Exercise 8.

Identification and control of linear systems

Introduction

The control design methods presented in the Control Engineering course (serial compensator design, two degrees of freedom controller, state space-based control) are model-based ones. As the name suggests, these methodologies assume that the model of the plant is known, and the design of the controller is based on its parameters.

Process models can be obtained by two ways. At first, they can be derived from physical principles defining the operation of the system (e.g. motion equations or heat transfer laws) and the corresponding parameters (e.g. body mass or heat transfer coefficients) might be measured in an adequate way.

However, in practice there are parameters that can hardly be measured or found in datasheets. For instance, consider a simple temperature control of a room. Based on the physical principles, one can derive the qualitative model (i.e. the transfer function) of the process. However, in the model of the system, the heat transfer coefficient of the heater and the walls and are present, and their measurement is quite difficult. On the other hand, we are able to vary the input signal (power applied to the heater) in a wide range, and we can easily measure the temperature of the room, i.e. we can apply an arbitrary signal to the input of the process and measure its response on the output. Based on these data and the qualitative model, the parameters of the process and therefore the numerical model can be found using adequate identification methods.

One must keep in mind that the identified parameters only approximate (better or less) the real parameters. Also, physical parameters of the process might vary as time passes. Consider the aging of the heating system: as scale appears inside the heating pipes reducing the heat transfer coefficient parameters (and therefore the efficiency) of the system, the model of the process changes. The controller has to be robust enough to handle these parameter changes and show a good performance even though the model of the process used during control design is not fully accurate.

This exercise presents these practical aspects of the material presented in the Control Engineering course. The laboratory exercise follows the path the engineer tracks during the design and implementation of control systems: data acquisition carried out on the plant, identification of the process model, development of a robust control algorithm and its software realization, final test and evaluation of the control system.

Aim of the exercise

The laboratory exercise allows students to try how principles and methods presented in the Control Engineering course can be used in practice, and therefore study the practical aspects of control design. The tasks will be carried out using a physical plant, represented by an electrical circuit, which is controlled by a PC equipped with a data acquisition card with AD and DA converters. Laboratory tasks include

1. measurement of the typical parameters (settling time, damping) of the plant using a signal generator and an oscilloscope
2. selection of the adequate sampling time and excitation signal for identification, computer-based data acquisition and model-fitting
3. specification of the parameters of the desired closed-loop dynamics and design of a state-feedback compensator with state estimator in discrete time
4. verification of the controller by simulation
5. real-time realization of the compensator in state space and evaluation of reference signal tracking, disturbance rejection and robustness properties of the physical control system

These tasks represent the path the control engineering has to track when designing a controller for a partially unknown physical system in the practice. During the laboratory practice, physical instruments like signal generator and oscilloscope and industry-standard software (Matlab, Simulink and Control System Toolbox) will be used.

Required knowledge

To successfully complete the measurement tasks, students need to possess knowledge of the following fields:

- modeling and properties of dynamical systems
- basics of system identification
- control design in state-space

The required knowledge is covered by the Signals and Systems and Control Engineering courses. However, most important parts are briefly summarized in the Preliminaries section of this laboratory guide. (Some paragraphs are placed in boxes, these parts are not necessary to know, but they help to understand the details.)

Read over the preliminaries section carefully, as the teacher might check your knowledge by oral questions or a written test.

Preliminaries

Dynamic properties of control loops

Given is a dynamic system which can be either a plant or the whole closed control loop. The dynamic properties of the system show, how the transient looks like on the output, if the input was changed. The speed of the transient and its overshoot can be characterized by different parameters.

Generally, fast operation and small overshoot (big stability margin) are required in closed loop. Usually, the speed-up results increasing overshoot, hence a good tradeoff should be found between these requirements. One should first examine the dynamic properties of the plant, and based on these parameters realizable specification can be defined for the closed loop behavior.

The properties (speed) of the continuous-time plant have to be taken into consideration at the selection of a sampling time. The sampling time is both required at the identification and the controller design phase.

During this exercise, the students have to estimate the dynamic properties of the plant first, and based on these values, the sampling time and the requirements for the controlled system can be determined.

To examine the dynamic properties, one can use the continuous-time transfer function of the system. A stable pole (a pole placed on the left-hand-side half plane of s) is slower if the real part of the pole is closer to zero. The pole or complex conjugate pair of poles of the *closed loop* which is the closest to the imaginary axis is referred to as the *dominant pole* or *dominant pair of poles* of the closed loop.

As a rule of thumb, one can conclude that if the other poles are placed leftwards to the dominant poles such that the absolute value of their real parts are at least three times the absolute value of real part of the dominant pair of poles, then the transients caused by these poles decay before the first peak of the step response and therefore the dynamic properties of the system are overwhelmingly determined by the dominant pair of poles.

Since a complex pair of poles corresponds to a second order term, the closed loop can be well approximated by the prototype second order term as specified by the dominant pair of poles. A typical pole-zero map of a closed loop with a dominant pair of poles is shown in *Fig. 8-1*.

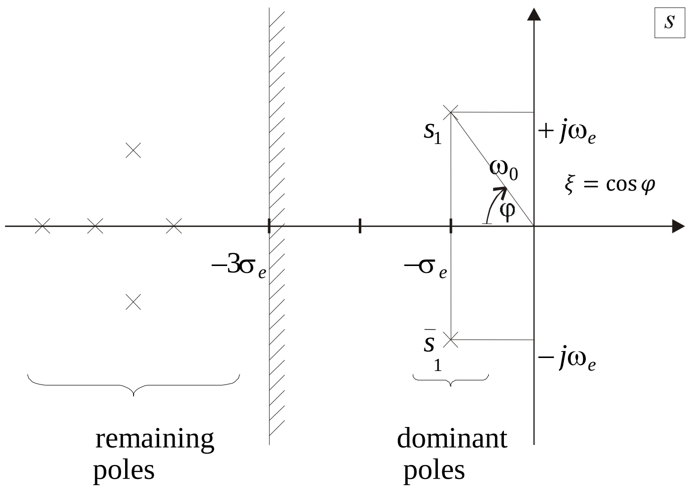


Fig. 8-1 Typical pole-zero map of a closed loop

A typical transient, the step response of the closed loop is shown in *Fig. 8-2*. The figure shows that the *steady-state error* for a step reference signal can be defined by $1-v(\infty)$. The dynamic properties of the closed loop are:

- the overshoot $\Delta v = [v(T_m)-v(\infty)]/v(\infty)$,
- the time to the first peak T_m,
- the settling time $T_{2\%}$
- and the rise time T_{rise}.

There is a direct connection between the dynamic properties and the location of the dominant pair of poles of the closed loop, defined by its undamped natural frequency ω_0 and damping factor ξ (see *Fig. 8-1*):

$$s_{1,2} = -\xi\omega_0 \pm j\omega_0\sqrt{1-\xi^2} = -\sigma_e \pm j\omega_e.$$

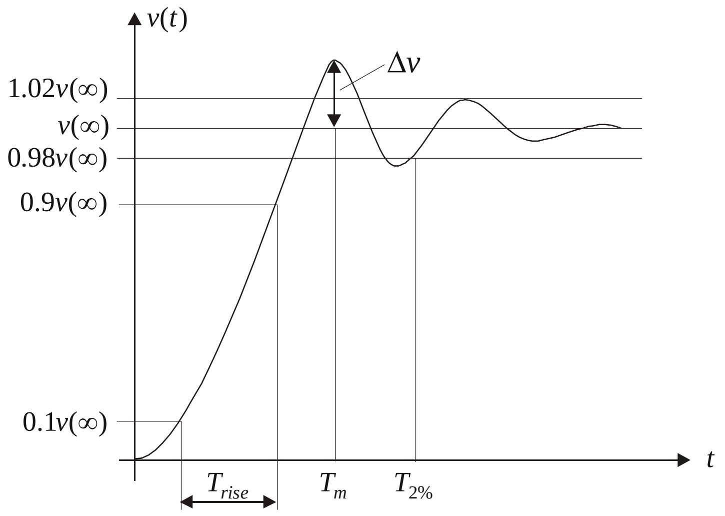


Fig. 8-2 Dynamic properties of the control loop

The dynamic properties of the closed loop can be than approximated using the known formulae for the prototype second order term, e.g. assuming zero steady-state error:

$$v(t) = 1-\frac{\omega_0}{\omega_e}\exp(-\sigma_e t)\sin(\omega_e t+\varphi) =$$

$$= 1-\frac{1}{\sqrt{1-\xi^2}}\exp(-\xi\omega_0 t)\sin(\sqrt{1-\xi^2}\,\omega_0 t + \arccos\xi)$$

$$\Delta v = \exp\left(-\frac{\pi\xi}{\sqrt{1-\xi^2}}\right),$$

$$T_m = \frac{\pi}{\omega_e} = \frac{\pi}{\omega_0\sqrt{1-\xi^2}},$$

$$T_{2\%} = \frac{\ln 50}{\sigma_e} \approx \frac{4}{\sigma_e} \quad \text{and} \quad T_{5\%} = \frac{\ln 20}{\sigma_e} \approx \frac{3}{\sigma_e}.$$

The damping factor ξ of the dominating pair of poles can be obtained from the maximal allowed overshoot of the closed loop, Δv. Usually, the undamped natural frequency is selected as $\omega_0 \approx 5/T_s$, where $T_s = \sum T_i$ denotes the aggregate time constant of the plant.

Our goal is to approximate the best the transfer function $W_{\text{closed}}(s) \approx 1$ in order to assure that the output signal tracks the reference signal with the smallest error possible by setting appropriate damping and frequency parameters.

If the damping factor is fixed, the speed of the transients can be influenced by setting the undamped natural frequency ω_0 of the dominant pair of poles. The marginal frequency of the closed loop (which is approximately the crossover frequency ω_c of the open loop transfer function) is nearly identical to the corner frequency $\omega_0 = 1/T$ of the second order prototype system approximating the closed loop. Therefore, the undamped natural frequency ω_0 of the dominating pair of poles can be determined from the marginal frequency $\omega_h \approx \omega_c$ affecting the speed of closed-loop transients.

However, if we have our model of the plant identified from its step response, and we have determined the time constants T_i from it, then the relative error of the step response of the plant will be significant in the neighborhood of $t \approx 0$. Therefore, if we accelerate the system so much that its transients are significant in the neighborhood of $t \approx 0$, then we might experience undesired closed-loop behavior due to model uncertainties.

In practice, by introducing the aggregate time constant $T_s = \sum T_i$ of the plant (by approximating the denominator of the plant by a first-order Taylor-series, so considering the plant as a first order prototype term with the time constant T_s), the thumb rule of choosing $\omega_0 \approx 5/T_s$ can be used, especially in case of aperiodic systems (systems which can be considered as a serial interconnection of first order terms).

The acceleration of the system is also limited by nonlinearities appearing in case of large signals (saturation etc.) and high frequency disturbances (their effects have to be cut down significantly by the closed loop).

Typical discrete-time process models

To carry out parametric system identification, at first a suitable model class has to be chosen. These classes differ in whether they include any external excitation signal, the or what filter they apply on the input and the noise. The Matlab-based *System Identification Toolbox* (IDENT) developed by Ljung et al defines many of these models, amongst which we will focus only on the ones of discrete-time SISO (Single Input Single Output) systems.

The naming convention of the system models corresponds to their properties. AR stands for Auto Regressive, MA stands for Moving Average, X refers to a model containing eXogenous input signal, OE denotes Output Error (additive error reduced to the output). BJ stands for the Box-Jenkins model while PEM denotes the most general Parameter Estimation Model. The normalized time used for the description of the models is $t = iT$, where T is the sampling time and i is the index of the sampling instance. $x(t)$ is a possible sample-series, while z^{-k} denotes the shift operator defined by $z^{-k}x(t) := x(t-k)$. Although only the ARX and ARMAX models will be used during the exercise, here we present also the most simple and most general models supported by IDENT, namely the AR and PEM models too:

(AR) $$A(z^{-1})y(t) = e(t),$$

(ARX) $$A(z^{-1})y(t) = B(z^{-1})u(t-n_k) + e(t),$$

(ARMAX) $$A(z^{-1})y(t) = B(z^{-1})u(t-n_k) + C(z^{-1})e(t),$$

(PEM) $$A(z^{-1})y(t) = \frac{B(z^{-1})}{F(z^{-1})}u(t-n_k) + \frac{C(z^{-1})}{D(z^{-1})}e(t).$$

The external input (control signal on the input of the plant) is denoted by $u(t)$, the white noise is denoted by $e(t)$ and the output signal with the added noise is denoted by $y(t)$. $A(z^{-1}), B(z^{-1}), C(z^{-1}), D(z^{-1}), F(z^{-1})$ are polynomials of the shift operator z^{-1} and IDENT uses n_k to describe the time delay or time lag ($T_h = n_k T$, where T is the sampling time). The polynomials $A(z^{-1}), C(z^{-1}), D(z^{-1}), F(z^{-1})$ of the models are monic ones, such that $A(z^{-1}) = 1 + a_1 z^{-1} + \cdots + a_{n_a} z^{-n_a}$.

In order to allow an arbitrary gain for the system without noise, $B(z^{-1})$ cannot be monic: $B(z^{-1}) = b_1 + b_2 z^{-1} + \cdots + b_{n_b} z^{-(n_b-1)}$. The gain of the noise channel can be set by the appropriate standard deviation of the white noise $e(t) \in N(0,\sigma)$. IDENT uses the $\lambda := \sigma^2$ parameter instead of the normal deviation. According to the conventions of Matlab, the polynomials $A(z^{-1}), \ldots, F(z^{-1})$ have to be defined by their coefficients in the descending order of the powers of z, e.g. as $A = [1\, a_1\, a_2 \ldots a_{n_a}]$ for $A(z^{-1})$. Note that n_k and B are defined simultaneously by inserting n_k leading zeros into B.

Identification methods

In the sequel, we omit the time delay since it can be modeled by the appropriate shift of the values of $u(t)$ if n_k is known.

Assume that an appropriate model type M has been chosen, and the corresponding $\mathsf{M}(\vartheta)$ models can be parameterized by the parameter vector ϑ, which potentially have to meet given requirements (stability etc.): $\vartheta \in \mathsf{D}_{\mathsf{M}} \subset R^p$. (For example, let the model type M be the AR model with $n_a = 2$, this means $(1 + a_1 z^{-1} + a_2 z^{-2}) y(t) = e(t)$ or in other form $y(t) = e(t) - a_1 y(t-1) - a_2 y(t-2)$, in this case the parameter vector which has to be determined is: $\vartheta = [a_1\ a_2]^T$.)

Clearly, every model provides a possibility for prediction, namely the current output can be predicted based on the actual input and the previous input and output data. Especially, if the model class is

$$y(t) = G(z^{-1},\vartheta) u(t) + H(z^{-1},\vartheta) e(t), \tag{8-1}$$

then the prediction can be carried out with the one-step-ahead predictor defined by

$$\mathsf{M}(\vartheta):\quad \hat{y}(t|t-1) = H^{-1}(z^{-1},\vartheta) G(z^{-1},\vartheta) u(t) + [1 - H^{-1}(z^{-1},\vartheta)] y(t), \tag{8-2}$$

which gives a prediction for $y(t)$ based on the outputs available until time moment $(t-1)$ and the inputs known until time t. The predicted output is denoted by $\hat{y}(t|t-1)$.

Then the prediction error is given by

$$\varepsilon(t,\vartheta) = H^{-1}(z^{-1},\vartheta)[y(t) - G(z^{-1},\vartheta) u(t)] \tag{8-3}$$

The one-step-ahead predictor gives an approximation which is optimal in the estimated value of the square error if the following conditions are met:

$e(t+1)$ and $z(1 - H^{-1}(z^{-1},\vartheta))[y(t) - G(z^{-1},\vartheta) u(t)]$ are independent.

$e(t+1)$ and $G(z^{-1}) u(t+1)$ are independent.

$E\, e(t) = 0,\ \forall t$.

The condition iii) is always fulfilled in case of white noise. If the system has a low-pass characteristics, then the conditions i) and ii) are also fulfilled in case of external noise. However, signals measured in closed loop or quantizing errors can cause problems.

The data available for the parameter estimation is $Z^N = \{y(1), u(1), y(2), u(2), \dots y(N), u(N)\}$, which contains N input-output pairs. The problem is to choose the appropriate parameter vector $\hat{\vartheta}_N$ and therefore the adequate model $\mathsf{M}(\hat{\vartheta}_N)$ from the class of systems $\mathsf{M}_* = \{\mathsf{M}(\vartheta) : \vartheta \in D_{\mathsf{M}}\}$ based on the information in Z^N:

$$Z^N \to \hat{\vartheta}_N \in D_{\mathsf{M}} . \tag{8-4}$$

Such a mapping is referred to as parameter estimation for model identification. We seek such a model which "describes" well the measured data, and we consider that the most important property of the model is its prediction capability. It means that, in order to choose a “good” model, at first the prediction error $\varepsilon(t, \vartheta)$ is determined from the information contained in Z^t. We choose the value of $\hat{\vartheta}_N$ at the time instance $t = N$ such that the prediction errors $\varepsilon(t, \hat{\vartheta}_N),\ t = 1, 2, \dots, N$ would be the smallest possible ones.

However, we shall at first define what do we mean by the word “small”. The series of prediction errors is a vector in R^N, so therefore the error can be characterized by the square of its norm:

$$V_N(\vartheta, Z^N) = \frac{1}{N} \sum_{t=1}^{N} \frac{1}{2} \left| \varepsilon_F(t, \vartheta) \right|^2 . \tag{8-5}$$

The approximation $\hat{\vartheta}_N$ is defined as a solution to an optimization problem:

$$\hat{\vartheta}_N = \arg \min_{\vartheta \in D_{\mathsf{M}}} V_N(\vartheta, Z^N). \tag{8-6}$$

In case of the ARX model, the optimum can be determined algebraically, if we do not restrict the class of models ($\mathsf{D}_{\mathsf{M}} = R^p$). In other cases, a suitable optimization method shall be used, which can use the exact or approximated derivatives of $V_N(\vartheta, Z^N)$.

The IDENT toolbox uses a quasi-Newton method for the identification of ARMAX and even more complex models. The main problem of optimization is that there might be a number of local optima beside the global optimum (especially in case of complex system models), so there is a significant risk of finding not the global, but only a local optimum. Therefore it can be necessary to repeat the optimization several times starting from different initial guesses.

Identification of the ARX model using the least square method (LS)

For an ARX model

$$y(t) = \frac{B(z^{-1})}{A(z^{-1})} u(t) + \frac{1}{A(z^{-1})} e(t) =: G(z^{-1}) u(t) + H(z^{-1}) e(t) . \tag{8-7}$$

The predictor reads

$$\hat{y}(t, \vartheta) = A(z^{-1}) \frac{B(z^{-1})}{A(z^{-1})} u(t) + [1 - A(z^{-1})] y(t) = [1 - A(z^{-1})] y(t) + B(z^{-1}) u(t), \tag{8-8}$$

and let us introduce the notations

$$\varphi^T(t) := [-y(t-1) \dots - y(t-n_a)\ u(t) \dots u(t-n_b+1)] , \tag{8-9}$$

$$\vartheta := (a_1 \dots a_{n_a}\ b_1 \dots b_{n_b})^T \tag{8-10}$$

Then a predictor linear in ϑ can be defined by

$$\hat{y}(t,\vartheta) = \varphi^T(t)\vartheta \,. \tag{8-11}$$

The prediction error and the criterion to minimize are as follows.

$$\varepsilon(t,\vartheta) = y(t) - \varphi^T(t)\vartheta \,, \tag{8-12}$$

$$V_N(\vartheta, Z^N) = \frac{1}{N}\sum_{t=1}^{N}\frac{1}{2}\left|y(t) - \varphi^T(t)\vartheta\right|^2 \,. \tag{8-13}$$

The latter is a linear parameter estimation problem, and its solution can be computed from the condition $V_N'(\vartheta, Z^N) = 0$:

$$V_N'(\vartheta, Z^N) = -\frac{1}{N}\sum_{t=1}^{N}\varphi(t)y(t) + \frac{1}{N}\sum_{t=1}^{N}\varphi(t)\varphi^T(t)\vartheta = 0 \,, \tag{8-14}$$

$$\hat{\vartheta}_N^{LS} := \left[\frac{1}{N}\sum_{t=1}^{N}\varphi(t)\varphi^T(t)\right]^{-1}\frac{1}{N}\sum_{t=1}^{N}\varphi(t)y(t) \,. \tag{8-15}$$

(8-14) and (8-15) can be written in another form as well. Note, that in (8-15) a matrix needs to be invertible, which is obtained from the $\varphi(t)$ vector containing input-output data. If the input for the identification was not selected properly (e.g. u is constant) the resulted matrix will not be invertible.

We suppose, that the observed data have been generated by the noisy system $y(t) = \varphi^T(t)\vartheta_0 + v_0(t)$ corresponding to the real parameter ϑ_0, where $v_0(t)$ denotes the noise. Our goal is that if $N \to \infty$ then $\hat{\vartheta}_N^{LS} \to \vartheta_0$ (i.e. the LS estimation is consistent). One condition is therefore that the measurements $\varphi(t)$ and the noise $v_0(t)$ have to be uncorrelated.

Generally, only the observations $y(t), u(t), 1 \le t \le N$ are available for the computation of the regression vector $\varphi^T(t)$, so the initial values corresponding to the time instances $t \le 0$, necessary for the LS estimation, are missing. The number of these initial values depend on n_a and n_b. Therefore, we consider the data series only from $t = n+1$, where $n = \max\{n_a, n_b - 1\}$. By that way, the problem can be transformed to the original problem by reindexing and redefinition of *N*.

<u>Identification of the ARX model using the instrumental variables method (IV)</u>

As it has been shown, the correlation of the observation $\varphi(t)$ and the noise $v_0(t)$ can cause problems in case of the linear regression model $\hat{y}(t,\vartheta) = \varphi^T(t)\vartheta$. In order to decrease the correlation, we can substitute $\varphi(t)$ by an appropriate instrumental variable $\xi(t)$ at a properly chosen position of the formula of estimation.

The instrumental variable $\xi(t)$ has to be correlated with the $\varphi(t)$ observations, but also has to be uncorrelated with the noise $v_0(t)$.

The IDENT toolbox uses a numerically suitable IV4 algorithm for determining the parameters of the ARX model by the instrumental variable method. The IV4 method determines the suitable instrumental variables $\xi(t)$ and the estimation $\hat{\vartheta}_N$ in four steps.

<u>Identification of the ARMAX model using quasi-Newton method</u>

The IDENT toolbox uses a quasi-Newton optimization method for the identification of ARMAX models given as $A(z^{-1})y(t) = B(z^{-1})u(t) + C(z^{-1})e(t)$. The algorithm of the one-step-ahead predictor is applicable for ARMAX models, so $\hat{y}(t,\vartheta)$ and $\varepsilon(t,\vartheta)$ can be computed. Based on the results, the gradient of the error criterion $V_N'(\vartheta, Z^N)$ can be obtained. The IDENT toolbox uses a multiple-step algorithm for the identification of ARMAX models with an IV4 estimation as initial step.

Services of the IDENT toolbox

Although the identification will be supported by a graphical user interface during the laboratory practice, it is useful to overview the services of the underlying IDENT toolbox. In the followings the data structures and most important function calls of the toolbox will be presented.

The system model is described by a special data structure named "th”. If the system is known (i.e. the coefficients of the polynomials and the square of the standard deviation of the white noise are numerically determined, e.g. because we would like to generate signals for the simulation of a known system in order to test the identification algorithms on a given benchmark system) the IDENT toolbox stores the system to a th-type variable upon the function call

```
th=poly2th(A,B,C,D,F,lambda,T).
```

The latter parameters (C, D, F, λ, T) are optional, the value 1 will be substituted if they are omitted. A noisy or no-noise output of the system can be generated from known input and noise data series u and e by

```
y=idsim([u,e],th)
y=idsim(u,th).
```

Input signals are to be defined as column vectors and resulting output data points will be also returned in a column vector. Appropriate input signals can be generated by, for example, the `rand` and `sign` functions of Matlab.

The identification of an unknown system starts with the measurement of the input signal $u(t)$ and the noisy output signal $y(t)$. These measurements are then collected to the column vectors u and y, and they are concatenated to the matrix $z=[y\ u]$ containing two columns. The identification using the services of IDENT has to be preceded by the selection of the adequate system model class (ARX, ARMAX, OE etc.), the selection of the orders of its polynomials and the specification of the time delay. The aim of identification methods is to obtain the optimal parameters of the polynomials by assuming that the noise $e(t)$ is unknown. The least square (LS) method, or more general parameter estimation techniques, like instrumental variable (IV) methods, numerical optimization or the combination of these methods can be used for finding optimal parameters. The IDENT toolbox chooses the identification method according to the type of the system. The instrumental variable (IV) method can be used only for ARX system models. The number of the nontrivial parameters in the polynomials of the system models, which is the order of the polynomial expect $B(z^{-1})$, is to be given in a column vector, e.g. $nn=[n_a\ n_b\ n_c\ n_d\ n_f\ n_k]$ in case of a PEM model. The zero order is allowed, e.g. if $n_a=0$, then $A(z^{-1})=1$, so therefore it has no effect. If the chosen system model does not contain all polynomials $A(z^{-1}),\ldots,F(z^{-1})$, then the orders of the missing polynomials is forbidden to include in nn. The following polynomial orders have to be given in the order presented for the PEM model:

(AR)	$nn=n_a$
(ARX)	$nn=[n_a\ n_b\ n_k]$
(IV4)	$nn=[n_a\ n_b\ n_k]$
(ARMAX)	$nn=[n_a\ n_b\ n_c\ n_k]$
(PEM)	$nn=[n_a\ n_b\ n_c\ n_d\ n_f\ n_k]$.

Identification methods of IDENT store the results (identified parameters and λ) to the th structure specified by the user:

(AR)	`thar=ar(y,nn)`
(ARX)	`tharx=arx(z,nn)`
(IV4)	`thiv4=iv4(z,nn)`
(ARMAX)	`tharmax=armax(z,nn)`
(PEM)	`thpem=pem(z,nn).`

By considering the non-noisy output of the identified system, or the known noise $e(t)$ in case of simulation experiments, the output of the noisy system can be generated, e.g. in case of a PEM model:

```
y=idsim(u,thpem)
y=idsim([u e],thpem)
```

The results of the identification can be displayed by the command

```
idplot([y u]),
```

which plots both the output and the excitation signal.

Coefficients of the polynomials can be extracted from the th structure, e.g. in case of a PEM model by the function call

```
[A,B,C,D,F]=th2poly(thpem).
```

Measurements and identification results can be displayed on the same plot and can be visually compared by general Matlab services (plot etc.). Note that several functions listed here can be called by a more general way, and they also allow the identification of MIMO (Multiple Input Multiple Output) systems.

The IDENT toolbox provides methods not only for discrete-time parameter identification as presented afore, but it also has several other services. Theoretical background of the algorithms of the IDENT toolbox is given in details in [1].

Requirements for the identified model

Although the identification is carried out on sampled data, and the identified model is also a discrete-time one, the underlying system is a continuous one. Therefore, we require that the identification results for discrete-time linear models correspond to sampled continuous-time (analogous) linear systems. It is known that if s_i is a pole of the continuous-time system, then it is mapped to the discrete-time pole $z_i = e^{s_i T}$ of the sampled system. It means that if z_i is a negative real pole of the identified model with an odd multiplicity, then the model does not correspond to any continuous-time linear system (with the same dimension) since $s_i = \ln(z_i)/T$-valued continuous poles can appear only along with their complex conjugate pairs, which is impossible if z_i is a pole on the negative real axis with odd multiplicity.

The IDENT toolbox assures that (assuming a stable system) the poles appearing outside the unity disk due to numerical inaccuracy return to the region of stability (the unity disk) by substituting z_i^{-1} instead of z_i if $|z_i| > 1$.

However, it is possible that even though the signals of the ARX model, obtained by using the LS method, approximate well the input and output signals measured on the unknown system, some poles of the discrete-time model appear inside the $[-1, 0)$ interval of the negative real axis with odd multiplicity (a reason for such a phenomenon can be a quantization error). In that case the use of more time-consuming but more accurate IV4 or ARMAX models is recommended.

State space-based compensator design in discrete-time

Assuming a known model of the identified plant, compensators can be designed using various techniques. One of these methodologies, which can be considered classical nowadays, is based on the use of state feedback (SF), state observer (SO) and reference gain terms (N_x, N_u) allowing setpoint control. The compensator design method assumes that the system is controllable and observable, i.e. the ranks of the controllability matrix M_c and the observability matrix M_o are the same as the dimension of the state-space of the system.

The state equation of a continuous-time system reads $\dot{x} = Ax + Bu$, $y = Cx$, where the size of the matrices are $A_{n\times n}, B_{n\times r}, C_{m\times n}$. In SISO case $m = r = 1$. The controllability matrix of the system is defined as $M_c = [B\, AB \ldots A^{n-1}B]$ while the observability matrix reads $M_o = [C^T\, A^T C^T \ldots (A^T)^{n-1} C^T]^T$. The requirement for the controllability and observability means that ranks of M_c and M_o equal n.

In discrete time the state equation of the plant reads $x_{i+1} = \Phi x_i + \Gamma u_i$, $y_i = Cx_i$, where $\Phi = e^{AT}$ and $\Gamma = \int_0^T e^{A\sigma} d\sigma B$ assuming that a zero-order hold element was used as digital-analog converter and T denotes the sampling time. Control System Toolbox provides the `c2dm` function with `'zoh'` (default) model for the conversion to such representations. In discrete time the controllability and observability matrices of the system read $M_c = [\Gamma \, \Phi\Gamma \ldots \Phi^{n-1}\Gamma]$ and $M_o = [C^T \, \Phi^T C^T \ldots (\Phi^T)^{n-1} C^T]^T$, respectively.

State space-based compensators can be designed both in continuous- and discrete-time. The Control Engineering course has addressed both techniques. Since the students have to use a discrete-time controller during this measurement task, only the discrete-time state-state based compensator design is presented in the sequel.

Pole placement with state feedback

If the state feedback is defined by $u_i = -Kx_i$, where the size of the gain is $K_{r\times n}$ (which means that the control input is the linear combination of the states), then the state equation of the closed loop is $x_{i+1} = (\Phi - \Gamma K)x_i$ and its characteristic equation reads $\varphi_c(z) = (zI - (\Phi - \Gamma K))$.

It is known that the poles of the transfer function and the eigenvalues of the state matrix are the same. Therefore, if one would like to place the poles $\varphi(z) = \det(zI - \Phi)$ of the plant in order to stabilize and/or accelerate the system in closed loop, one can choose $\varphi_c(z) = (z - z_1)(z - z_2)\cdots(z - z_n) = z^n + p_1 z^{n-1} + \cdots + p_{n-1}z + p_n$ as the characteristic polynomial of the closed-loop system and seek for the state feedback K which assures $\varphi_c(z) = \det(zI - (\Phi - \Gamma K))$ (pole placement problem). The region of stability in z is the unity disk and fast poles are in the neighborhood of $z = 0$ since $s_i \approx -\infty$ is mapped to $z_i \approx e^{-\infty T} = 0$. For SISO systems the algebraic problem can be solved using the Ackermann formula:

$$K = (0 \ldots 0 \; 1) M_c^{-1} \varphi_c(\Phi), \tag{8-16}$$

where $(0 0 \cdots 1) \in R^n$ is a unity row–vector and $\varphi_c(\Phi) = \Phi^n + p_1 \Phi^{n-1} + \cdots + p_{n-1}\Phi + p_n I$. In the sequel the use of the Ackermann formula will be denoted by

$$(\Phi, \Gamma) \xrightarrow[M_c]{\varphi_c(z)} K. \tag{8-17}$$

Control System Toolbox provides the `acker` function for the computation of K using the Ackermann formula.

Actual observer

Since the state x appearing in the state-feedback is usually not available for measurement (sensors measure only the output y), it has to be substituted by a suitable approximation $\hat{x}$. If the signals are deterministic, then the term calculating $\hat{x}$ is referred to as the state observer (in case of stochastic signals the notation state estimator is used, and the term is a Kalman filter in the majority of cases).

In discrete time it is practical to exploit the fact that the measurement y_i is available at the time instant $t = iT$, so by taking this property into consideration a delay of one sampling period can be eliminated in the controller. This strategy is not applicable on the input, since an algebraic loop would appear in the system, making the computations more difficult.

Let us consider that $(\Phi, C\Phi)$ is observable, e.g. (Φ, C) is observable and $\exists \Phi^{-1}$, i.e. there exists the inverse of the matrix Φ. The actual observer is a diskrete-time, linear time-invariant (LTI) dynamic system with the estimated state $\hat{x}$ as its output:

$$\hat{x}_i = F\hat{x}_{i-1} + Gy_i + Hu_{i-1}. \tag{8-18}$$

The estimation error $\tilde{x} = x - \hat{x}$ reads

$$\tilde{x}_i = F\left(x_{i-1} - \hat{x}_{i-1}\right) + \left(\Gamma - GC\Gamma - H\right)u_{i-1} + \left(\Phi - GC\Phi - F\right)x_{i-1}. \tag{8-19}$$

so we can obtain an asymptotic state observer if we choose the following parameters in order to assure $\tilde{x}_i \to 0$:

$$F = \Phi - GC\Phi, \quad H = \Gamma - GC\Gamma, \quad \tilde{x}_i = F\tilde{x}_{i-1} \textit{ is stable and fast} \tag{8-20}$$

If the speed of the transient of the observer is prescribed by its characteristic polynomial $\varphi_o(z)$, where $\varphi_o(z) = \det(zI - F) = \det(zI - (\Phi - GC\Phi))$, then G and F can be calculated by using the Ackermann-formula (or any other equivalent method) in SISO case:

$$(\Phi, C)_I \leftrightarrow \left(\Phi^T, \Phi^T C^T\right)_{II} \xrightarrow[M_{c,II}]{\varphi_o(z)} K_{II} \to G = K_{II}^T \to F = \Phi - GC\Phi \tag{8-21}$$

Finally, H can be computed if G is known.

Therefore, if the roots of the characteristic equation $\varphi_o(z) = \det(zI - F)$, corresponding to the decay of estimator error, are chosen to be suitably fast compared to the roots of $\varphi_c(z)$, then the transients of the estimator decay rapidly and (if the model of the system was adequate) the error of the state estimator vanishes.

The computation of $\hat{x}_i$ can be transformed into a form more suitable for real-time realization since $\hat{x}_i = \Phi\hat{x}_{i-1} + \Gamma u_{i-1} + G\{y_i - C(\Phi\hat{x}_{i-1} + \Gamma u_{i-1})\}$. Note that the computation of $\Phi\hat{x}_{i-1} + \Gamma u_{i-1}$ can be executed directly after the latest sampling while Gy_i can be computed at the next sampling instance after measuring y_i. It results efficient CPU-use, especially if $n = \dim x$ is large. By introducing x_i (which can be computed between two sampling instances) the state equation of the actual observer can be transformed into the following form:

$$\begin{aligned} x_i &= \Phi\hat{x}_{i-1} + \Gamma u_{i-1} && \text{"time-update"} \\ \hat{x}_i &= x_i + G\left(y_i - C x_i\right) && \text{"measurement-update"} \end{aligned} \tag{8-22}$$

Setpoint control

In the previous discussions on the controller design, we have assumed zero reference signal, which is unrealistic. Let us denote the reference signal by r, and assume that it is constant (or more precisely, it changes rarely and stays constant afterwards). We would like to assure that the difference $N_x r - x_\infty$ is zero in steady state and that there is no steady-state error on the output, i.e. $y_\infty = r$. The required control signal will be provided by $u_\infty = N_u r$ in steady state. Since the correction contains only feed-forward terms with respect to the state feedback, it leaves the characteristic equation of the closed loop unchanged, i.e. $\varphi_c(z) = \det(zI - (\Phi - \Gamma K))$. If $\dim y = \dim r = \dim u = m$, then the size of the matrices N_x and N_u is $n \times m$ and $m \times m$, respectively (in SISO case $m = 1$). We take into consideration that the solution of the state equation in steady state is characterized by the equivalence of the previous and actual states in discrete time: $x_\infty = \Phi x_\infty + \Gamma u_\infty \Leftrightarrow (\Phi - I)x_\infty + \Gamma u_\infty = 0$, so therefore

$$\begin{pmatrix} N_x \\ N_u \end{pmatrix} = \begin{bmatrix} \Phi - I & \Gamma \\ C & 0 \end{bmatrix}^{-1} \begin{pmatrix} 0_{n\times m} \\ I_m \end{pmatrix}. \tag{8-23}$$

Load estimation

We have seen, that the states can be observed based on the input and output of the plant. Now we extend the state observer to be able to estimate an unknown disturbance. Using the estimated load value, its compensation can be carried out.

The disturbance is assumed to be reduced to the input of the plant ("load change”). If we have knowledge about the characteristics of the disturbance, then it can be modeled. We may assume for instance that the disturbance is constant, so its difference equation is $d_{i+1} = d_i$ (d is a constant with unknown value). If we augment the system with the state variable $x_d = d$, and we introduce the notation $\tilde{x} = (x^T, x_d^T)^T$ then the extended state equation reads

$$\begin{pmatrix} x_{i+1} \\ x_{d,i+1} \end{pmatrix} = \begin{bmatrix} \Phi & \Gamma \\ 0 & I \end{bmatrix} \begin{pmatrix} x_i \\ x_{d,i} \end{pmatrix} + \begin{bmatrix} \Gamma \\ 0 \end{bmatrix} u_i \Rightarrow \tilde{x}_{i+1} = \tilde{\Phi}\tilde{x}_i + \tilde{\Gamma} u_i,$$
$$y_i = \begin{bmatrix} C & 0 \end{bmatrix} \begin{pmatrix} x_i \\ x_{d,i} \end{pmatrix} \Rightarrow y_i = \tilde{C}\tilde{x}_i. \tag{8-24}$$

Since the system $(\tilde{\Phi}, \tilde{\Gamma}, \tilde{C})$ is not controllable (the external signal x_d obviously cannot be controlled internally by u), we have to obtain the state feedback and the reference gains allowing setpoint control for the original system. However, the observer has to be designed for the extended system in order to make it capable of estimating both $\hat{x}$ and $\hat{x}_d$:

$$(\Phi, \Gamma, C) \to K \ (SF)$$
$$(\tilde{\Phi}, \tilde{\Gamma}, \tilde{C}) \to (\tilde{F}, \tilde{G}, \tilde{H}) \ (\mathrm{S}O) \tag{8-25}$$
$$(\Phi, \Gamma, C) \to (N_x, N_u)$$

The realization of the control loop including load estimation is illustrated by *Fig. 8-3.*

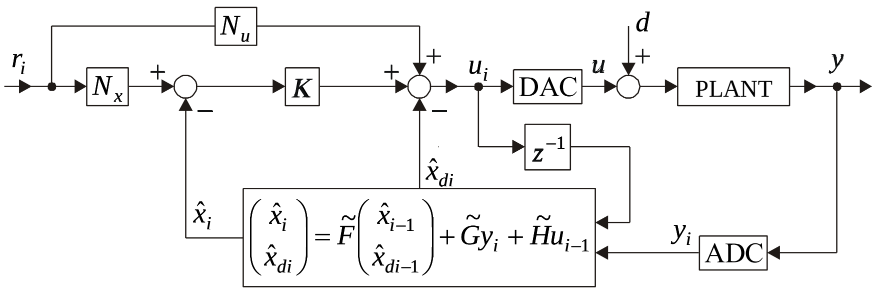

N_u
r_i
N_x
+
−
K
+
+
−
u_i
DAC
u
d
+
PLANT
y
z^{-1}
$\hat{x}_{di}$
$\hat{x}_i$
$\begin{pmatrix} \hat{x}_i \\ \hat{x}_{di} \end{pmatrix} = \tilde{F} \begin{pmatrix} \hat{x}_{i-1} \\ \hat{x}_{di-1} \end{pmatrix} + \tilde{G} y_i + \tilde{H} u_{i-1}$
y_i
ADC

Fig. 8-3. Control with load estimation in discrete time.

The essence of load estimation is that after the decay of the transients of the observer the term $-\hat{x}_d$ added to the output of the controller and the disturbance d on the input of the plant compensate each other, and therefore the system is operating as in case of no disturbance. However, an adequately exact model of the system is indispensable. Nevertheless, the load estimator is able to compensate parameter changes.

For further details and illustrative examples, see [2].

References

[1] Ljung,L: *System identification: Theory for the user*, Prentice Hall, 1987

[2] Lantos - Kiss - Harmati *Practices in Control Theory*, Department of Control Engineering and Information Technology, Budapest University of Technology and Economics, 2008.

Practice 5 (Design of continuous-time controllers in state-space),

Practice 6 (Design of discrete-time controllers in state space),

Practice 7 (Identification algorithms)

Measurement Instruments

Beside the well-known instruments (function generator and oscilloscope), a special test panel representing the plant to be identified and two special software products will be used during the laboratory practice. The PC running these software is interfaced to the test panel by an Advantech PCI-1711 data acquisition card.

Test panel: input scaling circuit and plant

The test panel comprises a scaling circuit and the plant itself. The scaling circuit converts the inputs to the [-10V, +10V] operating voltage range of the plant from the [0V, +10V] voltage range of the Advantech PCI-1711 data acquisition card. Most relevant tasks of the exercise are to carry out on the plant.

The plant is a third order linear dynamical system realized as an electronic circuit, and is the serial interconnection of a first order and a prototype second order term, i.e. its transfer function reads

$$W(s) = \frac{A_3}{(1 + 2\xi T_2 s + T_2^2 s^2)(1 + sT_3)}.$$

The system shows linear time-invariant behavior in a wide range of signals with a good approximation. However, it has an output offset in case of zero input.

The front panel of the device is shown by *Fig. 8-4.* The plant can operate in more than hundred operating modes, which realize systems with different parameters in the class of third-order systems. During the exercise the parameter values are unknown, and they have to be identified.

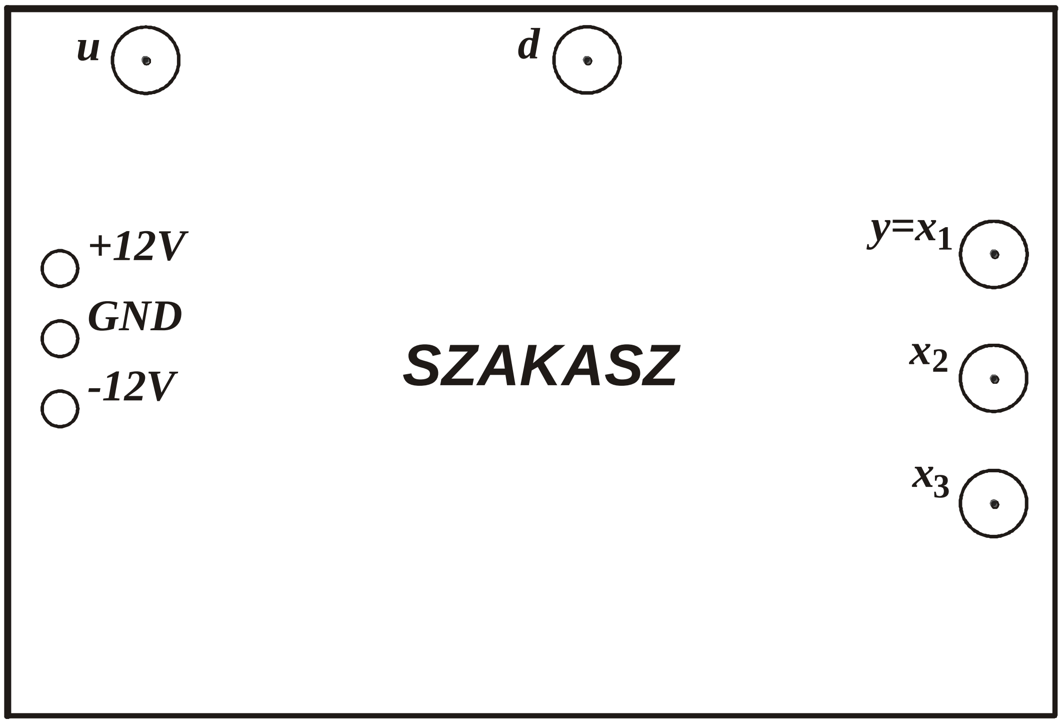


Fig. 8-4. Front panel of the plant (SZAKASZ in English means PLANT)

The identification has to be carried out on the nominal system, and the assigned controller is also to be designed for the nominal system. Experiments (simulation and real-time experiments) should be carried out on the nominal system at first. Afterwards, the experiments should be repeated on the plant with perturbed parameters (small parameter changes in negative and/or positive directions) in order to verify whether the controller is robust enough to deal with parameter changes.

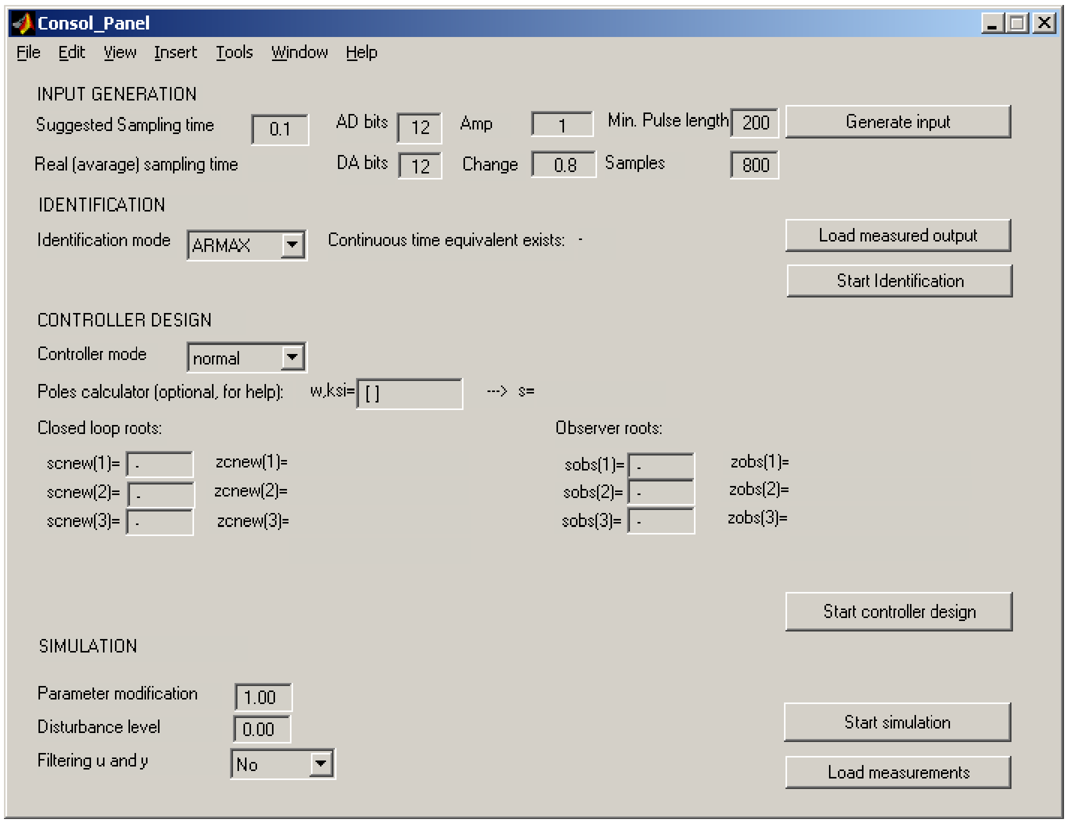
The plant is powered by the power supply available at the lab (plugs at the left-hand side of the front panel). Take care of connecting it with right polarity and accurate voltage settings (+12V, -12V).

The plant has two inputs: *u* and *d* (BNC connectors), where *u* is the control signal and *d* is the external disturbance on the input of the plant. All three states of the plant are available for measurement (BNC connectors at the right-hand side of the front panel), amongst which x_1 is assumed to be the output. The voltages applied to the input can vary between 0 and +10 V and these levels are converted to the [-10 V,+10 V] range by the scaling circuit of the panel. This conversion, originated from the properties of the Advantech data acquisition card, remains hidden since also Matlab and Windows-based software modules use input signal in the [-10 V, +10 V] range. The voltage measured on the outputs varies in the [-10 V,+10 V] range. Signals with higher amplitudes result saturation and the plant leaves the region of linear operation hence such high amplitudes should be avoided.

The offset of the output is compensated automatically by a constant term added to the controller output in the real-time control algorithm.

The Consol Matlab application

The *Consol* application, running in Matlab environment, is responsible for coordination of the identification, controller design, simulation and evaluation routines. The program can be started from Matlab Command Window by typing the command `consol`. The features of the application can be accessed from the *Consol_Panel* window, shown by *Fig. 8-5.*



INPUT GENERATION

Suggested Sampling time 0.1 | AD bits 12 | Amp 1 | Min. Pulse length 200

Real (avarage) sampling time | DA bits 12 | Change 0.8 | Samples 800

IDENTIFICATION

Identification mode ARMAX | Continuous time equivalent exists: -

CONTROLLER DESIGN

Controller mode normal

Poles calculator (optional, for help): w,ksi= [ ] ---> s=

Closed loop roots: | Observer roots:

scnew(1)= - | zcnew(1)= | sobs(1)= - | zobs(1)=

scnew(2)= - | zcnew(2)= | sobs(2)= - | zobs(2)=

scnew(3)= - | zcnew(3)= | sobs(3)= - | zobs(3)=

SIMULATION

Parameter modification 1.00

Disturbance level 0.00

Filtering u and y No

Fig. 8-5. Graphical User Interface of the Matlab application

The figure shows that control engineering tasks to be carried out in Matlab environment can be divided into four major parts.

INPUT GENERATION

This part deals with the generation and recording of the input signal necessary for identification. The signal generated is a series of varying-length square pulses which approximates a pseudo-random signal such that the minimal length of the constant periods can be set. If the default signal is not adequate, the parameters of the input signal can be redefined.

Suggested Sampling Time

The suggested sampling time passed to the *RT_DataAqu_Control* application. The user can set here the time between two samplings (in ideal case). However, due to the properties of Windows scheduler, it cannot be assured that the suggested sampling time is respected exactly by the *RT_DataAqu_Control* application.

Real (avarage) sampling time

This parameter is the average sampling time which *RT_DataAqu_Control* was able to achieve in Windows environment. In the followings, controller design is carried out using this sampling time. The generation of the reference signal for *RT_DataAqu_Control* is carried out using the *Suggested Sampling Time*, which will realize a sampling time approximately identical to *Real (avarage) sampling time*.

Amp

Magnitude of the excitation square signal. The signal is symmetric, i.e. it varies between +Amp and –Amp.

Min. Pulse length

Defines the minimal pulse length in multiple of sampling periods i.e. gives at least how many sampling instances elapse between two amplitude changes.

Change

Probability of the change of the input signal at a given time instance: a real number between 0 and 1. If the parameter is set to e.g. 0.3, then at each sampling instance, there is a 30% chance that the signal will change from *–Amp* to *+Amp* (or *+Amp* to *–Amp*). This probability is overridden (masked) if the number of samplings between the actual sampling instance and the last amplitude change is smaller than *Min. Pulse length*, since the input signal cannot change in that case.

Samples

Defines the length of the input signal in sampling instances.

ADbits

Defines the number of bits of the AD converter on the data acquisition board. It has significance only in case of simulation, where the plant is also simulated. It allows to cut out quantization (choosing high number of bits) or to check the effect of quantization on the properties of control. This parameter has no effect during the measurements since the data acquisition card has a 12 bit ADC.

DAbits

Used for the study of quantization effects during the simulation. During the data acquisition for identification or during the control, the number of bits of the DAC on the data acquisition board has to be given here. Based on it, the Matlab software passes a signal to *RT_DataAqu_Control* with a magnitude not exactly *Amp* but its adequately quantized value.

After setting the aforementioned parameters, the input signal can be generated by clicking on the *Generate input* button. The signal will be displayed in the window *Figure No.1*. If the signal does not fulfill the planned requirements, a new input signal can be generated by applying new parameter settings and clicking again on the *Generate input* button.

IDENTIFICATION

This part deals with the parameter settings for identification and the execution of the identification procedure.

Load measured output

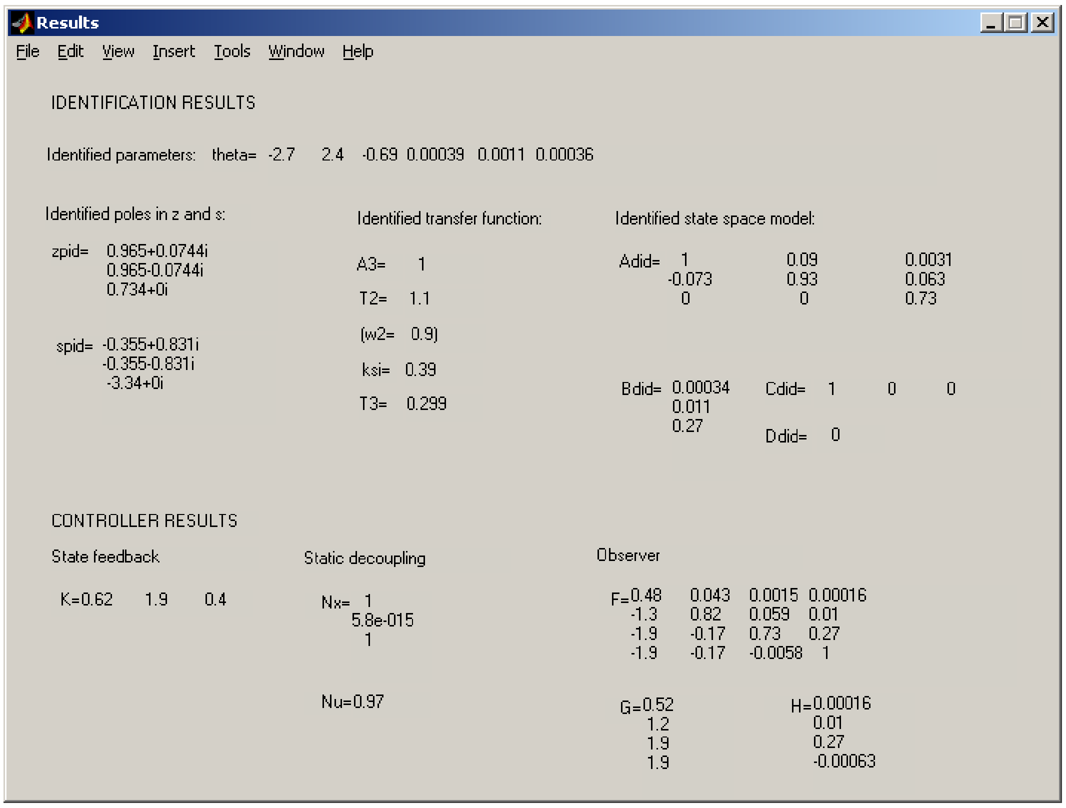
By clicking on the *Load measured output* button, the measurements of the output signal recorded by *RT_DataAqu_Control* are loaded into Matlab workspace. After that the input signal and the response of the plant are displayed in the window *Figure No. 1*. The value of *Real (avarage) sampling time* is determined simultaneously.

Identification mode

This allows the selection of the used identification model/method from the set LS, ARX, IV4 and ARMAX.

Start identification

The identification of the model, based on the input signal used during data acquisition and the measured output signal, can be initiated by clicking on the *Start identification* button. Results are displayed at the IDENTIFICATION RESULTS block of the *Results* window (see *Fig. 8-6*).



```
IDENTIFICATION RESULTS

Identified parameters:  theta=  -2.7    2.4   -0.69  0.00039   0.0011   0.00036

Identified poles in z and s:        Identified transfer function:      Identified state space model:

zpid=   0.965+0.0744i               A3=    1                            Adid=   1          0.09        0.0031
        0.965-0.0744i                                                          -0.073      0.93        0.063
        0.734+0i                    T2=    1.1                                  0          0           0.73

                                    (w2=   0.9)
 spid=  -0.355+0.831i
        -0.355-0.831i               ksi=   0.39
        -3.34+0i                                                        Bdid=  0.00034    Cdid=   1      0      0
                                    T3=    0.299                               0.011
                                                                               0.27       Ddid=   0


CONTROLLER RESULTS
State feedback                 Static decoupling                Observer

 K=0.62    1.9     0.4           Nx=   1                          F=0.48    0.043   0.0015  0.00016
                                     5.8e-015                      -1.3     0.82    0.059   0.01
                                     1                             -1.9    -0.17    0.73    0.27
                                                                   -1.9    -0.17   -0.0058  1


                                 Nu=0.97                          G=0.52              H=0.00016
                                                                    1.2                 0.01
                                                                    1.9                 0.27
                                                                    1.9                 -0.00063
```

Fig. 8-6. Window displaying the parameters of the identified model and the controller

The identified parameters are contained by the vector theta. The variables zpid and spid contain the poles of the identified model in z and in s domains, respectively. The A3 parameter gives the DC gain of the continuous model corresponding to the identified discrete-time one. T2 and ksi represent the time constant (reciprocal of the undamped natural frequency) and the damping of the second-order term of the model, respectively. T3 is the time constant of the first-order term of the model.

Consol_Panel (and also the *Results* window) displays whether there exists a continuous-time model corresponding to the identified one (identification is carried out in discrete time). If not, the T3 time constant is corrected, which is indicated in the *Results* window. After the execution of identification, the identification error, which is the difference of the output signals of the real plant and that of the identified model, is displayed in the *Figure No. 2.* window. Then the input signal, the measured output signal and the output signal of the identified model are plotted in the window *Figure No. 3.*

CONTROLLER DESIGN

This block designs a discrete-time controller using state-space based techniques for the identified model, according to the prescribed specifications.

Controller mode

Defines whether the controller contains normal state feedback (*normal*), state feedback with integrating control *(integral)* or state feedback with load estimation *(load).*

Poles calculator (optional, for assistance)

The undamped natural frequency (ω) and damping (ξ) of the dominant pair of poles can be given in a vector according to the conventions of Matlab. The software computes the pole itself (one of the pair of poles and its complex conjugate) based on these parameters. This function is useful for the specification of poles (scnew, sobs vectors).

scnew(1), scnew(2), scnew(3), scnew(4)

The prescribed continuous-time poles (dominant pair and additional poles) of the closed loop can be defined here. In case of integrating control, four poles are necessary. Otherwise, only three poles are needed. If a complex pole is specified, then its conjugate will also appear. After giving the poles (and pressing enter or leaving the edit window), the software calculates the corresponding discrete-time poles and displays them in the *zcnew* textbox.

sobs(1), sobs(2), sobs(3), sobs(4)

The prescribed continuous-time eigenvalues (dominant pair and additional poles) of the observer can be defined here. In case of load estimation, four eigenvalues are necessary. Otherwise, only three eigenvalues are needed. If a complex pole is specified, then its conjugate will also appear. After giving the eigenvalues (and pressing enter or leaving the edit window), the software calculates the corresponding discrete-time poles and displays them in the *zobs* textbox.

The application designs a controller according to the specifications after clicking on the *Start controller design* button. Parameters of state feedback, reference gains and the observer are saved, displayed at the *Results* window and the file for passing them to *RT_DataAqu_Control* is prepared. (In case of deficient specifications, the program warns the user to correct the specification, or to execute the identification if no identified model is present.) *Results* window displays the designed state feedback K, the state feedback corresponding to the integrator Ki (only in case of integrating control), the reference gains Nx, Nu allowing setpoint control and the matrices F, G, H of the state equation of the discrete-time actual observer.

SIMULATION

This block executes the simulation of the control loop with the controller designed for the identified model. It also compares simulation results with measurements on the controlled physical plant.

Parameter modification

The multiplier of parameter perturbation for the simulation can be defined here. The parameter 1.00 corresponds to the nominal system. The parameters A3, T3, ksi (ξ) and T2 of the identified model are multiplied by this parameter before the simulation.

Disturbance level

Gives the magnitude of the step function disturbance appearing on the input of the plant (load).

Filtering u and y

This is an optional feature which allows the filtering of the input and output signals of the plant during the simulation.

The simulation of the closed loop, comprising the previously designed controller and the (optionally perturbed) plant is carried out upon clicking on the *Start Simulation* button. (The length of the simulation, the shape of the reference signal and the time instance where the input load is switched on are determined automatically). Results are displayed in two windows.

The left sub-window of *Figure No. 4.* shows the output signal, the reference signal and the input load, while the right sub-window shows the control signal (the output of the controller).

Figure No. 5. is divided into four sub-windows. The top left one shows the state $y = x_1$ of the identified model, and its estimation $(\hat{x}_1)$ provided by the state observer. The top right sub-window shows the state x_2 of the identified model, and its estimation $(\hat{x}_2)$ provided by the state observer. The bottom left sub-window shows the state x_3 of the identified model, and its estimation $(\hat{x}_3)$ provided by the state observer. The bottom right sub-window shows the input load (d) and its estimation $(\hat{x}_d)$ provided by the observer (only in case of load estimation).

Upon clicking on the *Load measurements* button, *Figure No. 4.* is extended by the plot of the measured output while *Figure No. 5.* is extended by the plots of the states of the physical plant and the estimated states provided by the observer realized by *RT_DataAqu_Control* (i.e. there will be three plots shown in each sub-window).

The RT_DataAqu_Control application

The *RT_DataAqu_Control* application is running in Windows environment on the Lab PCs, i.e. it is a windows application with MFC (Microsoft Foundation Class) user interface. The program consists of two threads such that the one realizing data acquisition and control has higher priority.

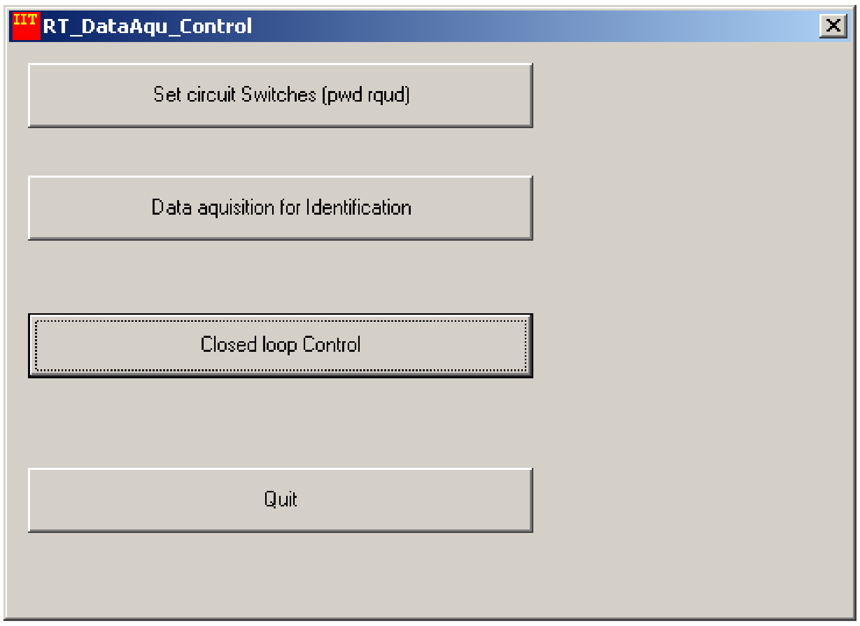

RT_DataAqu_Control
Set circuit Switches (pwd rqud)
Data aquisition for Identification
Closed loop Control
Quit

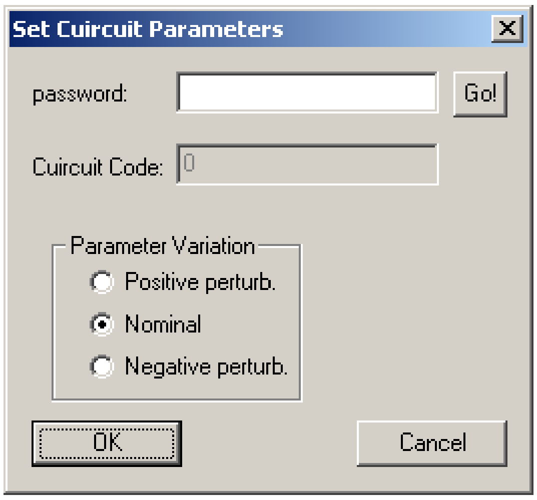
Fig. 8-7. User interface of the RT_DataAqu_Control application

The program starts with a main dialog window (see *Fig. 8-7.*), which allows to display three functional dialogs: Set circuit Switches, Data aquisition for Identification, Closed loop Control. There is also a *Quit* button, which closes the application.

Circuit selection

The Set circuit Switches dialog allows the *teacher* to set which circuit will represent the nominal plant (see *Fig. 8-8*). Moreover, *students* can choose whether they would like to work with the nominal plant, or one with parameters perturbed to negative or positive direction. The selected settings appear in the windows used during data acquisition and closed loop control. It is recommended to check these parameters before starting the aforementioned functions.

If the appropriate password is entered, the Circuit Code window is activated. Then the previously set code can be redefined. The radio buttons in the Parameter Variation frame allow the adjustment of perturbation.


Set Cuircuit Parameters

password: Go!

Cuircuit Code: 0

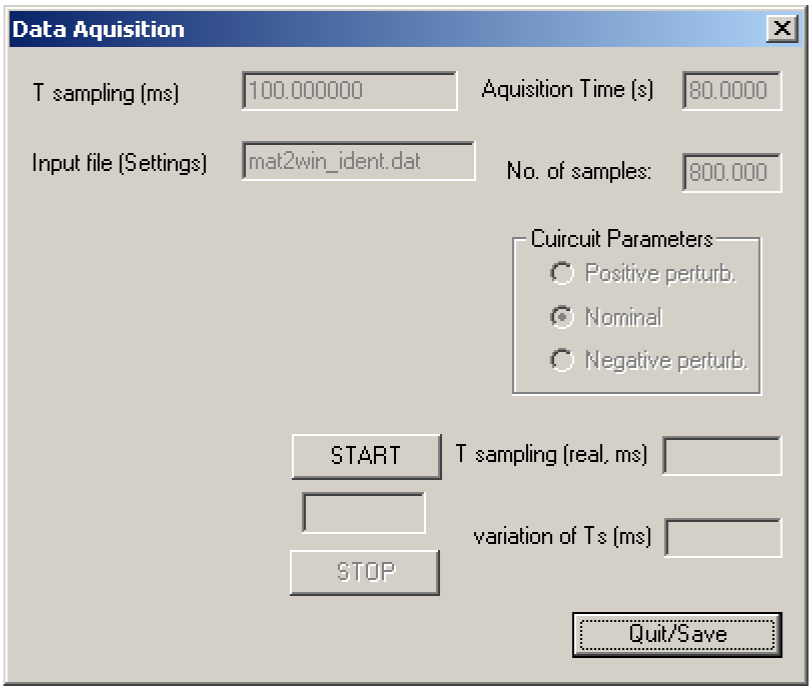
Parameter Variation
- Positive perturb.
- [✓ Nominal]
- Negative perturb.

OK Cancel

Fig. 8-8. Dialog for parameter settings

<u>Data acquisition for identification</u>

The Data acquisition for Identification dialog shows the parameter set in the Matlab application: the suggested sampling time and the length of data acquisition (see *Fig. 8-9*).


Data Aquisition

T sampling (ms): 100.000000

Aquisition Time (s): 80.0000

Input file (Settings): mat2win_ident.dat

No. of samples: 800.000

Cuircuit Parameters
- Positive perturb.
- [✓ Nominal]
- Negative perturb.

START

T sampling (real, ms)

STOP

variation of Ts (ms)

Quit/Save

Fig. 8-9. User interface of the data acquisition

After clicking on the Start button, the program waits for a safety period (approx. 50 seconds), in order to let the system pass to its steady state. Then the data acquisition is started, and the application displays its progress in percentages.

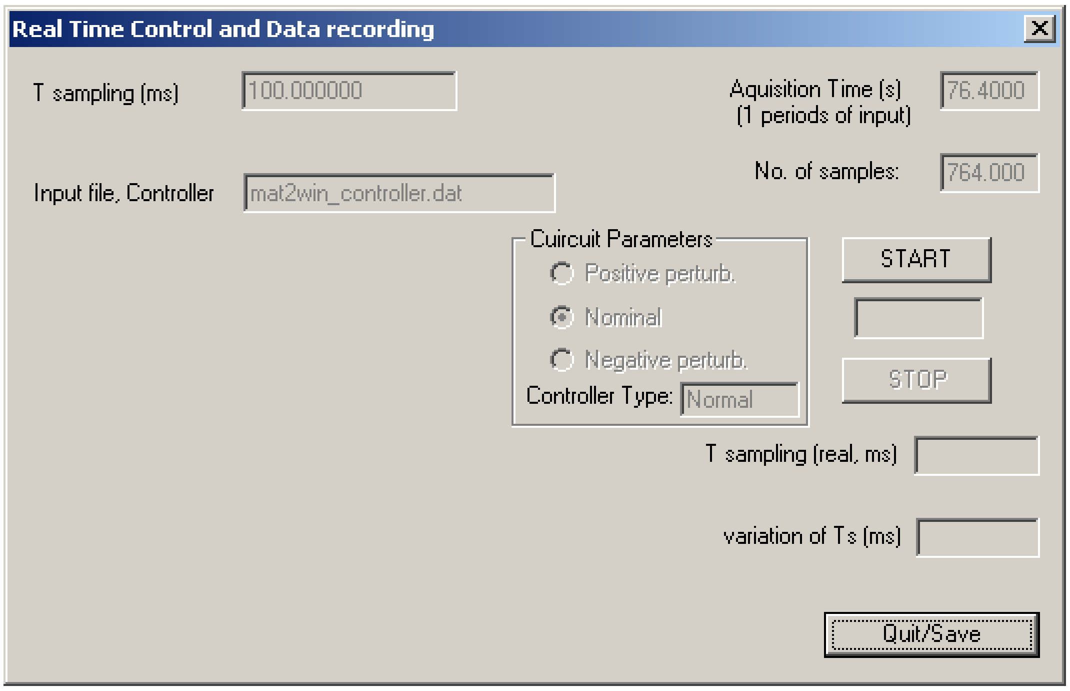
Except the START, STOP and QUIT buttons, all controls of the dialog box are inactive ones, since data displayed by them are either set in the Matlab program or are measurement data.

Due to the properties of Windows scheduler, the prescribed sampling time cannot be exactly respected, but the difference from it is nearly constant at each sampling period and the deviation of the real sampling time is minimal. The program uses the clock pulse of the CPU for fast and precise time measurement. (For CPUs with automatic clock scaling, e.g. Intel Centrino platforms, the automatic scaling feature has to be switched off by the system administrator.)

The application informs the user when the data acquisition is finished. Clicking on the Quit/Save button closes the dialog and saves the measurements to a file. The application uses fixed-name files for communication with the *Consol* Matlab application, which files are re-written upon running the data acquisition. It is practical to save these files or the appropriate Matlab variables and plots to your own directory in order to facilitate preparing the Laboratory report. The data acquisition can be interrupted any time by the Stop button. However, in that case only a truncated file is generated, which is not handled by the *Consol* Matlab application, so interrupting the data acquisition is not suggested.

Control

The functionalities of the Closed loop Control dialog box are similar to the ones of the Data acquisition for Identification dialog, but it displays also the type of the loaded controller (see *Fig. 8-10.*). In the control loop the offset of the output signal is automatically compensated by the term $u_{offset} := -y_{offset} / A_{ident}$ added to the output of the control algorithm.


Real Time Control and Data recording

T sampling (ms): 100.000000

Input file, Controller: mat2win_controller.dat

Aquisition Time (s) (1 periods of input): 76.4000

No. of samples: 764.000

Cuircuit Parameters: Positive perturb. [✓ Nominal] Negative perturb.

Controller Type: Normal

T sampling (real, ms):

variation of Ts (ms):

Fig. 8-10. Dialog of the closed loop control

The latter two dialogs load the files created by Matlab upon clicking on the appropriate button of the main dialog box. In case of corrupted files, the application deactivates all buttons except the Quit/Save one, with which the user can return to the main dialog.

Instruments for the laboratory tasks

The following table summarizes which instruments are to be used to carry out the various laboratory tasks.

Task	Instrument
Measurement of the parameters (speed, damping) of the plant	Signal generator and oscilloscope
Excitation signal generation	Consol Matlab application
Data acquisition	Test circuit and RT_DataAqu_Control application
Identification	Consol Matlab application
Controller design	Consol Matlab application
Simulation	Consol Matlab application
Closed loop control	Test circuit and RT_DataAqu_Control application
Evaluation of closed loop behavior	Consol Matlab application

Table 8-1. Tasks and platforms

Laboratory tasks

1. **Analysis of the plant with external instruments**

   1.1. Select a square wave signal as the input signal of the plant such that the transients of the step response decay during one half-period of the square signal. The amplitude should be high, but care not to saturate the plant!

   1.2. Approximate the DC gain, the undamped natural frequency ω_0 and the damping ξ of the plant including a second order term (dominant pair of poles)

   1.3. Propose an adequate sampling time for data acquisition and control. Recall that the sampling time should satisfy Shannon's theorem for all signals of the *closed loop*. For an initial guess, use e.g. $0.2/\omega_0$.

2. **Identification of the plant**

   2.1. Based on the approximated parameters of the plant, design adequate excitation signals in the Consol Matlab application (refer to the Measurement Instruments section for the meaning of minimal pulse width and change parameters). The minimal sampling time is 20 ms.

   2.2. Carry out the data acquisition using the RT_DataAqu_Control application.

   2.3. Do the parametric identification of the plant using the LS, ARX, IV4 and ARMAX methods.

   2.4. Evaluate the results and select the best model for control design.

3. **Control design**

   3.1. Choose and set the poles of the closed loop and the observer in continuous-time and design a controller with state feedback, state observer (without load estimation) and reference gains.

   3.2. Simulate the closed loop and evaluate the results.

   3.3. Using the RT_DataAqu_Control application, verify the operation of the designed controller on the physical process. Compare the results to the simulated ones.

4. **Test of robustness**

   4.1. Analyze the closed loop behavior of the physical plant with perturbed (positive or negative direction) parameters (do not change the parameters of the controller).

   4.2. Change the controller designed in the previous task by including load estimation to the observer and simulate it along with the perturbed plant.

   4.3. Verify the closed loop behavior of the physical plant using the new controller also on the original and the perturbed plant and evaluate the results.

Useful hints

Recall that the sampling time should be appropriate for each signal of the closed loop by Shannon's theorem. For an initial guess, use e.g. $0.2/\omega_0$.

Apply such a (so-called persistent) signal on the input of the plant during the identification which excites the dynamics of the system. For this purpose, a signal with frequent amplitude-changes is more suitable than a constant one. The identification is carried out in sampled time, and in some special cases it's possible that there exists no continuous-time model corresponding to the identified discrete-time one. In that case, it is worth considering changing the sampling time so that the data acquisition has to be carried out again with the new sampling time.

Ensure that the simulation time interval set in the Matlab application is long enough to represent the whole dynamics of the plant. During this interval, two reference signal changes and one disturbance change is applied to the system, and their compensation may need significant time. We are interested in the whole transient of the controlled loop. However, the identification is carried out at the same time interval, which should contain several amplitude changes.

Recall that significant acceleration of the system may result large control signals which can cause saturation in the controller. On the other hand one of the most important goals of control design is to accelerate the systems as much as possible to have a quick closed loop behavior. Finding a suitable tradeoff between these requirements is a crucial point of control engineering.

When designing the observer, mind that the observation of state variables not available for measurement is possible only with dynamical systems (observers) which are faster than the plant and even the closed loop accelerated by pole placement.

A transient caused by a non-dominant (stable, in closed loop) pole decays before the time of the first peak determined by the dominant pair of poles if its absolute value is approximately three times the absolute value of the real part of the dominant pole.

Review questions

1. Give the system models used for identification!
2. Why identification is necessary?
3. Define the conception of state feedback!
4. Give the characteristic equation of the closed loop in case of state feedback!
5. List the major problems corresponding to simple state feedback $u = -Kx$ in typical control systems!
6. Give the definition of the dominant pair of poles!
7. Give the connection of the damping and undamped natural frequency of the prototype second-order term with the values of its poles!
8. Give the conception of allowing setpoint control in systems with state feedback.
9. Why is it necessary to use a state observer?
10. Give the connection between the nomination “load” and the disturbance signal!
11. Give the concept of disturbance compensation!
12. Define the discrete-time actual observer and specify its advantages!
13. How a continuous-time pole is mapped into the discrete-time domain?