Version 5 of the System Identification Toolbox
for use with MATLAB – With Object Orientation

Lennart Ljung

Department of Electrical Engineering
Linkoping University, S-581 83 Linkping, Sweden
WWW: http://www.control.isy.liu.se
Email: ljung@isy.liu.se

2000-03-07

Report no.: LiTH-ISY-R-2221
For the IFAC Symposium on System Identification, SYSID2000,
Santa Barbara, CA, July 2000

Technical reports from the Automatic Control group in Linkping are available
by anonymous ftp at the address ftp.control.isy.liu.se. This report is
contained in the pdf file 2221.pdf.
VERSION 5 OF THE SYSTEM IDENTIFICATION TOOLBOX FOR USE WITH MATLAB - WITH OBJECT ORIENTATION

Lennart Ljung *

* Division of Automatic Control, Linköping University, SE-58183, Linköping, Sweden, email: ljung@isy.liu.se

Abstract: Version 5 of the System Identification Toolbox is entirely rewritten, making use of MATLAB 5's objects. While the old syntax is still honored, the object orientation gives a substantial improvement of user interaction with model properties and algorithm options. In addition to the new objects, version 5 has a number of new features: handling of multiple data sets, free state-space parameterizations, estimating initial conditions for input-output models, etc.

1. INTRODUCTION

Version 5 of the System Identification Toolbox, (Ljung 2000), is a substantially rewritten version. The main change is that the code is object oriented, and a number of objects to deal with models and data have been introduced.

The following motivations for the rewrite can be singled out:

1. To allow for an entirely transparent relationship with control systems toolbox, and other Matlab toolboxes that deal with linear time invariant systems.
2. To allow for easier user interaction with the central model properties.
3. To be able to handle a larger number of options (related to algorithms and other aspects) in an easy and transparent manner.
4. To have a transparent way of handling multivariable data and systems: Dealing with subsets of input and output channels, and letting the toolbox take care of the bookkeeping.

In addition to the object orientation, version 5 of the Identification Toolbox also contains the following new features:

1. Introduction of free state-space parameterizations that can be used for maximum likelihood/prediction error estimation without any canonical forms. This allows excellent numerical properties, and links very well to state-space models obtained by subspace algorithms.
2. A updated version of the subspace algorithm, now equipped with a number of options that the user can control.
3. Several options to deal with initial conditions for the filters involved in the estimation. This is of particular importance when dealing with output error models, with slow time constants. Dramatic improvements in the model quality can be achieved by proper handling of the initial conditions.
4. The possibility to handle "estimation focus" in a simple manner. Each of the estimation algorithms can be asked to focus on certain frequency ranges in fitting the model, without destroying the noise model.
5. A number of new options to interact, and control the iterative search have been introduced.

2. THE OBJECTS

2.1 IDDATA

The IDDATA object contains the actual estimation data, as well as considerable amount of
optional book keeping facilities, such as signal names, signal units, etc. Information about the input, such as periodicity and inter sample behavior is also stored, so that this information can be used in the correct fashion by the estimation algorithms.

The IDDATA object also allows to store several different experiments in one object, to handle not equally sampled data and a fair amount of logistics to select subsets of the data (both in terms of samples, channels, and experiments) as well as to obtain merged data sets from various different channels and samples.

2.2 IDMODEL

The different models that can be produced by the System Identification Toolbox, such as polynomial models, state space models, and structured grey-box models are associated with different objects, so that structure properties easily can be set, adjusted and read.

The model objects are also closely related to the LTI-models of the Control Systems Toolbox. The IDMODEL objects and the LTI objects can be easily transformed into each other. This also allows the same type of syntax in the System Identification Toolbox as in the Control Systems Toolbox to analyze and display properties of linear time invariant models.

3. DEALING WITH THE OBJECTS

Here is a sequence, where we define a model, simulate it, and then use the data to estimate and evaluate a model.

```
>> m0 = idpoly([1 -1.5 0.7],...  
[0 1 0.5;1 2 3],[1 -1 0.2]);
>> m0
```

Discrete-time IDPOLY model:
\[
A(q)y(t) = B(q)u(t) + C(q)e(t)
\]
\[
A(q) = 1 - 1.5 q^{-1} + 0.7 q^{-2}
\]
\[
B1(q) = q^{-1} + 0.5 q^{-2}
\]
\[
B2(q) = 1 + 2 q^{-1} + 3 q^{-2}
\]
\[
C(q) = 1 - q^{-1} + 0.2 q^{-2}
\]
This model was not estimated from data.
Sampling interval: 1

```
>> m0.noisevar = 0.1;
>> u = idinput([400 2])
```

Data set with 400 samples.
Sampling interval: 1
Inputs  Unit (if specified)
   u1
   u2

```  
>> y = idsim([u,randn(400,1)],m0)
```

Data set with 400 samples.
Sampling interval: 1
Outputs  Unit (if specified)
y1

```
>> dat = [y u]
```

Data set with 400 samples.
Sampling interval: 1
Outputs  Unit (if specified)
y1
Inputs  Unit (if specified)
u1
   u2

```  
>> m = pem(dat);
>> size(m)
```

State space model with 1 output, 2 inputs, 2 states, and 14 free parameters.
```  
>> m.a
```

```
ans =
    0.8215    0.5949
   -0.2408    0.6763
```

```
>> m.b
```

```
ans =
   3.9395   14.7738
   1.9266    7.7128
```

```
>> m.ssp = 'canonical';
```

```
>> m.a
```

```
ans =
   0   1.0000
   0  1.0989
```

```
>> nyquist(m,10)
```

```
>> step(m,dat)
```
>> m. EstimationInfo

ans =

    Status: 'Estimated model (PEM)'
    Method: 'PEM'
    LossFcn: 0.0876
    FPE: 0.0921
    DataName: 'dat'
    DataTs: 1
    DataInterSample: {2x1 cell}
    WhyStop: 'No improvement ... along search direction.'
    UpdateNorm: 0.0229
    LastImprovement: 0
    N4Horizon: [15 5 5]
    N4Weight: 'MOESP'

>>

Fig. 1. The nyquist curve for the model with uncertainty regions corresponding to 10 standard deviations.

4. THE 'FOCUS' FACILITY

An algorithm property 'Focus' has been introduced, that can take the values 'Prediction', 'Simulation', or any SISO filter. With 'Prediction', a standard prediction error method is obtained. 'Simulation' means that the transfer function from input to output is computed as an Output Error model, while the noise model is estimated by a prediction error criterion, fixing the dynamics to the earlier estimated transfer function.

>> load iddata1
>> m1 = arx(z1(1:150), [2 2 1], 'Foc', 'Pre');
>> m2 = arx(z1(1:150), [2 2 1], 'Foc', 'Sim');
>> compare(z1(151:300), m1, m2)

5. DEALING WITH MIMO MODELS

The input and output channels can be selected in both iddata and idmodel objects by simple sub-referencing by channel number or name, like in data(:, [1 2], 5) or model('temp', 'current', 'voltage'). The channel names are used for bookkeeping so that I/O channels with the same names will be plotted together, and matched for simulation and prediction. In that way, submodels can be constructed, estimated, and analyzed in a transparent manner. This is best illustrated by an example. Here the two SISO data sets are loaded, concatenated to a MIMO data set. A default MIMO model (m) is estimated using these data. Then a SISO model m1 is extracted from m by just considering the transfer function from input 1 to output 1. In addition, another SISO model, m2 is constructed.

Fig. 2. The step responses of the model m and a model directly estimated from data.

Fig. 3. The simulated responses of model m1 (focus = prediction) and m2 (focus = simulation). m1 has a fit of 53% while m2 has a fit of 70%. The fit is the percentage of the output variation that is reproduced by the model.
from the data, by estimating it from the data, using just input channel 2 and output channel 2. The three models are then compared on a MIMO validation data set:

```matlab
>> load iddata1
>> load iddata2
>> zz = [z1,z2(1:300)]
```

Data set with 300 samples.
Sampling interval: 0.1
Outputs Unit (if specified)
y1
y2
Inputs Unit (if specified)
u1
u2
```matlab
>> m = pem(zz(1:200));
>> m2 = pem(zz(1:200,2,2));
>> compare(zz(201:300),m,m22,ms)
>> figure
>> ms = m(1,1);
>> compare(zz(201:300),m,m22,ms)
```

Fig. 4. Comparisons between the MIMO models and SISO submodels. The upper plot shows how models m and ms reproduce the output y1 (the model m22 does not deal with this output. The lower plot shows how models m and m22 reproduce the second output.

6. INITIAL STATES

To compute the model predictions, initial values of predictors and signals are required. These are not known and typically taken as zero. For systems with strong transients, this may lead to very bad models. To deal with this an algorithm property 'InitialState' has been introduced. This may take the following values:

- 'Zero': Zero initial state
- 'Estimate': Estimate the initial state as unknown parameters
- 'Backcast': Use Knudsen's, (Knudsen 1994), technique to filter backwards to find estimated of the initial state
- 'Auto': An automatic, and data dependent choice among the above.

To illustrate the use and importance of this feature, simulate a model with poles very close to the unit circle. Estimate it with zero initial state and with backcast. Compare the model step responses with that of the true system:
```matlab
>> m0 = idpoly([0 1 2 3],1,1,...
               [1 -1.5 0.999]);
>> u=idinput(400);
>> y = idsim([u randn(400,1)],m0);
>> z=[y u];
>> m1=oe(z(201:400),[3 2 1],'init','zer');
>> m1.f
ans =
    1.0000   -1.5472    1.0000
>> m2=oe(z(201:400),[3 2 1],'init','back');
>> m2.f
ans =
    1.0000   -1.5001    0.9987
>> step(m0,m1,m2)
```

Fig. 5. Step responses from true system (m0), the 'Backcast' model m2 (these essentially coincide) and the 'Zero' initial condition model m1 (the lowest curve).

7. FREE STATE SPACE PARAMETERIZATIONS

The toolbox also implements the free state-space parameterization, described in (McKelvey and
8. MULTI-EXPERIMENT DATA

An IDDATA object can hold an arbitrary number of separate experiments. All estimation and validation routines accept such multi-experiment data sets. This is quite useful in two common cases:

- When data have been collected on different occasions, and we want to use all of them to build models.
- When data contain “bad” portions with large disturbances, or no information. Then it is of interest to cut out the informative pieces and use all of them for model estimation.

This is easy to handle:

```matlab
>> m = pem(zl);
>> m.ss
ans =
    Free
>> m.a
ans =
    0.8146  -0.5524
         0.2636   0.7158
>> m.ss = 'can';
>> m.ss
ans =
    Canonical
>> m.a
ans =
    0.0  1.0000
         -0.7287  1.5304
```

9. MODEL ERROR MODELS

So called model error models, (Ljung 1999), are simply models from the measured input to the residuals from another model. Building explicit model error models may be a useful alternative to conventional residual analysis. In the toolbox, the command resid, when called with an output argument, generates an IDDATA object, from which model error models can be directly estimated:

```matlab
>> m = pem(Data);
>> e = resid(Datv,m);
>> mem = spa(e);
>> mem2 = bjl(e,[0 10 2 2 0]);
>> bode(mem,mem2,'z1','3','fill')
```

An analysis of this kind, for model error models of default structure, is also offered by resid itself:

```matlab
>> resid(Datv,m,'fr') % frequency response
>> resid(Datv,m,'ir') % Impulse response
```

This shows the frequency response of the model error model (Figure 6) with uncertainty region. The model that has a significant error around 0.5 rad/s. The impulse response of the error model is shown in Figure 7. The default use of resid is still with traditional correlation analysis, but the above options typically are more informative for control applications.

10. SUMMARY

The updated version of the toolbox offers new facilities and new, more efficient ways of interaction. The old THETA format has been replaced by new model objects. However, 100% downward compatibility is maintained, the old syntax still works, and previously saved models and GUI-session can still be loaded (and automatically converted to the new objects.)
Fig. 6. The amplitude frequency response of the model error model. The filled area is the region of no significant response.

Fig. 7. The impulse response of the model error model. The filled area is the region of no significant response.

11. REFERENCES


