I1-IDENTIFICATION TOOLBOX FOR MATLAB

Tore K. Gustafsson
Åbo Akademi, Process Control Laboratory
E-mail: tore.gustafsson@abo.fi

Pertti M. Mäkilä
Luleå University of Technology, Division of Automatic Control
E-mail: Pertti.Makila@sm.luth.se

Version 1.1
August 1994

This work was partly supported by the Academy of Finland
under grants FA 1081195 and FA 4522

1 MATLAB is a trademark of The MathWorks, Inc.
Table of Contents

I Tutorial
1. INTRODUCTION ................................................................. 5
2. INSTALLATION ................................................................. 5
3. $l_1$-IDENTIFICATION ...................................................... 6
4. MODEL STRUCTURES FOR NOMINAL MODELS .................... 8
   4.1 Fixed-Pole Models ....................................................... 8
   4.2 Laguerre models ......................................................... 8
   4.3 Kautz models ............................................................. 9
   4.4 Combined models ....................................................... 10
   4.5 Model validation ......................................................... 11
5. ESTIMATING UNCERTAINTY MODELS .............................. 11
6. REFERENCES ................................................................. 13

II Reference
ktz2fir ...................................................................................... 15
l1arx ....................................................................................... 16
l1fir ......................................................................................... 18
l1firgr ..................................................................................... 19
l1firud ..................................................................................... 21
l1fparx ..................................................................................... 23
l1fpoes .................................................................................... 24
l1fpoeu ..................................................................................... 25
l1fpud ....................................................................................... 26
l1fpunc ..................................................................................... 27
l1kautz ...................................................................................... 28
l1klt .......................................................................................... 30
lgr2fir ....................................................................................... 32
lpsimplx ..................................................................................... 33
prbs ......................................................................................... 34
rndlapl ...................................................................................... 35
I TUTORIAL

1. INTRODUCTION

The $l_1$-identification toolbox for Matlab (version 4) is a set of functions for approximate identification of discrete dynamic models and estimation of uncertainty in the identified nominal models. Most functions minimise the LSAD-criterion (Least Sum of Absolute Deviations) using the Simplex algorithm for Linear Programming. The formulation of the identification problem as a linear programming problem allows inclusion of several types of priors, and practical estimation of uncertainty.

The toolbox is available through anonymous ftp at ftp.abo.fi in directory /pub/rt/l1idtools. This toolbox must not be used for commercial purpose. The authors take no responsibility for the correctness of the programs, nor for any inconvenience or damage this toolbox can cause to the user. The programs may contain untested features.

Many routines are based on two fortran programs for linear programming, see Barrodale and Roberts (1983) and Press et al. (1986). The first one, cl1app, is based on a specialised algorithm for $l_1$-approximation. The second one, simplx used in lpsimplx, is a common simplex algorithm for linear programming. It should be noted that the function lp in the Optimization Toolbox for Matlab (Grace, 1990) is not a simplex algorithm and it is not suitable for solving large linear programs.

Version 1.1 contains a MATLAB-language linear programming function, wlpsmplx, which can be used instead of the mex-function lpsimplx. The function wlpsmplx is based on functions wsimplex and wsimpreg, written from the interactive LP-program by Henry Wolkowicz, and found at ftp.mathworks.com in directory /pub/contrib/optim/simplex1. The following gives a rule of thumb for the performance of different linear programming functions. In an example with l1fpud with 200 variables, 1 inequality constraint and 92 equality constraints l1fpud using lpsimplx needed 48 s, l1fpud with lpsmplx needed 226 s, and l1fpud with lp (from Optimization Toolbox) needed 1293 s CPU-time on a Sun-4.

2. INSTALLATION

Many of the routines are based on two fortran programs for linear programming. The fortran code is supplied in f-files as well as in compiled and linked mex4-files. The mex4-files can only be used on a Sun-4 computer. The f-files must be separately compiled and linked for other computers. The procedure is

```
fmex cl1app.f cl1appg.f
fmex simplx.f simplxg.f
fmex l1arx.f cl1app.f.
```
3. \textit{l}_1-\textit{IDENTIFICATION}

This tutorial gives a brief introduction to the ideas and nomenclature of \textit{l}_1-identification. A more thorough introduction is given by Mäkilä et al. (1994). The theory behind the functions of the \textit{l}_1 toolbox is discussed by Gustafsson and Mäkilä (1993a, 1993b, 1993c). The latter references contain also some application examples.

System modelling for robust \textit{l}_1-control (Dahleh and Khammash, 1993) involves approximate modelling of the time-domain input/output behaviour of the system. This theory uses the \textit{l}_1-norm as a measure of uncertainty. The functions of this toolbox are designed to perform these tasks. Another motivation for choosing \textit{l}_1-identification instead of the more common \textit{l}_2-identification (using least squares), is that \textit{l}_1-identification uses linear programming, which is a convenient tool for solving large approximation problems including linear constraints. This facilitates the use of prior knowledge about the system. Worst case analysis of identification in \textit{l}_1 has recently been treated by several authors, e.g. Mäkilä (1991), Jacobsen et al. (1992), Lin et al. (1992), Milanese and Vicino (1993), Tse et al. (1993) and Milanese (1994).

We consider input-output data sequences, \(\{u(t), y(t)\}_{t=0}^{t=N-1}\), from an identification experiment on an unknown system \(y(t) = (G*u)(t) + v(t)\), where \(v(t)\) is a bounded experimental error term (noise/disturbance) and \(N\) denotes the number of input/output data pairs available.

We want to adjust a model with a chosen structure to the input-output sequence at hand, such that the model output error is minimised. We choose a discrete linear time-invariant model

\[
y(t) = (\hat{G}*u)(t) + \xi(t) = \sum_{k \geq 1} g(k)u(t-k) + \xi(t)
\]

where the total output error contains both the noise \(v(t)\) and unmodelled dynamics. We can also write the model in the form,

\[
\hat{G}(z) = \sum_{k \geq 1} g(k) z^k
\]

The coefficients \(g(k)\) equal the unit impulse response of the model \(\hat{G}\). When \(\hat{G}\) is a rational function of \(z\) the coefficients \(g(k)\) can be calculated as the impulse response of the model. Alternatively the coefficients \(g(k)\) can be obtained through a Laurent expansion of \(\hat{G}(z)\). Calculation of \(g(k)\)-coefficients is performed in functions \texttt{lgr2fir} and \texttt{ktz2fir} for the special Laguerre- and Kautz-model structures.

We define the \textit{l}_1 norm of the model \(\hat{G}\) as \(\|\hat{G}\| = \sum_{k \geq 1} |g(k)|\). In order to estimate \(\hat{G}\) we minimise the total output error in the \textit{l}_1-norm. The identification criterion is the least sum of absolute deviations (LSAD) criterion,

\[
\min \frac{1}{N} \sum_{t=0}^{N-1} |\xi(t)|
\]
For some model structures it is possible to penalize the complexity of the behaviour of the model by including a suitable weighted $l_1$-norm of the derivative of the model in the criterion. In the toolbox the functions for FIR-model identification can use the following criterion:

$$
\min \left[ \frac{1}{N} \sum_{t=0}^{N-1} |\xi(t)| + \lambda \sum_{k \geq 1} |g(k)| \right] 
$$

(3.4)

The model $\hat{G}$, which is identified using criterion (3.3) or (3.4), is called a nominal model. A nominal model is simply the best fit to the data, without any estimates of the size or structure of the uncertainty, which are treated later.

**Model and error priors.**

The $l_1$-identification functions allow the use of priors in the form of constraints. Eqn (3.3) or (3.4) is solved by linear programming, and consequently all priors which can be described as linear constraints on the adaptable parameters can be applied.

The parameter values can directly be constrained with inequality or equality constraints,

$$
E_p \leq f 
$$

(3.5)

$$
C_p = d 
$$

(3.6)

where $p$ is the parameter vector.

Smoothness priors in the form of constraints on the derivative of the transfer function of the identified model,

$$
\|\hat{G}\|_1 = \sum_{k \geq 1} |g(k)| \leq \gamma_1 
$$

(3.7)

can be applied exactly in FIR-model identification and approximately in Laguerre- and Kautz-model identification.

Constraints on the $l_1$-gain of the identified FIR-model can be applied as

$$
\|\hat{G}\|_2 = \sum_{k \geq 1} |g(k)| \leq \gamma_2 
$$

(3.8)

$v(t)$ is usually in robust identification considered as a bounded noise sequence, e.g. $\max_{0 \leq t \leq N-1} |v(t)| \leq \varepsilon$. In the toolbox error priors are included as minimum and maximum values,

$$
\min_{0 \leq t \leq N-1} \xi(t) \geq \varepsilon_- \quad \max_{0 \leq t \leq N-1} \xi(t) \leq \varepsilon_+ 
$$

(3.9)
4. MODEL STRUCTURES FOR NOMINAL MODELS

A general model structure is

\[ A(q) y(t) = \frac{B(q)}{F(q)} u(t-n_k) + \frac{C(q)}{D(q)} \xi(t) \]  

(4.1)

with the nomenclature of Ljung (1991). \( A, B, C, D \) and \( F \) are polynomials in the backward shift operator \( q^{-1} \), with dimensions \( n_a, n_b, n_c, n_d \) and \( n_f \), respectively.

\[ A(q) = 1 + a_1 q^{-1} + a_2 q^{-2} + \cdots + a_{n_a} q^{-n_a} \]  

(4.2)

\[ B(q) = b_1 + b_2 q^{-1} + b_3 q^{-2} + \cdots + a_{n_b} q^{-n_b+1} \]  

(4.3)

\[ F(q) = 1 + f_1 q^{-1} + f_2 q^{-2} + \cdots + f_{n_f} q^{-n_f} \]  

(4.4)

\( C \) and \( D \) are defined analogously, but they are not used in this toolbox. In the sequel we will use \( C(q) = 1 \) and \( D(q) = 1 \). We distinguish the following forms of (4.1).

The ARX structure has \( F(q) = 1 \), and is thus given by

\[ A(q) y(t) = B(q) u(t-n_k) + \xi(t) \]  

(4.5)

The FIR structure has additionally \( A(q) = 1 \), and is thus given by

\[ y(t) = B(q) u(t-n_k) + \xi(t) \]  

(4.6)

The Output Error structure is given by

\[ y(t) = \frac{B(q)}{F(q)} u(t-n_k) + \xi(t) \]  

(4.7)

\( l_1 \)-identification of ARX models is performed in function \texttt{l1arx} and of FIR models in function \texttt{l1fir}.

4.1 Fixed-Pole Models

Identification using a general model structure (4.1) cannot be guaranteed to result in a stable model even if the system is stable. Therefore we consider using fixed-pole models, which are guaranteed to produce stable models. The simplest fixed-pole model structure is the FIR model (4.6). FIR models, however, must often be of a large order \( n_b \) to approximate the system to a reasonable degree, and contain thus a large number of parameters. A remedy might be to use the special Laguerre- or Kautz-model structures. A more general procedure is to simply use an ARX or output-error structure with a fixed polynomial \( A(q) \) or \( F(q) \), respectively. This procedure is used in the functions \texttt{l1fparx} and \texttt{l1fpoe}, respectively.

4.2 Laguerre models

The Laguerre model structure we use is a FIR model in the Laguerre domain,
with the definition

\[ B_L(q, \alpha) = b_1 L_1(q, \alpha) + b_2 L_2(q, \alpha) + \ldots + b_{n_L} L_{n_L}(q, \alpha) \]  

(4.2.2)

where \( L_k \) is the discrete Laguerre polynomial of order \( k \),

\[ L_k(q, \alpha) = \frac{1}{\sqrt{1 - \alpha^2}} \left( \frac{q^{-1} - \alpha}{1 - \alpha q^{-1}} \right)^{k-1} \]  

(4.2.3)

The Laguerre parameter \( \alpha, |\alpha| \leq 1 \), is selected based on prior knowledge about the system, or it can be adjusted during the identification procedure. With \( \alpha \) selected, the model (4.2.1) is a fixed-pole ARX model with multiple poles at \( \alpha \). The Laguerre parameter can be seen as a time-scaling parameter, which is used in order to reduce the number of parameters compared to a FIR model. Laguerre-model identification is performed in function `l1firlg`.

During identification of Laguerre models, the model (4.2.1) is transferred into state space form

\[ x(t + 1) = Fx(t) + Gu(t) \]

\[ y(t) = Bx(t) + \xi(t) \]  

(4.2.4)

where \( B \) is a row vector of Laguerre coefficients. The state space model (4.2.4) is in the toolbox function initialised with \( x(0) = 0 \), and simulated for a number of time intervals in order to let the state vector converge. A part of the input/output data pairs is thus used for initialisation and cannot be used in the LSAD minimisation. The number of input/output data pairs which must be used for initialisation depends on the selected value \( \alpha \).

### 4.3 Kautz models

The Kautz model structure is a modified form of the Laguerre structure with a complex parameter (Wahlberg, 1991)

\[ y(t) = B_K(q, \alpha) u(t) + \xi(t) \]  

(4.3.1)

with the definition

\[ B_K(q, \alpha) = b_1 \Psi_1(q, \alpha) + b_2 \Psi_2(q, \alpha) + \ldots + b_{n_K} \Psi_{n_K}(q, \alpha) \]  

(4.3.2)

where \( \Psi_k \) is the \( k \):th base function,
\[
\Psi_{2j-1}(q, \alpha) = C_1 \frac{-aq^{-1} + q^{-2}}{1 + b(c-1)q^{-1} - cq^{-2}} \left( -c + b(c-1)q^{-1} + q^{-2} \right)^{j-1}
\]
\[
\Psi_{2j}(q, \alpha) = C_2 \frac{q^{-2}}{1 + b(c-1)q^{-1} - cq^{-2}} \left( -c + b(c-1)q^{-1} + q^{-2} \right)^{j-1}
\]

(4.3.3)

\( j = 1, 2, \ldots, n_K / 2 \)

\( \alpha \) is a complex parameter, |\( \alpha \)| \( \leq 1 \). The coefficients of eqn (4.3.3) are defined as

\[
a = \left( 1 + \alpha \alpha^* \right) / (\alpha + \alpha^*)
\]

\[
b = 1 / a
\]

\[
c = -\alpha \alpha^*
\]

(4.3.4)

\[
C_1 = -b \sqrt{1 - c^2}
\]

\[
C_2 = \sqrt{(1 - c^2)(1 - b^2)}
\]

where \( \alpha^* \) denotes complex conjugate.

The Kautz structure is used for modelling resonant systems. With \( \alpha \) selected, the model (4.3.1) is a fixed-pole ARX model with multiple poles at \( \alpha \). The Kautz parameter is selected based on prior knowledge about the system, or it is adjusted during the identification procedure, such that the main resonance of the model equals the main resonance of the system. Kautz-model identification is performed in function \texttt{l1kautz}.

During identification of Kautz models, the model (4.3.1) is transferred into state space form

\[
x(t+1) = Fx(t) + Gu(t)
\]

\[
y(t) = BTx(t) + \xi(t)
\]

(4.3.5)

where \( B \) is a row vector of Kautz coefficients. The state space model (4.3.5) is in the toolbox function initialised with \( x(0) = 0 \), and simulated for a number of time intervals in order to let the state vector converge. A part of the input/output data pairs is thus used for initialisation and cannot be used in the LSAD minimisation. The number of input/output data pairs which must be used for initialisation depends on the selected value \( \alpha \).

4.4 Combined models

The function \texttt{l1klt} uses a model which is a combination of Kautz models, Laguerre models and a FIR model. The transfer function is a sum of \( m_K \) Kautz transfer functions, \( m_L \) Laguerre transfer functions and one FIR transfer function,

\[
y(t) = \left[ B_K(q, \alpha_1) + \cdots + B_K(q, \alpha_{m_K}) + B_L(q, \beta_1) + \cdots + B_L(q, \beta_{m_L}) + B(q) \right] u(t) + \xi(t)
\]

(4.4.1)

The polynomials \( B_K, B_L \) and \( B \) are defined by equations (4.3.2), (4.2.2) and (4.3), respectively.
During identification of combined models, the model (4.4.1) is transferred into state space form

\[
\begin{align*}
    x(t+1) &= Fx(t) + Gu(t) \\
    y(t) &= Bx(t) + \xi(t)
\end{align*}
\] (4.4.2)

The state space model (4.4.2) is in the toolbox function initialised with \(x(0) = 0\), and simulated for a number of time intervals in order to let the state vector converge. A part of the input/output data pairs is thus used for initialisation and cannot be used in the LSAD minimisation. The number of input/output data pairs which must be used for initialisation depends on the selected values \(\alpha\) and \(\beta\).

### 4.5 Model validation

Model validation is used to check the identified nominal model on another set of data. Some functions, which use state models as model structures (l1firlgr, l1kautz and l1klt) perform an automatic model validation on the rest of the given data sequence, which has not been used for estimation. The result from the model validation is the average absolute prediction error,

\[
R_{val} = \frac{1}{N_{val}} \sum \left| \xi(t) \right|_{validation\ set}
\] (4.5.1)

This value can be compared to the average prediction error in the estimation,

\[
R_{id} = \frac{1}{N} \sum \left| \xi(t) \right|_{estimation\ set}
\] (4.5.2)

### 5. ESTIMATING UNCERTAINTY MODELS

We consider uncertainty models of the form

\[
y(t) = \left[ (\hat{G} + \Delta G)*u \right](t) + \xi(t)
\] (5.1)

We test the feasibility of an uncertainty model by testing that eqn (5.1) holds for \(t \in \{\text{a validation set}\}\), for some \(\Delta G\) satisfying \(\|\Delta G\| \leq \rho(N)\) and for some bounded \(\xi(t)\) such that \(1/N \sum \xi(t) \leq \varepsilon\). This procedure is followed in the function l1firud for a time varying uncertainty, \(\Delta G(t)\), of FIR structure. The uncertainty model in function l1firud is

\[
y(t) = \left[ B(q) + \Delta B(t, q) \right]u(t-n_k) + \xi(t)
\] (5.2)

where \(B(q)\) contains the nominal model.

The actual computational procedure of function l1firud is the following: given a nominal model \(B(q)\) (and \(n_k\)) and a value \(\rho\) we minimise the LSAD criterion (3.3) with respect to \(\Delta B(t, q)\) and \(\xi(t)\). l1firud uses piece-wise constant values for \(\Delta B(t, q)\).
The procedure is more closely described by Gustafsson and Mäkilä (1993a, 1993b), where also an example is given.

Another procedure is described by Gustafsson and Mäkilä (1993b) and Mäkilä et al (1994), and realised in function \texttt{l1fpunc}. Here the uncertainty model is of a fixed-pole ARX-model structure,

\[
A(q)y(t) = B(q)u(t) + \Delta B(q)u(t-n_k) + \xi(t), \quad t = 0,1,\ldots,N-1
\]  

(5.3)

where \( A \) is given as a fixed polynomial and \( B \) is the identified nominal \( B \)-polynomial. The size of \( \Delta B \) is estimated by the following procedure, where the order of \( \Delta B(q) \) is given, \( m \), and \( n_k \) is given,

\[
\kappa(\varepsilon, m, N) = \max_{\Delta B(q)} \| \Delta B(q) \|
\]

such that eqn (5.3) is satisfied and

\[
\frac{1}{N} \sum_{t=0}^{N-1} |\xi(t)| \leq \varepsilon.
\]  

(5.4)

The quantity \( \kappa(\varepsilon, m, N) \) can be interpreted as the maximal unfalsified value of \( \| \Delta B \| \) within the chosen uncertainty structure. The procedure involves the solution of \( 2^m \) linear programs, and can thus be very demanding in computer resources for larger \( m \).

The same procedure is used in function \texttt{l1fpoecu} for an uncertainty model of output error structure,

\[
y(t) = \frac{B(q)}{F(q)}u(t) + \frac{\Delta B(q)}{F(q)}u(t-n_k) + \xi(t), \quad t = 0,1,\ldots,N-1.
\]  

(5.5)

We consider the problem (5.4) to be a procedure for black-box estimation of uncertainty, that is for estimating uncertainty without prior knowledge of \( \varepsilon \). We calculate \( \kappa(\varepsilon, m, N) \) for different values of \( m \) and \( \varepsilon \), and plot \( \kappa(\varepsilon, m, N) \) as functions of \( m \) and \( \varepsilon \). With \( \varepsilon \) bigger than the actual value, \( \kappa(\varepsilon, m, N) \) should be linearly increasing with \( \varepsilon \), but at values of \( \varepsilon \) less than the actual value \( \kappa(\varepsilon, m, N) \) decreases rapidly. An estimate of \( \kappa(\varepsilon, m, N) \) is thus obtained at a point where the \( \kappa \)-curve turns into a straight line.

A tight lower limit is given by

\[
\kappa_*(m, N) \equiv \inf_{\varepsilon} \kappa(\varepsilon, m, N)
\]  

(5.6)

where the infimum is taken over all \( \varepsilon > 0 \) such that the constraints (5.3) and (5.4) holds. The quantity \( \kappa_*(m, N) \) is conveniently calculated by minimising the LSAD criterion (3.3) with respect to \( \Delta B(q) \) and \( \xi(t) \) for the model (5.3). This is the same procedure as used in function \texttt{l1firud} for an uncertainty model (5.2). The corresponding function \texttt{l1fpud} calculates (5.6) for the fixed-pole model (5.3) with a time invariant uncertainty.
Example. We have simulated a system with transfer function \( G(z) = \frac{(2z-1.40z^2)}{(1-1.40z+0.48z^2)} \), with \( \{v(t)\} \) being a realisation of Gaussian white noise with variance 0.02, and \( \{u(t)\} \) being a PRBS signal of unit amplitude. A nominal model (4.7) was estimated using l1fpoe with \( F(q) = 1 - 0.8q^{-1} \), which gave the model a pole equal to the dominating pole of the system. With \( n_B = 2 \), \( n_k = 1 \) and \( N = 100 \) we obtained for a specific simulation run \( B(q) = 2.0967q^{-1} - 0.3705q^{-2} \). Writing \( G(z) = B(z)/F(z) \), we get the size of the model error expressed in the form of the uncertainty model (5.3) as \( \|\Delta B\| = 0.5672 \).

\( \kappa \)-values were then estimated for \( m = 4, 6, 8 \) with function l1fpunc, using both the same data as used in the estimation of the nominal model, and validation data with \( N = 100 \). The results are presented in Figure 1. In the figure the plots for the data used also in the estimation of the nominal model are indicated in solid. The true averages (E) of the absolute values of the noise \( \nu(t) \) are also indicated for the two data sets. Furthermore, \( \kappa_s \)-values calculated with l1fpud are indicated with circles. The results indicate that even \( \kappa_s(m,N) \) estimates the size of the uncertainty (error) in \( B \) remarkably well for \( m = 6, 8 \). For \( m = 4 \) there is still considerable unmodelled dynamics outside the chosen uncertainty structure, and so in this case \( \kappa_s \) underestimates the true error more clearly. Note that the steepness of the curves varies in a systematic manner. Clearly random effects start to influence more at the last steep part of the curves. Thus the plots allow us to make a realistic choice of the size of the modelling uncertainty.

6. REFERENCES


ktz2fir

Purpose
Expand a Kautz model into a truncated FIR model.

Synopsis
B = ktz2fir(BK, alpha, n)

Description
The Kautz model, eqn (4.3.1), is approximated by a FIR model, eqn (4.6).

\[ BK = \begin{bmatrix} b_1 & b_2 & \cdots & b_{n_k} \end{bmatrix} \]

alpha is the complex Kautz parameter, \( \alpha \in \mathbb{C}, \quad |\alpha| < 1 \). n is the order of the resulting FIR model.

The result is the row vector of FIR coefficients, \( B = \begin{bmatrix} b_1 & b_2 & \cdots & b_n \end{bmatrix} \), with \( n_k = 0 \).
**Purpose**

Estimate the parameters of an ARX or AR model minimising the $l_1$ norm of the prediction error.

**Synopsis**

$[A, B, \text{stat}, \text{epsilon}, \text{err}, \text{it}, \text{res}] = \text{l1arx}(yu, nn, \text{epsilon}, [E f], [C d])$

**Description**

The estimated model is of a single output, multiple input ARX structure,

$$A(q) y(t) = B_1(q) u_1(t-n_{k1}) + \cdots + B_{n_u}(q) u_{n_u}(t-n_{knu}) + \xi(t)$$

with the $A$- and $B$-polynomials defined by eqs(4.2-3). The minimisation criterion is the LSAD criterion, eqn (3.3).

$yu$ contains the output-input data, $yu = [y u]$, where $y$ is a column vector containing $y(t)$ for $t = 0, 1, \ldots, N-1$. $u$ is a matrix with input sequences as columns, $u = [u_1 \ldots u_{n_u}]$.

$nn$ defines the orders of polynomials $A$ and $B$, and the delays associated with each input, $nn = [n_a \ n_{b1} \ \cdots \ n_{b_{n_{nu}}} \ n_{k1} \ \cdots \ n_{k_{n_{nu}}}]$.

$\text{epsilon} = [\epsilon_- \ \epsilon_+]$ are the error priors, eqn (3.9). If no constraints are applied $\text{epsilon}$ should be given the value $[0 \ 0]$.

$[E f]$ and $[C d]$ define inequality ($E x \leq f$) and equality constraints ($Cx=d$), where $x$ is the parameter vector,

$$x = [-a_1 \ \cdots \ -a_{n_a} \ b_{1,1} \ \cdots \ b_{1,n_{b1}} \ b_{2,1} \ \cdots \ b_{2,n_{b2}} \ \cdots \ b_{n_u,1} \ \cdots \ b_{n_u,n_{b_{nu}}}]^T$$

The result is given in $A$ and $B$:

$$A = \begin{bmatrix} 1 & a_1 & a_2 & \cdots & a_{n_a} \end{bmatrix}$$

$$B = \begin{bmatrix} 0_1 & \cdots & 0_{n_{k1}} & b_{1,1} & b_{1,2} & \cdots & b_{1,n_{b1}} & 0 & \cdots & 0 \\ 0_1 & \cdots & 0_{n_{k2}} & b_{2,1} & b_{2,2} & \cdots & b_{2,n_{b2}} & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \cdots & \vdots & \ddots & \vdots & \vdots \\ 0_1 & \cdots & 0_{n_{knu}} & b_{n_u,1} & b_{n_u,2} & \cdots & b_{n_u,n_{b_{nu}}} & 0 & \cdots & 0 \end{bmatrix}$$

$\text{stat}$ indicates the status of the result:

- $\text{stat} = 0$: optimal solution found
- $\text{stat} = 1$: no feasible solution found
- $\text{stat} = 2$: calculations terminated prematurely due to rounding errors
- $\text{stat} = 3$: maximum number of iterations reached.
\[ \text{epsil} = \left[ \hat{\xi}_- \quad \hat{\xi}_+ \right] \] returns an error estimate, \( \min_t \xi(t) \geq \hat{\xi}_- \), \( \max_t \xi(t) \leq \hat{\xi}_+ \).

err returns the value of the criterion function, it returns the number of iterations and res returns the residual vector.

\[
\text{res} = \begin{bmatrix}
    y - \hat{y} \\
    d - Cx \\
    f - Ex \\
    \xi_+ - (y - \hat{y}) \\
    (y - \hat{y}) - \xi_-
\end{bmatrix}
\]

**Note**

The exact number of data used for identification is

\[ N = \text{dim}(y) - \max_i \left\{ n_{ki} + \max_j n_{bj} \right\} + 1. \]
**Purpose**

Estimate the parameters of a FIR model minimising the $l_1$ norm of the prediction error.

**Synopsis**

$$[B, \text{FPE}, \text{stat}, \epsilon_{\text{epsil}}, l\text{1norm}, \text{err}, \text{it}] = l1fir(yu, nn, \epsilon, [E f], [C d], \lambda, \gamma_1, \gamma_2)$$

**Description**

The estimated model is of a FIR structure, eqn(4.6) The minimisation criterion is the augmented LSAD criterion, eqn (3.4).

$yu$ contains the output-input data, $yu = [y u]$, where $y$ and $u$ are column vectors containing $y(t)$ and $u(t)$ for $t = 0, 1, ..., N-1$.

$nn = [n_b \ n_k]$.  

$\epsilon_{\text{epsil}} = [\epsilon_- \ \epsilon_+]$ are the error priors, eqn (3.9). If no constraints are applied $\epsilon_{\text{epsil}}$ should be given the value $[0 \ 0]$.  

$[E f]$ and $[C d]$ define inequality $(Ex \leq f)$ and equality constraints $(Cx = d)$, where $x$ is the parameter vector, $x = [b_1 \ b_2 \ \cdots \ b_{n_b}]^T$.  

$\lambda$ is the weight factor ($\lambda$) of eqn (3.4), $\gamma_1$ and $\gamma_2$ are the constraints $\gamma_1$ and $\gamma_2$ of eqns (3.7) and (3.8), respectively. $\gamma_1 \leq 0$ and $\gamma_2 \leq 0$ imply no constraints.

The result is given in $B = [0 \ \cdots \ 0 \ b_1 \ b_2 \ \cdots \ b_{n_b}]$, with the first $n_k$ elements zero.

FPE is Akaike's Final Prediction Error (Ljung, 1991).

$\text{stat}$ indicates the status of the result:

- $stat=0$: optimal solution found
- $stat=1$: no feasible solution found
- $stat=2$: calculations terminated prematurely due to rounding errors
- $stat=3$: maximum number of iterations reached.

$\epsilon_{\text{epsil}} = [\epsilon_- \ \epsilon_+]$ returns an error estimate, $\min_t \xi(t) \geq \hat{\epsilon}_-, \ \max_t \xi(t) \leq \hat{\epsilon}_+$.  

$l1\text{norm}$ is the averaged one-step ahead prediction error, $l1\text{norm} = \|y - \hat{y}\|_1/N$.  

$\text{err}$ returns the value of the criterion function and $\text{it}$ returns the number of iterations.

The exact number of data used for identification, $N = \text{dim}(y) - (n_k + n_b) + 1$.  

18
**Purpose**

Estimate the parameters of a FIR model in the Laguerre domain minimising the $l_1$ norm of the prediction error. l1firlgr performs also a model validation.

**Synopsis**

$$[Bl, FPE, stat, epsil, l1norm, l1val, err, it] = l1firlgr(yu, nn, alpha, t0, nt, epsilon, [E f], [C d], gamma1)$$

**Description**

The estimated model is of a FIR structure in the Laguerre domain, eqns (4.2.1) to (4.2.3). The minimisation criterion is the LSAD criterion, eqn (3.3).

Matrix $yu$ contains the output-input data, $yu = [y u]$, where $y$ and $u$ are column vectors containing $y(t)$ and $u(t)$ for $t = 0, 1, ..., N-1$.

$nn = [0, n_L]$. alpha is the Laguerre parameter $\alpha$, $|\alpha| < 1$.

$t0$ denotes the first data pair ($y, u$) to be used for identification. Data pairs $\{y(1), u(1)\}$ to $\{y(t_0-1), u(t_0-1)\}$ are used to let the state space form (4.2.4) of the Laguerre model approximately converge to the correct state before starting the estimation procedure.

$nt$ is the number of data pairs used for identification.

$\epsilon = [\epsilon_-, \epsilon_+]$ are the error priors, eqn (3.9). If no constraints are applied epsilon should be given the value [0 0].

$[E f]$ and $[C d]$ define inequality and equality constraints, which are not implemented in this version.

$\gamma_1$ is the constraint $\gamma_1$ of eqn (3.7). $\gamma_1 \leq 0$ implies no constraint.

The result is given in $Bl = \begin{bmatrix} b_1 & b_2 & \cdots & b_{n_L} \end{bmatrix}$.

$FPE$ is Akaike’s Final Prediction Error (Ljung, 1991).

$stat$ indicates the status of the result:

- stat=0: optimal solution found
- stat=1: no feasible solution found
- stat=2: calculations terminated prematurely due to rounding errors
- stat=3: maximum number of iterations reached.

$\epsilon = [\epsilon_-, \epsilon_+]$ returns an error estimate, $\min_{t} \xi(t) \geq \hat{\epsilon}_-$. $\max_{t} \xi(t) \leq \hat{\epsilon}_+$.

$l1norm$ is the averaged one-step ahead prediction error, $l1norm = R_{id} = \|y - \hat{y}\|_1 / N$, where $N = nt$. 


llval returns the averaged l1 norm of the prediction error, \( llval = R_{val} = \|y - \hat{y}\|_1 / (n - t_0 - n_t - 1) \), for the model validation data set, \( \{y(t_0 + n_t + 1), u(t_0 + n_t + 1)\} \) to \( \{y(n), u(n)\} \), where \( n \) is the total length of the times series \( y(t) \) and \( u(t) \).

err returns the value of the criterion function and it returns the number of iterations.
Purpose
Estimate the parameters of a FIR model minimising the $l_1$ norm of the prediction error. Take slowly time-varying unmodelled dynamics into consideration.

Synopsis
$$[B, FPE, stat, epsilon, l1norm, err, it, Delta] = l1firud(yu, nn, epsilon, [E f], [C d], lambda, gamma1, gamma2, kappa, ndelta, tdelta)$$

Description
The estimated model is of a FIR structure, eqn (5.2). The minimisation criterion is the augmented LSAD criterion, eqn (3.4).

Only the first lefthand argument and two first righthand arguments are obligatory.

Matrix $yu$ contains the output-input data, $yu = [y u]$, where $y$ and $u$ are column vectors containing $y(t)$ and $u(t)$ for $t = 0, 1, ..., N-1$. $nn = [n_b ~ n_k]$.

$epsilon = [\varepsilon_- ~ \varepsilon_+]$ are the error priors, eqn (3.9). If no constraints are applied $epsilon$ should be given the value $[0 0]$.

$[E f]$ and $[C d]$ define inequality ($Ex \leq f$) and equality constraints ($Cx=d$), where $x$ is the parameter vector, $x = [b_1 ~ b_2 ~ \ldots ~ b_{n_b}]^T$.

$lambda$ is the weight factor ($\lambda$) of eqn (3.4), $gamma1$ and $gamma2$ are the constraints $\gamma_1$ and $\gamma_2$ of eqns (3.7) and (3.8), respectively. $gamma1 \leq 0$ and $gamma2 \leq 0$ imply no constraints.

$kappa$ is a constraint ($\kappa$) on the $l_1$-norm of the unmodelled dynamics,
$$\sum_{i=1}^{n_{\delta}} |\delta_i(t)| \leq \kappa \quad \forall \quad t \in \{0,1,\ldots,N-1\}$$
where $\delta_i$ are the coefficients of the polynomial $\Delta B(t, q)$, and $n_{\delta} = ndelta$ is the order of the unmodelled dynamics.

tdelta ($t_{\delta}$) is the length of the time periods with constant unmodelled dynamics.
The unmodelled dynamics is identified as a set of piece-wise constant polynomials for time periods of length $t_{\delta}$, $1 \leq t_{\delta} \leq N$.
$$\Delta B(t, q) = \Delta B_i(q), \quad i = 1, \ldots, n_i, \quad t = (i-1) \cdot t_{\delta} + 1, \ldots, i \cdot t_{\delta}.$$ The last period can be shorter than $t_{\delta}$. $N$ is the total number of values in the time series used for identification and $nt$ is defined as ceil($N/t_{\delta}$).

$N = \dim(y) - n_k - \max(n_b, n_{\delta})$. If tdelta is greater than $N$, then $t_{\delta} = N$ is used.

The result is given in $B = [0 \cdots 0 ~ b_1 ~ b_2 ~ \cdots ~ b_{n_b}]$, with the first $n_k$ elements zero.

stat indicates the status of the result:
stat=0: optimal solution found
stat=1: no feasible solution found
stat=2: calculations terminated prematurely due to rounding errors
stat=3: maximum number of iterations reached.

epsil = [ε_- ε_+] returns an error estimate, \( \min_t \xi(t) \geq \hat{\xi}_- \), \( \max_t \xi(t) \leq \hat{\xi}_+ \).

l1norm is the averaged one-step ahead prediction error, \( \text{l1norm} = \|y - \hat{y}\|_1/N \).

err returns the value of the criterion function and it returns the number of iterations.

Delta returns the coefficients of the unmodelled dynamics in the form of a matrix where the \( n_k \) leftmost elements are zero,

\[
\begin{bmatrix}
0 & \cdots & 0 & \delta_{1,1} & \cdots & \delta_{1,n_6} \\
0 & \cdots & 0 & \delta_{2,1} & \cdots & \delta_{2,n_6} \\
\vdots & \ddots & \vdots & \ddots & \ddots & \vdots \\
0 & \cdots & 0 & \delta_{n,1} & \cdots & \delta_{n,n_6}
\end{bmatrix}
\]

Algorithm

Note that the estimation of unmodelled dynamics can be computationally very demanding. The vector of parameters to be estimated will be \( \text{dim}(x) = n_b + 2n_\delta \) and the number of inequalities will be \( \text{dim}(f') = \text{dim}(f') + 1 + 3n_\delta + n_t \). The epsilon constraint will still increase the dimension of the inequalities with \( 2N \). The lambda coefficient will increase the dimension of the equalities with \( n_b \). The gamma1 constraint will increase the dimension of the estimation vector with \( n_b \) and the dimension of the inequalities with \( 3n_b + 1 \). The gamma2 constraint will also increase the dimension of the estimation vector with \( n_b \) and the dimension of the inequalities with \( 3n_b + 1 \).
**Purpose**
Estimate the parameters of a fixed-pole ARX model minimising the $l_1$ norm of the prediction error.

**Synopsis**

\[
[B, \text{FPE, stat, epsilon, l1norm, err, it} ] = \text{l1fparx}(yu, \text{nn, A, epsilon, [E f], [C d]})
\]

**Description**
The estimated model is of an ARX structure, eqn (4.5) with a given fixed $A$-polynomial. The minimisation criterion is the LSAD criterion, eqn (3.3).

Only the first lefthand argument and the two first righthand arguments are obligatory.

Matrix $yu$ contains the output-input data, $yu = [y \ u]$, where $y$ and $u$ are column vectors containing $y(t)$ and $u(t)$ for $t = 0, 1, ..., N-1$.

\[
nn = \begin{bmatrix} n_b & n_k \end{bmatrix}.
\]

$A$ is a row vector containing the coefficients of the apriori given polynomial $A(q)$, with the first element equal to one.

$\text{epsilon} = \begin{bmatrix} \epsilon_- & \epsilon_+ \end{bmatrix}$ are the error priors, eqn (3.9). If no constraints are applied $\text{epsilon}$ should be given the value $[0 \ 0]$.

$[\text{E f}]$ and $[\text{C d}]$ define inequality ($\text{Ex} \leq \text{f}$) and equality constraints ($\text{Cx} = \text{d}$), where $x$ is the parameter vector, $x = \begin{bmatrix} b_1 & b_2 & \cdots & b_{n_p} \end{bmatrix}^T$.

The result is given in $B = \begin{bmatrix} 0 & \cdots & 0 & b_1 & b_2 & \cdots & b_{n_p} \end{bmatrix}$, with the first $n_k$ elements zero.

Stat indicates the status of the result:

- stat=0: optimal solution found
- stat=1: no feasible solution found
- stat=2: calculations terminated prematurely due to rounding errors
- stat=3: maximum number of iterations reached.

$\text{epsilon} = \begin{bmatrix} \epsilon_- & \epsilon_+ \end{bmatrix}$ returns an error estimate, $\min_{t} \xi(t) \geq \hat{\epsilon}_-, \max_{t} \xi(t) \leq \hat{\epsilon}_+.$

$l1\text{norm}$ is the averaged one-step ahead prediction error, $l1\text{norm} = \|y - \hat{y}\|_1/N.$

Err returns the value of the criterion function and it returns the number of iterations.

The exact number of data used for identification, $N = \dim(y) - \max\{n_a - 1, n_k + n_b - 1\}$.  

23
**Purpose:**
Estimation of the parameters of a fixed-pole Output Error model using $l_1$-approximation.

**Synopsis:**
$[B, FPE, stat, epsil, l1norm, err, it] = l1fpoe(yu, nn, F, epsilon, [E f], [C d])$

**Description:**
The identification model is of the OE-model structure, eqn (4.7), with a given fixed $F$-polynomial. The minimisation criterion is the LSAD criterion (3.3).

Only the first lefthand and the two first righthand arguments are obligatory.

Matrix $yu$ contains the output-input data $yu=[y u]$, where $y$ and $u$ are column vectors. $nn$ is given as
$$nn = \begin{bmatrix} n_b & n_k \end{bmatrix}.$$

$F$ is a row vector containing the coefficients of the given polynomial $F(q)$, with the first element equal to one.

$epsilon = [\epsilon_- \epsilon_+]$ are apriori given minimum and maximum constraints on the disturbance, eqn (3.9). $epsilon$ can be set to $[0 0]$ if no constraints should be applied.

$[E f]$ and $[C d]$ define inequality $(Ex \leq f)$ and equality constraints $(Cx=d)$, where $x$ is the parameter vector, $x = \begin{bmatrix} b_1 & b_2 & \cdots & b_n \end{bmatrix}^T$.

The result is stored in the row vector $B$, with $nk+nb$ columns and with the first $nk$ elements equal to zero,
$$B = \begin{bmatrix} 0 & \cdots & 0 & b_1 & b_2 & \cdots & b_n \end{bmatrix}.$$

$stat$ indicates the status of the result:
- $stat=0$: optimal solution found
- $stat=1$: no feasible solution
- $stat=2$: calculations terminated prematurely due to rounding errors
- $stat=3$: maximum number of iterations reached.

The maximum number of iterations is set to 500.

$epsil$ returns an estimate of the minimum and maximum value of the disturbance. The format is $epsil = [\hat{\epsilon}_- \hat{\epsilon}_+]$.

$l1norm$ returns the average of the absolute prediction error, that is $l1norm = \|y - \hat{y}\|_1/N$.

$err$ returns the value of the criteria function and it returns the number of iterations.
**Purpose:**

Estimation of uncertainty models for fixed-pole Output Error nominal models.

**Synopsis:**

\[ \text{[kappa, DeltaB, P, LPlog]} = \text{l1fpoeu(yu, nn, F, B, epsil, trace)} \]

**Description:**

The uncertainty model is of output error type, eqn (5.5). The algorithm estimates the size of \( \Delta B(q) \) as the maximum unfalsified value, \( \kappa(\epsilon, m, N) \) for a given order of \( \Delta B(q) \), according to the procedure (5.4) with respect to the model (5.4) instead of (5.3).

Matrix \( yu \) contains the output-input data \( yu = [y \; u] \), where \( y \) and \( u \) are column vectors containing \( y(t) \) and \( u(t) \) for \( t = 0, 1, ..., N-1 \). \( nn \) is given as \( nn = [m \; nk] \) where \( m \) defines the order of \( \Delta B(q) \) and \( nk \) defines the time delay associated with the input.

\( F \) is a row vector containing the coefficients of the polynomial \( F(q) \), with the first element equal to one. \( B \) is a row vector containing the coefficients of the polynomial \( B(q) \). Normally the first element of \( B \) is zero. \( \text{epsil} \) is the constraint \( \epsilon \) on the noise in equation (5.4).

\( \text{trace} = 1 \) gives an intermediate result after each of the \( 2^m \) maximisations. \( \text{trace} = 0 \) suppresses intermediate output.

The output variables are \( \text{kappa} = \kappa(\epsilon, m, N) \). \( \text{DeltaB} \) is a row vector containing the coefficients of the polynomial \( \Delta B(q) \), which gives a maximum \( \kappa(\epsilon, m, N) \). \( P \) is an \( (m, 2) \)-matrix giving the minimum and maximum values for each coefficient of \( \Delta B(q) \) from the \( 2^m \) linear programs. \( \text{LPlog} \) is a column vector with \( 2^m \) entries, describing the state of each linear program, 0 means that maximum is achieved, -1 means infeasibility and 1 means unboundness.

**Algorithm**

The maximisation problem is decomposed into \( 2^m \) standard linear programming problems, which are solved using the routine lpsimplx.
Purpose
Estimate a lower limit $\kappa_*$, eqn (5.6), for the size of the uncertainty, according to the ARX uncertainty model of equation (5.3). The nominal model is to be given.

Synopsis
\[ [\varepsilon, \kappa, \Delta B, \text{LPlog}] = \text{l1fpud}(y_u, nn, A, B) \]

Description
Matrix $y_u$ contains the output-input data, $y_u = [y \ u]$, where $y$ and $u$ are column vectors containing $y(t)$ and $u(t)$ for $t = 0, 1, ..., N-1$. $nn = \begin{bmatrix} m & n_k \end{bmatrix}$.

$A$ is a row vector containing the coefficients of the polynomial $A(q)$, with the first element equal to one. $B$ is a row vector containing the coefficients of the polynomial $B(q)$, with the first element equal to zero. $A$ and $B$ defines the nominal model.

$\varepsilon$ is the value of the LSAD criterion (3.3), that is the value of $\varepsilon$ which gives the infimum of eqation (5.6).

$kappa$ is the resulting value $\kappa_*(m, N)$, equation (5.6).

$\Delta B$ gives the resulting coefficients of $\Delta B(q)$, $\Delta B = \begin{bmatrix} \delta_1 & \delta_2 & \cdots & \delta_m \end{bmatrix}$, with the first element nonzero.

LPlog describes the state of the linear program, 0 means that minimum is achieved, -1 means infeasibility and 1 means unboundness.
**Purpose**

Estimation of uncertainty models for fixed-pole ARX nominal models.

**Synopsis**

\[
[kappa, DeltaB, P, LPlog] = l1fpunc(yu, nn, A, B, epsil, trace)
\]

**Description:**

The uncertainty model is of ARX structure, equation (5.3). The algorithm estimates the size of \( \Delta B(q) \) as the maximum unfalsified value, \( \kappa(\varepsilon, m, N) \) for a given order of \( \Delta B(q) \), according to the procedure (5.4).

Matrix \( yu \) contains the output-input data \( yu = [y \ u] \), where \( y \) and \( u \) are column vectors containing \( y(t) \) and \( u(t) \) for \( t = 0, 1, \ldots, N-1 \). \( nn \) is given as \( nn = [m \ nk] \) where \( m \) defines the order of \( \Delta B(q) \) and \( nk \) defines the time delay associated with the input.

\( A \) is a row vector containing the coefficients of the polynomial \( A(q) \), with the first element equal to one. \( B \) is a row vector containing the coefficients of the polynomial \( B(q) \). Normally the first element of \( B \) is zero. \( A \) and \( B \) defines the nominal ARX model.

\( epsil \) is the constraint \( \varepsilon \) on the noise in equation (5.4).

\( trace=1 \) gives an intermediate result after each of the \( 2^m \) maximisations. \( trace=0 \) suppresses intermediate output.

The output variables are \( kappa = \kappa(\varepsilon, m, N) \). \( DeltaB \) is a row vector containing the coefficients of the polynomial \( \Delta B(q) \), which gives a maximum \( \kappa(\varepsilon, m, N) \). \( P \) is an \( (m, 2) \)-matrix giving the minimum and maximum values for each coefficient of \( \Delta B(q) \) from the \( 2^m \) linear programs. \( LPlog \) is a column vector with \( 2^m \) entries, describing the state of each linear program, 0 means that maximum is achieved, -1 means infeasibility and 1 means unboundness.

**Algorithm:**

The maximization problem above is decomposed into \( 2^m \) standard linear programming problems, which are solved using the routine lpsimplx.
Purpose
Estimate the parameters of a discrete Kautz model minimising the $l_1$ norm of the prediction error. l1kautz performs also a model validation.

Synopsis
[Bk, FPE, stat, epsil, l1norm, l1val, err, it, FGH] = l1kautz(yu, nn, alpha, t0, nt, epsilon, [E f], [C d], gamma1)

Description
The estimated model is of a Kautz structure, eqns (4.3.1) to (4.3.4). The minimisation criterion is the LSAD criterion, eqn (3.3).

Matrix yu contains the output-input data, $yu = [y u]$, where $y$ and $u$ are column vectors containing $y(t)$ and $u(t)$ for $t = 0, 1, ..., N-1$.

$nn = [0 \ n_K]$. alpha is the complex Kautz parameter $\alpha$, $|\alpha| < 1$.

t0 denotes the first data pair $(y, u)$ to be used for identification. Data pairs $(y(1),u(1))$ to $(y(t_0-1),u(t_0-1))$ are used to let the state space form (4.3.5) of the Kautz model approximately converge to the correct state before starting the estimation procedure.

nt is the number of data pairs used for identification.

$\text{epsilon} = [\epsilon_- \ \epsilon_+]$ are the error priors, eqn (3.9). If no constraints are applied epsilon should be given the value $[0\ 0]$.

$[E f]$ and $[C d]$ define inequality and equality constraints, which are not implemented in this version.

$\gamma_1$ is the constraint $\gamma_1$ of eqn (3.7). $\gamma_1 \leq 0$ implies no constraint.

The result is given in $Bk = \begin{bmatrix} b_1 & b_2 & \cdots & b_{n_K} \end{bmatrix}$.

FPE is Akaike's Final Prediction Error (Ljung, 1991).

stat indicates the status of the result:
stat=0: optimal solution found
stat=1: no feasible solution found
stat=2: calculations terminated prematurely due to rounding errors
stat=3: maximum number of iterations reached.

$\text{epsilon} = [\epsilon_- \ \epsilon_+]$ returns an error estimate, $\min_t \xi(t) \geq \hat{\epsilon}_-, \max_t \xi(t) \leq \hat{\epsilon}_+$.

$l1\text{norm}$ is the averaged one-step ahead prediction error, $l1\text{norm} = R_{id} = \|y - \hat{y}\|_1 / N$, where $N = nt$. 

28
llval returns the averaged $l_1$ norm of the prediction error, $llval = R_{val} = \| y - \hat{y} \|_1 / (n - t_0 - n_t - 1)$, for the model validation data set, $\{ y(t_0 + n_t + 1), u(t_0 + n_t + 1) \}$ to $\{ y(n), u(n) \}$, where $n$ is the total length of the times series $y(t)$ and $u(t)$.

err returns the value of the criterion function and it returns the number of iterations.

FGH returns the state-space model (4.3.5) in the form $FGH = [F G H]$, where $H = BT$. 

---

29
**Purpose**

Estimate the parameters of a combination of discrete Kautz, Laguerre and FIR models minimising the $l_1$ norm of the prediction error. $l1klt$ performs also a model validation.

**Synopsis**

$$[F, G, B, FPE, \text{stat}, l1\text{norm}, l1\text{val}, \text{err}, \text{it}] = l1klt(yu, nn, kautz, laguerre, t0, nt)$$

**Description**

The estimated model is of a fixed-pole state space structure, eqn (4.4.2), with a special selection of poles according to equation (4.4.1). The minimisation criterion is the LSAD criterion, eqn (3.3).

Matrix $yu$ contains the output-input data, $yu = [y \ u]$, where $y$ and $u$ are column vectors containing $y(t)$ and $u(t)$ for $t = 0, 1, ..., N-1$.

$$nn = [n_K \ n_L \ n_b]$$

where $n_K$ is the even order of the Kautz polynomials, $n_L$ is the order of the Laguerre polynomials and $n_b$ is the order of the FIR polynomial (see equations (4.3.2), (4.2.2) and (4.3), respectively.

$kautz$ is a column vector of complex Kautz parameters $\alpha_i$, $|\alpha_i| < 1$, $i = 1...m_K$.

$laguerre$ is a column vector of real Laguerre parameters $\beta_i$, $|\beta_i| < 1$, $i = 1...m_L$.

$t0$ denotes the first data pair $(y, u)$ to be used for identification. Data pairs $\{y(1), u(1)\}$ to $\{y(t0-1), u(t0-1)\}$ are used to let the state space model (4.4.2) approximately converge to the correct state before starting the estimation procedure.

$nt$ is the number of data pairs used for identification.

The identified state model (4.4.2) is stored in matrices $F$, $G$ and $B$.

$FPE$ is Akaike's Final Prediction Error (Ljung, 1991).

$stat$ indicates the status of the result:

- $stat=0$: optimal solution found
- $stat=1$: no feasible solution found
- $stat=2$: calculations terminated prematurely due to rounding errors
- $stat=3$: maximum number of iterations reached.

$l1\text{norm}$ is the averaged one-step ahead prediction error, $l1\text{norm} = R_{id} = \|y - \hat{y}\|_1/N$, where $N = nt$.

$l1\text{val}$ returns the averaged $l_1$ norm of the prediction error, $l1\text{val} = R_{val}$

$$= \|y - \hat{y}\|_1/(n - t_0 - n_t - 1)$$

for the model validation data set.
\[ \{y(t_0 + n_t + 1), u(t_0 + n_t + 1)\} \] to \( \{y(n), u(n)\} \), where \( n \) is the total length of the time series \( y(t) \) and \( u(t) \).

err returns the value of the criterion function and it returns the number of iterations.
**Purpose**

Expand a Laguerre model into a truncated FIR model.

**Synopsis**

\[ B = lgr2fir(B_L, \alpha, n) \]

**Description**

The Laguerre model, eqn (4.2.1), is approximated by a FIR model, eqn (4.6).

\[ \text{BK} \text{ is a row vector of Laguerre coefficients, } \text{BL} = \begin{bmatrix} b_1 & b_2 & \cdots & b_{n_L} \end{bmatrix}. \]

\[ \alpha \text{ is the real Laguerre parameter, } |\alpha| < 1. \]

\[ n \text{ is the order of the resulting FIR model.} \]

The result is the row vector of FIR coefficients, \[ B = \begin{bmatrix} b_1 & b_2 & \cdots & b_n \end{bmatrix}, \text{ with } n_k = 0. \]
Purpose
Solves linear programming problems using the simplex algorithm.

Synopsis
[x, obj, how] = lpsimplx(f, A1, b1, A2, b2, A3, b3)

Description
lpsimplx solves the linear programming problem
\[
\max_x \{ f^T x \} \quad \text{subject to} \quad A_1 x \leq b_1 \\
A_2 x = b_2 \\
A_3 x \geq b_3 \quad \text{and} \quad x \geq 0.
\]
where \( f \) is a vector of constant coefficients. The matrices \( A \) and vectors \( b \) are the coefficients of the linear constraints. It is recommended (but not required) that the vectors \( b_1, b_2 \) and \( b_3 \) contain only non negative entries. Negative entries in the \( b \)-vectors cause lpsimplx to rearrange the constraints before constructing the tableau used by the simplex algorithm.

x returns a solution to the linear problem.

obj returns the value of the objective function \( f^T x \) for the obtained solution.

how returns a string indicating the state of the result. how = 0 indicates that the obtained solution is a true solution to the linear program, how = -1 indicates that the problem is infeasible, e.g. there is no solution because of too restrictive constraints, and how = 1 indicates that the problem has an unbounded solution.

Arguments A3, b3 and A2, b2 can be omitted. Missing constraints can also be indicated with empty matrices.

Algorithm
The algorithm is based on the implementation of the simplex method by Kuenzi, et al. [1], and is copied from the fortran version of Press et al. [2]. This routine consists of a matlab function, lpsimplx.m, which constructs the tableau used in the program of Press et al., and a function, simplx.mex4, which is compiled from two fortran files, simplxg.f and simplx.f.

References
prbs

Purpose
Pseudo-random binary signal generator.

Synopsis
y = prbs(n, mist, mast)
y = prbs(n, mist, mast, F)

Description
prbs generates a column vector (dimension y(n,1)) with a pseudo-random sequence of numbers +1 and -1. The period between two switches (between +1 and -1 or vice versa) is stochastic variable with an approximate exponential distribution between a minimum switching period, mist, and a maximum switching period, mast. mist and mast must be given in discrete time periods (integers).

F is an optional parameter, 0.001 ≤ F ≤ 0.999. F = 0 would imply uniform distribution of switching periods between mist and mast. F = 1 would imply exponential distribution of switching periods between mist and mast. The default is F = 0.8.

prbs generates uniform random numbers using the MATLAB function rand. The seed of rand may be set before calling prbs.

Note
Version 4 of Matlab allows the user to change the function rand to give normally distributed random numbers. Make sure that rand gives uniformly distributed random numbers before using prbs.
Purpose
Laplace-distributed random number generator.

Synopsis
x = rndlapl
x = rndlapl(m, n)
x = rndlapl(m, n, var)

Description
rndlapl generates Laplace-distributed (doublesided exponential) random numbers. The probability density function is

\[ f_X(x) = \frac{1}{\sigma \sqrt{2}} e^{-\frac{|x|}{\sigma \sqrt{2}}} \]

where \(\sigma\) is the standard deviation. The expected value is 0.

rndlapl generates one random sample from the Laplace-distributed stochastic variable with variance 1. rndlapl(m, n) generates an m-by-n matrix with random entries from the Laplace-distributed stochastic variable with variance 1. rndlapl(m, n, var) generates random numbers from the Laplace-distributed stochastic variable with variance var.

Algorithm
rndlapl generates Laplace-distributed random numbers by transforming uniformly distributed random numbers, \(u \in [-1/2, 1/2]\), using the inverse of the distribution function,

\[ x = \text{sign}(u) \cdot \frac{\sigma}{\sqrt{2}} \ln(1 - 2|u|) \]

rndlapl generates Laplace-distributed random numbers using the standard MATLAB function rand combined with the shuffling routine of Bays and Durham [1]. The seed of rand may be set before calling rndlapl.

Note
Version 4 of Matlab allows the user to change the function rand to give normally distributed random numbers. Make sure that rand gives uniformly distributed random numbers before using rndlapl.

References
Purpose
Solves linear programming problems using the simplex algorithm.

Synopsis
[x, obj, how] = wlpsmplx(f, A1, b1, A2, b2, A3, b3)

Description
wlpsmplx solves the linear programming problem
\[
\max_{\mathbf{x}} \{ \mathbf{f}^T \mathbf{x} \} \quad \text{subject to} \quad \mathbf{A}_1 \mathbf{x} \leq \mathbf{b}_1 \\
\mathbf{A}_2 \mathbf{x} = \mathbf{b}_2 \\
\mathbf{A}_3 \mathbf{x} \geq \mathbf{b}_3 \quad \text{and} \quad \mathbf{x} \geq \mathbf{0}.
\]
where \( \mathbf{f} \) is a vector of constant coefficients. The matrices \( \mathbf{A} \) and vectors \( \mathbf{b} \) are the coefficients of the linear constraints. It is recommended (but not required) that the vectors \( \mathbf{b}_1, \mathbf{b}_2 \) and \( \mathbf{b}_3 \) contain only non negative entries. Negative entries in the \( \mathbf{b} \)-vectors cause wlpsmplx to rearrange the constraints before constructing the tableau used by the simplex algorithm.

\( \mathbf{x} \) returns a solution to the linear problem.

\( \mathbf{obj} \) returns the value of the objective function \( \mathbf{f}^T \mathbf{x} \) for the obtained solution.

\( \mathbf{how} \) returns a string indicating the state of the result. \( \mathbf{how} = 0 \) indicates that the obtained solution is a true solution to the linear program, \( \mathbf{how} = -1 \) indicates that the problem is infeasible, e.g. there is no solution because of too restrictive constraints, and \( \mathbf{how} = 1 \) indicates that the problem has an unbounded solution. \( \mathbf{how} = 2 \) indicates a solution with significant roundoff error and \( \mathbf{how} = 3 \) indicates violation of the iteration bound in function wsimpreg. The iteration bound for the phase 1 and phase 2 iterations are set at 10000. It can be changed in the file wsimreg.m.

Arguments \( \mathbf{A}_3, \mathbf{b}_3 \) and \( \mathbf{A}_2, \mathbf{b}_2 \) can be omitted. Missing constraints can also be indicated with empty matrices.

Algorithm
The algorithm is based on the implementation of the simplex method by Henry Wolkowicz, in functions wsimplex and wsimreg. These functions are rewritten forms of the interactive LP program from the files main.m and reg.m, respectively, of Henry Wolkowicz.

Compared to lpsimplex, wlpsmplx is much slower (4.5 times slower in an example with \( \dim(\mathbf{x}) = 200, \dim(\mathbf{b}_1) = 1, \dim(\mathbf{b}_2) = 92 \) and \( \dim(\mathbf{b}_3) = 0 \)). Contrary to lpsimplex, wlpsmplx is written completely in MATLAB language, and is thus simply portable between computers.

See also
wsimplex, lpsimplex
Purpose
Solves linear programming problems using the simplex algorithm.

Synopsis
[x, obj, how, iter] = wsimplex(A, b, c, m1, m2,)

Description
wsimplex solves the linear programming problem
\[
\min_{x} \{ c^T x \} \quad \text{subject to} \quad A x \leq b, \ x \geq 0.
\]
where \( c \) is a row vector of constant coefficients. The matrix \( A \) and vector \( b \) are the coefficients of the linear constraints. It is required that the vector \( b \) contains only non negative entries. The first \( m1 \) constraints are less or equal. The last \( m2 \) constraints are equality constraints. \( m1 + m2 \) must equal the row dimension of \( A \) and \( b \).

\( x \) returns a solution to the linear problem.

\( \text{obj} \) returns the value of the objective function \( c^T x \) for the obtained solution.

\( \text{how} \) returns a string indicating the state of the result. \( \text{how} = 0 \) indicates that the obtained solution is a true solution to the linear program, \( \text{how} = -1 \) indicates that the problem is infeasible, e.g. there is no solution because of too restrictive constraints, and \( \text{how} = 1 \) indicates that the problem has an unbounded solution. \( \text{how} = 2 \) indicates a solution with significant roundoff error and \( \text{how} = 3 \) indicates violation of the iteration bound in function wsimpreg. The iteration bound for the phase 1 and phase 2 iterations are set at 10000. It can be changed in the file wsimpreg.m.

\( \text{iter} \) returns the number of iterations performed in phase 2 (or in phase 1 if phase 2 was never performed).

Algorithm
The algorithm is based on the implementation of the simplex method by Henry Wolkowicz, in functions wsimplex and wsimpreg. These functions are rewritten from the interactive LP program from the files main.m and reg.m, respectively, of Henry Wolkowicz.

Compared to lpsimplex, wsimplex is much slower (4.5 times slower in an example with \( \text{dim}(x) = 200, m1 = 1 \) and \( m2 = 92 \)). Contrary to lpsimplex, wsimplex is written completely in MATLAB language, and is thus simply portable between computers.

See also
wlpsmplx, lpsimplx