

This formula can be derived by taking the  $\mathcal{Z}$ -transform of (16.50) – (16.51) and then solving for output  $y(z)$ , cf. Section 13.3. However, in practice you will probably use a proper function in e.g. MATLAB or LabVIEW, cf. the examples below.

All the five matrices in (16.50) – (16.51) are estimated. These matrices are assumed having some special canonical forms. Even the initial state is estimated, from known time-series of the input  $u$  and the corresponding output  $y$ . Once a state-space model is estimated, a transfer function may be calculated using (16.52). You must select the model order  $n$  so that you are content with the accuracy of the model. This can be done by running simulations with the model for different orders  $n$ , see Figure 16.8.

Note that if you assume that the system contains a time-delay of  $T_d$  [s], you need at least the following model order to include the time-delay in the model properly:

$$n_{\min} = \frac{T_d}{T_s} \quad (16.53)$$

where  $T_s$  is the sampling time.

In the following are two examples about estimating a black-box model of an electrical motor using subspace estimation, with MATLAB (System Identification Toolbox) and LabVIEW (System Identification Toolkit), respectively.

### **Example 16.3** *Subspace model estimation in MATLAB*

Figure 16.10 shows an electrical DC motor. It is manipulated with an input voltage signal,  $u$ , and the rotational speed is measured with a tachometer which produces a output voltage signal,  $y$ , which is proportional to the speed. During one experiment lasting for about 17 seconds,  $u$  was adjusted manually, and both the sequences (time series) of  $u(t_k)$  and  $y(t_k)$  were saved to a file. The sampling time was 0.02 s. Figure 16.11 shows a small extract of the log file as displayed with Notepad.<sup>5</sup> The first column contains time stamps, but they are not used in this application. The second column contains the input sequence  $u(t_k)$ , and the third column contains the output sequence,  $y(t_k)$ .

Actually, the input and output sequences were divided into two parts:

---

<sup>5</sup>The whole logfile, named logfile1.lvm, is available from the home page of this book at <http://techteach.no>.

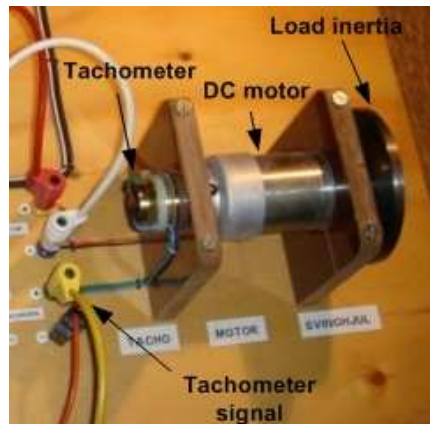


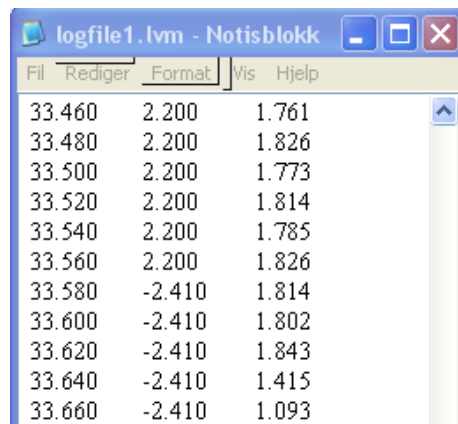
Figure 16.10: Example 16.3: Electrical (DC) motor for which an  $s$ -transfer function is estimated

- The first half, named  $u_{estim}$  and  $y_{estim}$ , were used to estimate a transfer function.
- The second half, named  $u_{valid}$  and  $y_{valid}$ , were used to check that the model is a good model. This is done by simulating the model with  $u_{valid}$  as input signal and comparing the simulated response,  $y_{sim}$ , with  $y_{valid}$ . If  $y_{valid}$  is quite similar to  $y_{sim}$  we can conclude that the model is good.

The estimation is made with the **n4sid**-function in MATLAB's System Identification Toolbox.

Below is the MATLAB-code which accomplishes the task.

```
%Loads data from file into workspace.
load logfile1.lvm;
Ts=0.02; %Sampling interval
L=length(logfile1);%(Matrix name becomes same as logfile name.)
N=round(L/2);
%Generates proper time signal, and extracts data from logfile:
t_estim=Ts*[1:N]';
u_estim=logfile1(1:N,2);
y_estim=logfile1(1:N,3);
t_valid=Ts*[N+1:L]';
u_valid=logfile1(N+1:L,2);
y_valid=logfile1(N+1:L,3);
modelorder=1;%Defines order of estimated model.
```



Time	Input	Output
33.460	2.200	1.761
33.480	2.200	1.826
33.500	2.200	1.773
33.520	2.200	1.814
33.540	2.200	1.785
33.560	2.200	1.826
33.580	-2.410	1.814
33.600	-2.410	1.802
33.620	-2.410	1.843
33.640	-2.410	1.415
33.660	-2.410	1.093

Figure 16.11: Example 16.3: An extract of the log file.

```

%Estimation of model. Model is on internal theta-format:
model_est=n4sid([y_estim u_estim],modelorder);
%th2tf-function calculates numerator and denominator coeff. arrays
%in z-transfer function:
[num,den]=th2tf(model_est);
H_disc=tf(num,den,Ts); %Generates an LTI-model from z-transf func.
y_sim=lsim(H_disc,u_valid,t_valid);%Simulates with u_valid as input.
figure(1)
%Plots y_estim and u_estim:
plot(t_estim,[y_estim,u_estim]);
ylabel('V');xlabel('t [s]')
figure(2)
%Plots y_sim, y_valid, and u_valid.
plot(t_valid,[y_sim,y_valid,u_valid]);
ylabel('V');xlabel('t [s]')
H_cont=d2c(H_disc,'zoh') %Converts to s-transfer function.

```

Figure 16.12 shows the input  $u_{estim}$  and output  $y_{estim}$  used for the estimation.

I selected order  $n = 1$  for the model because it seemed not to be any large improvement in using a higher order, and according to the parsimony principle of system identification, you should select the simplest model among proper models. Figure 16.13 shows the input  $u_{valid}$  and output  $y_{valid}$  used for validating the model, together with the simulated output  $y_{sim}$ . Since  $y_{valid}$  is quite similar to  $y_{sim}$  we can conclude that the model is good.

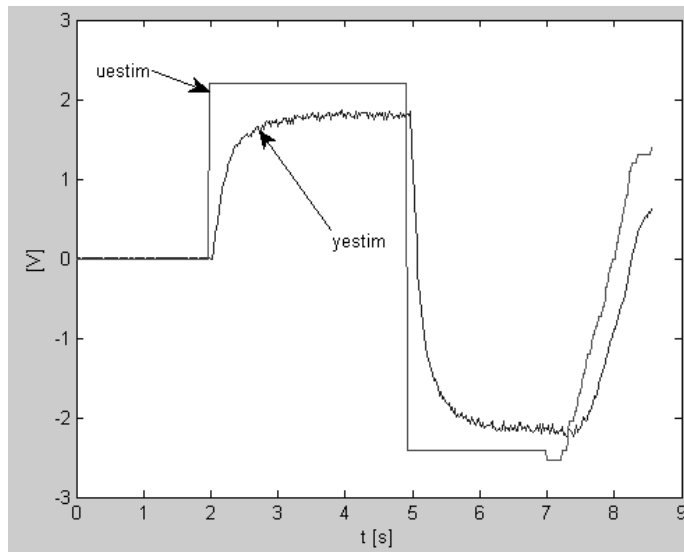


Figure 16.12: Example 16.3: The input  $u_{estim}$  and output  $y_{estim}$  used for the estimation.

The resulting discrete-time transfer function model was

$$H_{disc}(z) = \frac{0.05788}{z - 0.9344} \quad (16.54)$$

This model was converted to the following continuous-time transfer function using the **d2c** function:

$$H_{cont}(s) = \frac{2.993}{s + 3.393} \quad (16.55)$$

$$= \frac{2.993/3.393}{(1/3.393)s + 1} = \frac{0.88}{0.29s + 1} = \frac{K}{Ts + 1} \quad (16.56)$$

Thus, the gain is 0.88, and the time-constant is 0.29 sec. I know from experience with the motor that these are good values!

[End of Example 16.3]

The next example shows how I used LabVIEW to accomplish the same system identification task as in Example 16.3.

#### **Example 16.4** *Subspace model estimation in LabVIEW*

The introduction to the example is the same as in is Example 16.3.

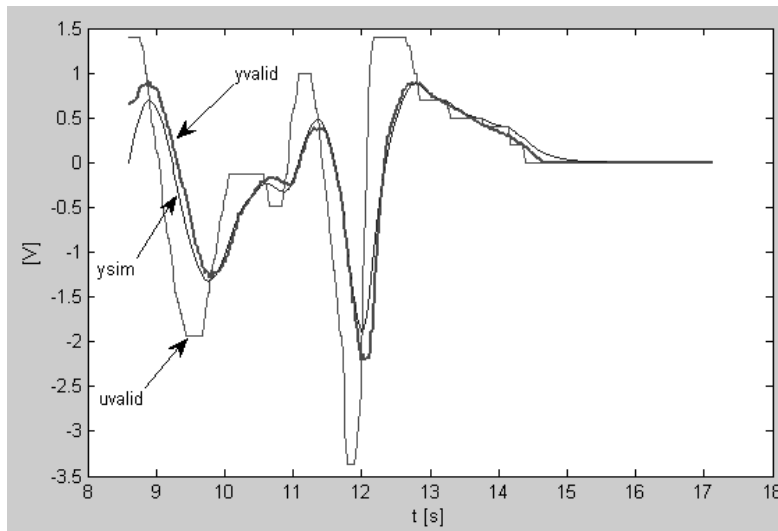


Figure 16.13: Example 16.3: The input  $u_{valid}$  and output  $y_{valid}$  used for validating the model, together with the simulated output  $y_{sim}$ .

Figure 16.14 shows the front panel (the user interface) of the LabVIEW program (sysid.vi), Figure 16.15 shows the block diagram (programming code) of the program.

The result of the estimation is the continuous-time transfer function

$$H_{cont}(s) = \frac{y(s)}{u(s)} = \frac{0.0031s + 3.08}{s + 3.54} \approx \frac{3.08}{s + 3.54} \quad (16.57)$$

$$= \frac{3.08/3.54}{(1/3.54)s + 1} = \frac{0.87}{0.28s + 1} = \frac{K}{Ts + 1} \quad (16.58)$$

with no time-delay. Thus, the gain is 0.87, and the time-constant is 0.28 sec, which are very similar to the values found with MATLAB in Example 16.3 (and these parameter values are good).

Below are a number of comments to the front panel and to the block diagram of the LabVIEW program:

Comments to the *front panel*, see Figure 16.14:

1. The program loop time of 1 sec is the loop or cycle time of the program. This means that the code implementing the estimation etc. is executed each second. This continuous program execution makes it possible for the user to see the consequences of adjusted settings “immediately”.

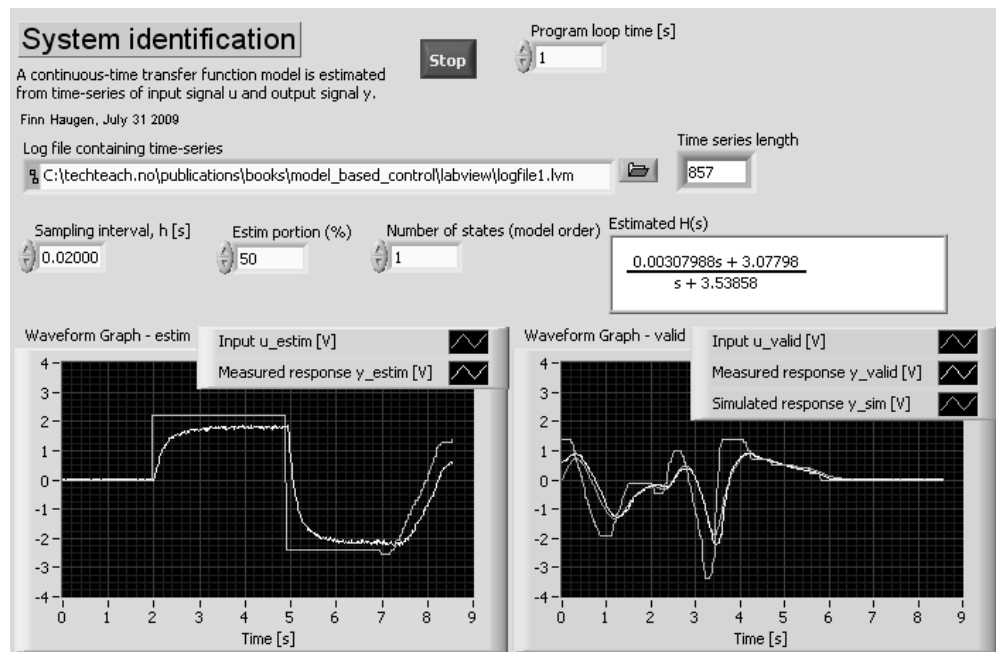


Figure 16.14: Example 16.4: The front panel of the LabVIEW program sysid.vi.

2. The user can select the model order.
3. The user can select the percentage portion of the original time-series to be used for estimation. In this example I have selected to use the first 50% for estimation, and the second 50% for validation via simulation.
4. The front panel shows in respective charts the data used for estimation (left) and the data used for validation (right). From the chart to the right we see that the model is good, as  $y_{sim}$  and  $y_{valid}$  are quite similar.

Comments to the *block diagram*, see Figure 16.15:

1. The **Read From Measurement File** function reads the saved data (sequences) from the log file and makes them available to the LabVIEW program. The 2-dimensional arrays of data in the file are extracted to proper 1-dimensional arrays using **Index Array** functions.
2. The function **SI Split Signal** splits the signals to be used for

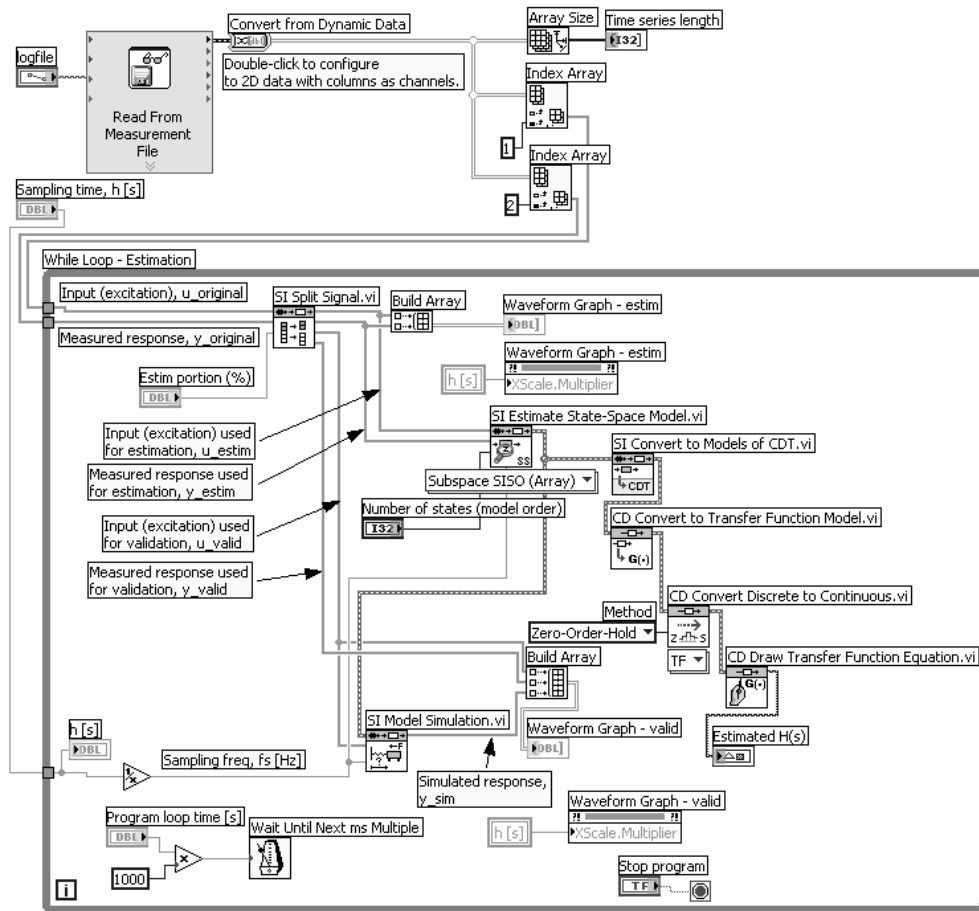


Figure 16.15: Example 16.4: The block diagram of the LabVIEW program `sysid.vi`.

estimation and validation (simulation) according to the **Estim Portion (%)** value.

3. The function **SI Estimate State-Space Model**<sup>6</sup> estimates a discrete-time state-space model using a subspace estimation method.
4. The estimated model is eventually converted to a continuous-time transfer function,  $H_{cont}(s)$ , using functions in the Control and Simulation toolkit (at the right part of the block diagram). The result is (16.57).
5. The function **SI Model Simulation** simulates the estimated model using  $u_{valid}$  as input signal. The simulated response,  $y_{sim}$ , is shown

<sup>6</sup>SI = System Identification

together with *yvalid* in the chart at the right side of the front panel.

[End of Example 16.4]