by:

$$G_m(s) = \frac{y_m}{w} = \frac{f_m \dfrac{K_m}{T_m}}{s^2 + \left(\dfrac{1}{T_m} + \dfrac{K_m}{T_m} q_{0m}\right) s + \dfrac{K_m}{T_m} q_{1m}}$$

The transfer function of the basic loop is given by:

$$G_w(s) = \frac{y_p}{w} = \frac{f_p \dfrac{K_p}{T_p}}{s^2 + \left(\dfrac{1}{T_p} + \dfrac{K_p}{T_p} q_{0p}\right) s + \dfrac{K_p}{T_p} q_{1p}}$$

Assume that we attempt to change the parameters of the controller so that the error ε between outputs of the process and the reference model is driven to zero. To make the value of the criterion *I* small it is reasonable to change the parameters in the direction of the negative gradient of *I*.

Applying this concept for the first parameter leads to:

$$\Delta f_p = -g_{f_p} grad_{f_p} I = -g_{f_p} \frac{\partial I}{\partial f_p}$$

For changing speed of the first parameter variations it follows that:

$$\frac{df_p}{dt} = -g_{f_p} \frac{\partial}{\partial t} \frac{\partial I}{\partial f_{f_p}} = -g_{f_p} \frac{\partial}{\partial f_p} \frac{\partial I}{\partial t}$$

If the performance criterion is introduced:

$$\frac{df_p}{dt} = -g_{f_p} \frac{\partial I}{\partial f_{f_p}} \varepsilon^2(t) = -2g_{f_p} \varepsilon(t) \frac{\partial \varepsilon}{\partial f_p} = -2g_{f_p} \varepsilon(t) \frac{\partial y_p}{\partial f_p}$$

Integrating leads to:

$$\mathrm{f_p} = -\mathrm{g_{f_p}} \int_0^t \varepsilon(\mathrm{t}) \frac{\partial \mathrm{y_p}}{\partial \mathrm{f_p}} \mathrm{dt} + \mathrm{f_p}(0)$$

For the remaining parameters we obtain:

$$\mathrm{q_{0p}} = -\mathrm{g_{q_{0p}}} \int_0^t \varepsilon(\mathrm{t}) \frac{\partial \mathrm{y_p}}{\partial \mathrm{q_{0p}}} \mathrm{dt} + \mathrm{q_{0p}}(0)$$

$$\mathrm{q_{1p}} = -\mathrm{g_{q_{1p}}} \int_0^t \varepsilon(\mathrm{t}) \frac{\partial \mathrm{y_p}}{\partial \mathrm{q_{1p}}} \mathrm{dt} + \mathrm{q_{1p}}(0)$$

where the sensitivity functions can be obtained by corresponding derivations and suitable approximations as follows:

$$\frac{\partial y_p}{\partial f_p} = \frac{\partial}{\partial f_p}(G_w(s)w) = \frac{\frac{K_p}{T_p}}{s^2 + \left(\frac{1}{T_p} + \frac{K_p}{T_p} q_{0p}\right)s + \frac{K_p}{T_p} q_{1p}} w = \frac{y_p}{f_p} \approx \frac{K_p T_m}{f_m K_m T_p} y_m$$

$$\frac{\partial y_p}{\partial q_{0p}} = \frac{\partial}{\partial q_{0p}}(G_w(s)w) = -\frac{f_p\left(\frac{K_p}{T_p}\right)^2 s}{\left(s^2 + \left(\frac{1}{T_p} + \frac{K_p}{T_p} q_{0p}\right)s + \frac{K_p}{T_p} q_{1p}\right)^2} w =$$

$$= -\frac{s\frac{K_p}{T_p}}{s^2 + \left(\frac{1}{T_p} + \frac{K_p}{T_p} q_{0p}\right)s + \frac{K_p}{T_p} q_{1p}} y_p \approx -\frac{K_p T_m}{f_m K_m T_p} G_m(s) \cdot s \cdot y_m$$

$$\frac{\partial y_p}{\partial q_{1p}} = \frac{\partial}{\partial q_{1p}}\left(G_w(s)w\right) = -\frac{f_p\left(\frac{K_p}{T_p}\right)^2}{\left(s^2 + \left(\frac{1}{T_p} + \frac{K_p}{T_p}q_{0p}\right)s + \frac{K_p}{T_p}q_{1p}\right)^2}w =$$

$$= -\frac{\frac{K_p}{T_p}}{s^2 + \left(\frac{1}{T_p} + \frac{K_p}{T_p}q_{0p}\right)s + \frac{K_p}{T_p}q_{1p}}y_p \approx -\frac{K_p T_m}{f_m K_m T_p}G_m(s)y_m$$

We do not know the actual parameters of the DC motor, hence we can assume the constants determining the rate of gradient adaptation $g_{q_{frp}}, g_{q_{0rp}}, g_{q_{1rp}}$ as:

$$g_i = g_i \frac{f_m K_m T_p}{K_p T_m}$$

In this case the parameters of the controller are given by:

$$f_p = -g_{f_p}\int_0^t \varepsilon(t) y_m dt + f_p(0)$$

$$q_{0p} = g_{q_{0p}}\int_0^t \varepsilon(t)\cdot s \cdot G_m(s)\cdot y_p dt + q_{0p}(0)$$

$$q_{1p} = g_{q_{1p}}\int_0^t \varepsilon(t)\cdot G_m(s)\cdot y_p dt + q_{1p}(0)$$

8.3. MODEL REFERENCE ADAPTIVE CONTROLLER

The MRAC algorithm was implemented using the EXTERNAL INTERFACE..

To run the adaptive controller type: ***es_mracm*** in Matlab Command Window or click *Adaptive Controller* button in the *Main Control Window.*

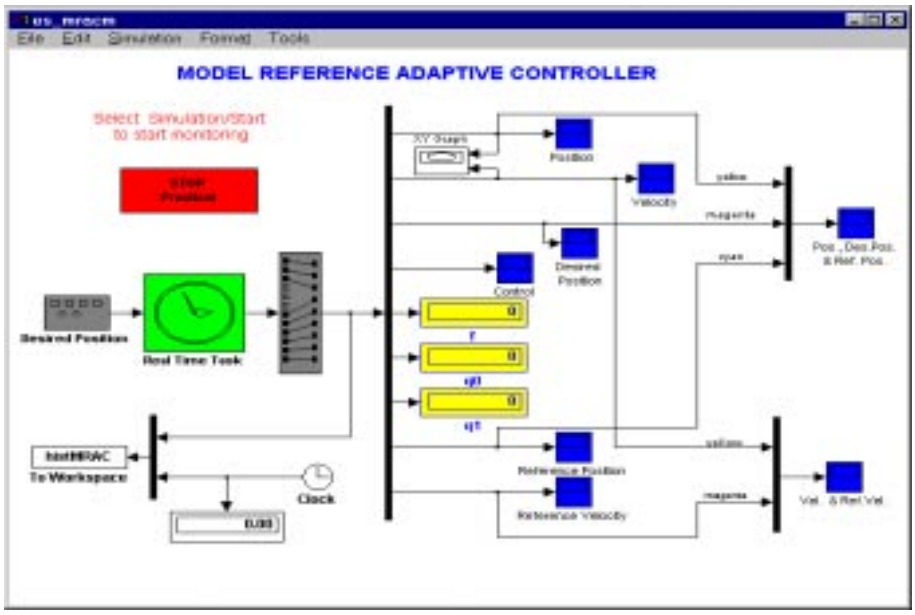


Figure 8-3: Simulink model for the *es_mrac.dll* library (*es_mracm.mdl* file).

The *es_mracsf.m* file contains the S-function associated with the *Real Time Task* block. The function has six parameters:

- *Reference source* - this parameter is set by selecting the appropriate element in the listbox which contains three fields:
  - ◊ *Simulink generator* - this option causes the Simulink signal generator *Desired Position* (see Figure 8-4) to be the source of the reference angle,
  - ◊ *External signal* - the reference signal comes from the MS150 hardware,
  - ◊ *Built-in generator (square)* - the signal generator built into the RTK is the source of the square wave which determines reference value,
- *Reference model parameters [K_m T_m]* - K_m and T_m parameters of the reference model,
- *Reference model controller parameters [f q0 q1]* - Parameters to control the reference model,
- *Constants rate of gradient adaptation [gf gq0 gq1]* - gain factors of gradient adaptation

- *Max control* - maximum value of the control,
- *Downsampling ratio* - this parameter decreases the rate of output data flow from the output of the *Real Time Task* (Figure 8-5). For example, if the downsampling coefficient is 10, for each 10 sampling periods only one is transferred to the output of the block.

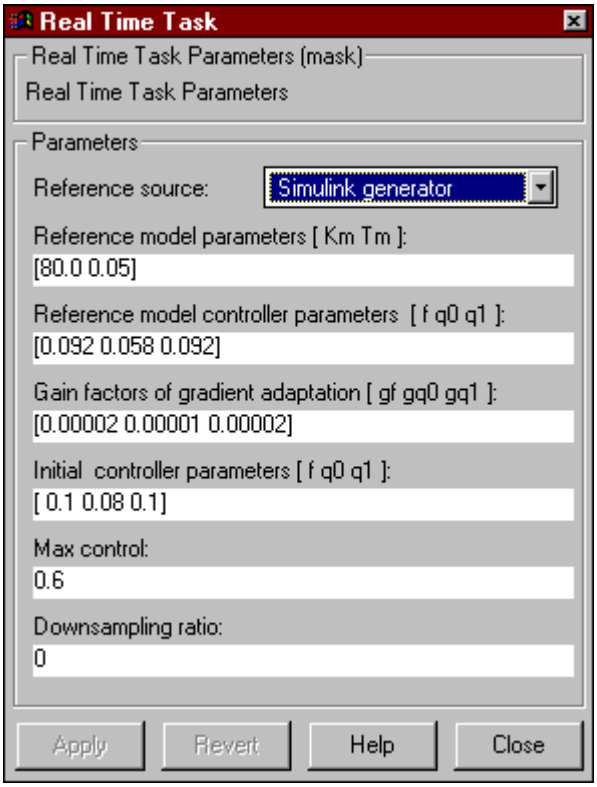


Figure 8-4: Parameters of the *Real-Time Task* block.

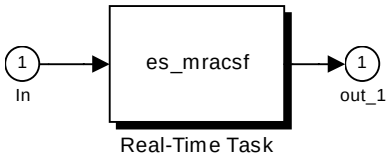


Figure 8-5: The *Real-Time Task* block.

Mask edit fields:

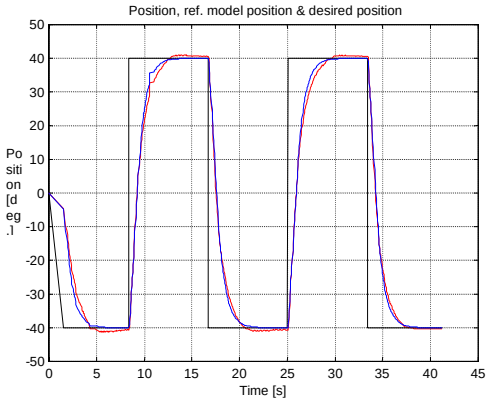
Mask field	Parameter (s)
Reference model parameters [Km Tm]	[K_m T_m]
Reference model controller parameters [f q0 q1]	[f_m q_{0m} q_{1m}]
Gain factors of gradient adaptation [gf gq0 gq1]	[$\bar{g}_{f_p}$ $\bar{g}_{q_{0p}}$ $\bar{g}_{q_{1p}}$]
Initial controller parameters [f q0 q1]	[$f_p(0)$ $q_{0p}(0)$ $q_{1p}(0)$]
Max control [Umax]	|Umax|
Downsampling ratio	see s-function description

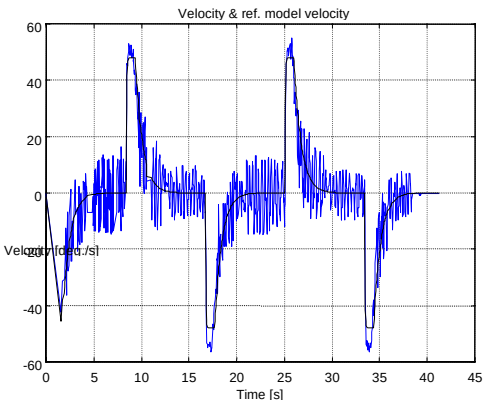
8.4. EXAMPLE

The main goal of this example is to follow the reference position signal set as a square wave.

Set the brake “on” and the start the algorithm with the following parameters (Figure 8-6):

Parameter (s)	Value
[K_m T_m]	[80.0 0.05]
[f_m q_{0m} q_{1m}]	[0.092 0.058 0.092]
[$\bar{g}_{f_p}$ $\bar{g}_{q_{0p}}$ $\bar{g}_{q_{1p}}$]	[0.00002 0.00001 0.00002]
[$f_p(0)$ $q_{0p}(0)$ $q_{1p}(0)$]	[0.1 0.08 0.1]
|Umax|	0.6
Downsampling ratio	0





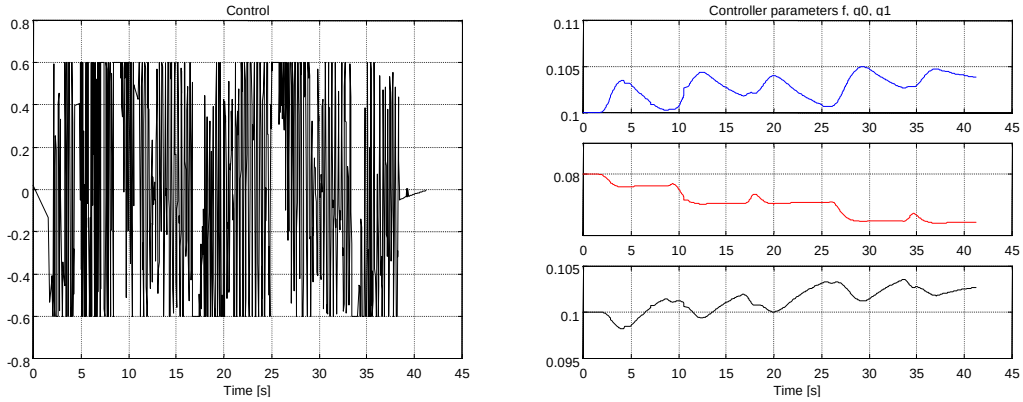


Figure 8-6: Tracking the reference signal (brake is on)

Next, switch off the brake and observe the behaviour of the DC motor (Figure 8-7).

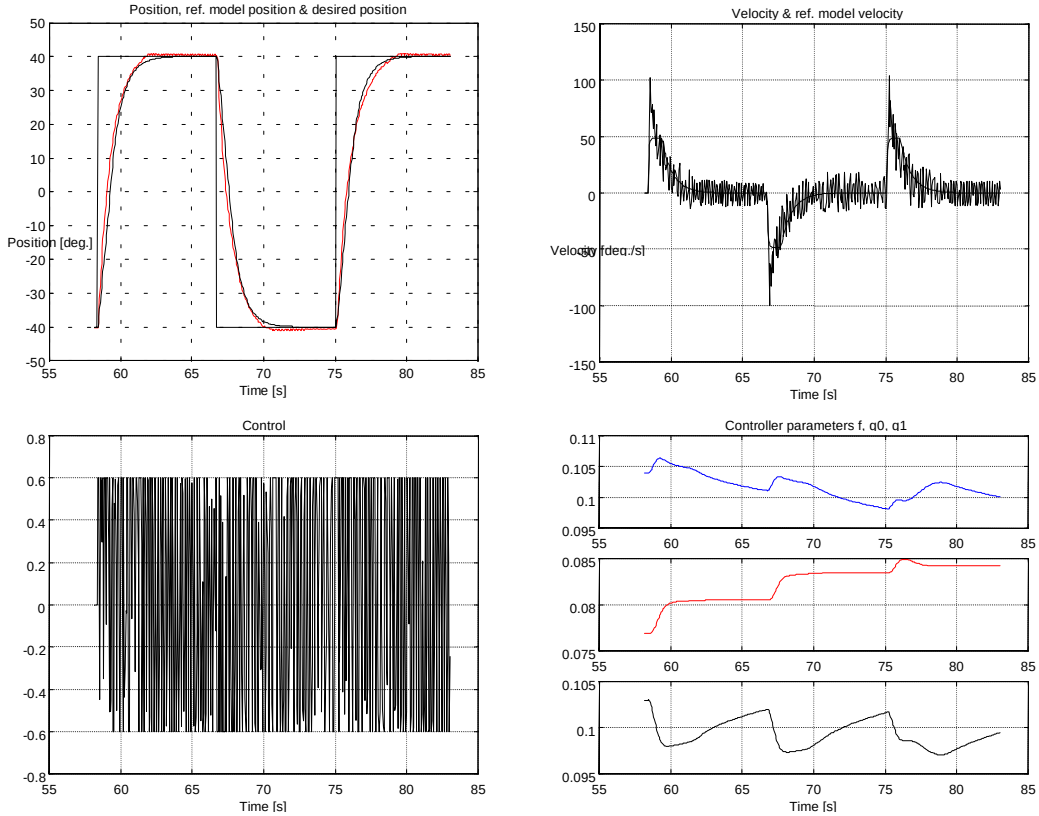


Figure 8-7: Tracking the reference signal before adaptation (brake is off)

Notice, that the parameters of the DC motor controller started to adapt.

After few periods you can observe a better response of the DC motor (Figure 8-8).

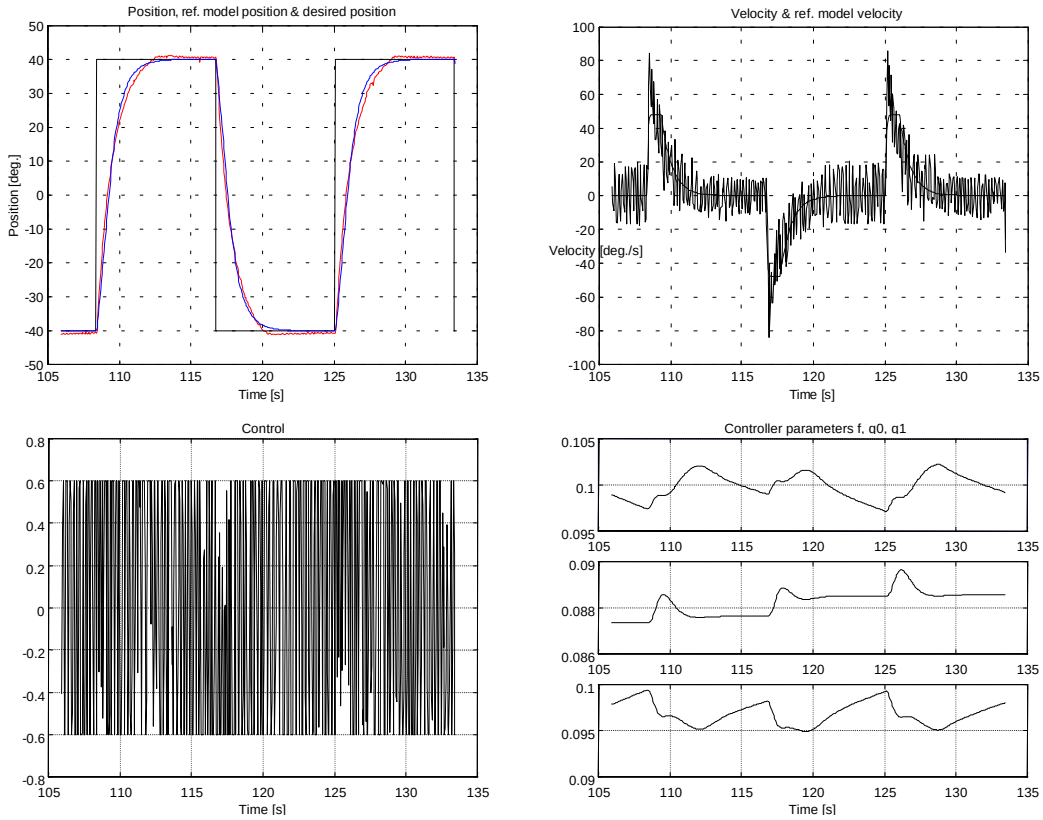


Figure 8-8: Tracking the reference signal after adaptation (brake is off)

The algorithm is ready to make experiments with different model reference parameters, gradient adaptation constants and initial values of the real controller.

9. REFERENCES

This manual refers to the following textbooks and manuals:

[1].Aström K.J. and Wittenmark B., *Computer Controlled Systems,*, Prentice-Hall, 1989.

[2].Franklin G.E., Powell J.D and Workman M.L., *Digital Control of Dynamic Systems,* Addison-Wesley (second edition 1990) .

[3].Iserman R., *Digital Control Systems,* Springer, vol.1 1989, vol2. 1991.

[4].*Modular Servo Workshop - Getting Started -33-008-1M5,* Feedback Ltd, Crowborough, 1998.

[5].*Feedback Servo Fundamentals Trainer 33-003,* FI Ltd, Crowborough, 1993.

[6].Halang W.A., *Real-time systems, World Scientific,* London, 1992.

[7]. MATLAB 5*, Reference Guide,* The MathWorks Inc.,1997.

[8].Takahashi Y., Rabins M., Auslander D., *Control and Dynamic Systems*, Addison-Wesley (1972)

[9]. Strejc V.: *Ausvertung der dynamischen Eigenschaften von Regelstrecken bei gemessen Einund Ausgangssignalen allgemeiner Art.* Zaitschrift fur Messen, Steuern, Regelen, 1960, Heft 1.

[10]. Auslander D.M., Tham C.H., *Real-time software for control,* Prentice Hall, New Jersey, 1990.

[11]. *Modular Servo Workshop - Reference Manual (33-008-2M5) for MATLAB 5,* Feedback Ltd, Crowborough, 1998.

[12]. Iserman R., Lachman K., Matko D. : *Adaptive Control Systems*, Prentice Hall International 1992.