Parameter Estimation - Computer Laboratory 3

Table of Contents

Dynamic models of discrete time systems

A general discrete time linear time invariant dynamic model can be written in an input-output model form using the backward difference method:

$$y(k) + a_1 y(k-1) + \ldots + a_n y(k-n) = b_0 u(k-d) + \ldots + b_m u(k-d-m)$$

where

- y is the output
- uu is the input
- k is the discrete time instance
- $a_1, \ldots, a_n, b_0, \ldots, b_m$ are the parameters of the model
- $d = n - m$ is the time delay between the input and the output. (d>0)

The above model is usually written in a compact form

$$A^*(q^{-1})y(k) = B^*(q^{-1})u(k-d)$$

where q^{-1} is the time shift operator.

Different types of models can be distinguished based on the used terms.

The general from is the ARMAX process (AutoRegressive Moving Average with Exogeneous inputs):

$$A^*(q^{-1})y(k) = B^*(q^{-1})u(k) + C^*(q^{-1})e(k)$$

where

- $A^*(q^{-1}) = 1 + a_1 q^{-1} + \ldots + a_n q^{-n}$ (AR part)
- $B^*(q^{-1}) = b_0 q^{-d} + b_1 q^{-d-1} + \ldots + b_m q^{-d-m}$ (X part)
- $C^*(q^{-1}) = c_0 + c_1 q^{-1} + \ldots + c_n q^{-n}$ (MA part)

- $e(k)$ is a white noise process

In this course we focus on the ARX models, where the moving average part is missing, i.e. only white noise is present:

$$A^*(q^{-1})y(k) = B^*(q^{-1})u(k) + e(k)$$

Creating input-output models in MATLAB (System Identification Toolbox)

In MATLAB the System Identification Toolbox provides tools to estimate parameters of different types of dynamic models.

The `idpoly` function can be used to create different types of polynomial models, given in the following form:

$$A(q^{-1})y(k) = \frac{B(q^{-1})}{F(q^{-1})}u(k) + \frac{C(q^{-1})}{D(q^{-1})}e(k)$$

with the polynomials

- $A(q^{-1}) = 1 + a_1q^{-1} + \ldots + a_nq^{-n}$
- $B(q^{-1}) = b_0 + b_1q^{-1} + \ldots + b_mq^{-m}$
- $C(q^{-1}) = 1 + c_1q^{-1} + \ldots + c_lq^{-l}$

The `idpoly` finction creates an `idpoly` object, which represent the specified model.

The syntax of the function is the following

- `sys=idpoly(A,B,C,D,F,NoiseVariance,Ts)` creates a polynomial model with identifiable coefficients. `A`, `B`, `C`, `D`, and `F` specify the initial values of the coefficients. `NoiseVariance` specifies the initial value of the variance of the white noise source. `Ts` is the model sample time.
- The polynomials A,B,C,D,F can be defined with their coefficients.
- Default values: `B=[]; C=1; D=1; F=[]; NoiseVariance=1` or `eye(n); Ts=1;`

Example:

Consider the following I/O model:

$$y(k) - 0.2y(k-1) + 0.5y(k-2) = u(k-1) + 0.8u(k-2) + e(k) - 0.3e(k-1)$$

- What kind of process is $y(k)$? - ARMAX
- What are the coefficients of A, B, C?

$$A(q^{-1}) = 1 - 0.2q^{-1} + 0.5q^{-2}$$

$$B(q^{-1}) = 0 + q^{-1} + 0.8q^{-2}$$

$$C(q^{-1}) = 1 - 0.3q^{-1}$$

- Create the model in MATLAB:

```
A=[1 -0.2 0.5];
B=[0 1 0.8];
C=[1 -0.3];
sys=idpoly(A,B,C)
```

```
sys =
Discrete-time ARMAX model: A(z)y(t) = B(z)u(t) + C(z)e(t)
  A(z) = 1 - 0.2 z^-1 + 0.5 z^-2

  B(z) = z^-1 + 0.8 z^-2

  C(z) = 1 - 0.3 z^-1

Sample time: unspecified

Parameterization:
   Polynomial orders:   na=2   nb=2   nc=1   nk=1
   Number of free coefficients: 5
   Use "polydata", "getpvec", "getcov" for parameters and their uncertainties.

Status:
Created by direct construction or transformation. Not estimated.
```

```
na=sys.na
A=sys.A
a2=sys.A(2)
```

Look at the solution:

`z^-1` is equivalent to `q^-1`, you can change the variable name with the `idpoly(A,B,C,'Variable','q-1')` option.

The polynomial orders have the following meaning:

- na is the order of $A(q^{-1})$. It can be accessed directly by `sys.na`
- nb is the order of $B(q^{-1}) + 1$ (`sys.nb`)
- nc is the order of $C(q^{-1})$ (`sys.nc`)
- The A, B, C etc. polynomials can be accessed by `sys.A`, `sys.B`, `sys.C`, etc. They return the coefficient vector of A, B, C etc. respectively.

nk is the delay between the input and the output = the difference between the greatest exponent of $A(q^{-1})$ and $B(q^{-1})$ = number of zeros at the beginning of B (`sys.nk`)

Individual task

- Create the following I/O model in MATLAB (`sys2`):

$$y(k) - 1.5y(k-1) + 0.7y(k-2) = u(k-1) + 0.5u(k-2) + e(k)$$

Simulating input-output models in MATLAB

The created input-output models can be simulated with a given input. The syntay of the simulation is the following

```
y=sim(sys,udata)
```

where

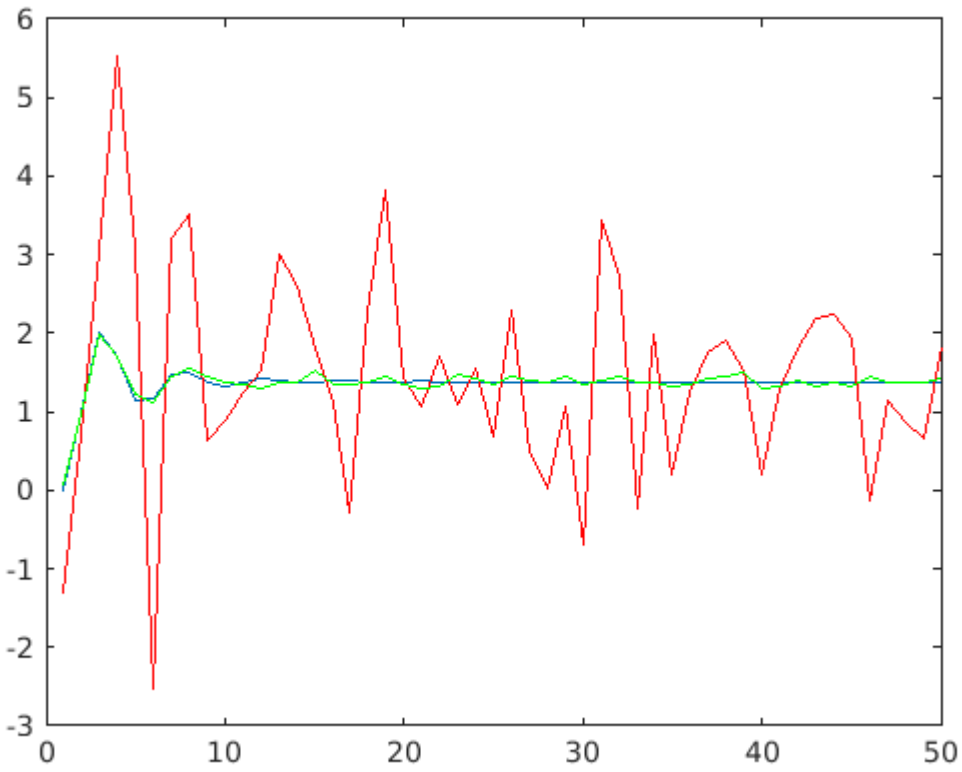
`sys:` is the system model given in various forms, in our case an `idpoly` object

`udata:` is the simulation input data, specified as an `iddata` object or a matrix.

`opt:` additional options. eg. initial conditions, additive noise

`y:` is the simulated response of the model

```
u=ones(50,1);
y=sim(sys,u);
plot(y)
hold on
opt=simOptions('AddNoise',true); %adding gaussian white noise to the data
y_noise=sim(sys,u,opt);
plot(y_noise,'r');
e=0.05*randn(length(u),1);
opt2=simOptions('AddNoise',true,'NoiseData',e); %adding custom noise data
y_noise2=sim(sys,u,opt2);
plot(y_noise2,'g');
hold off
```



Creating input data

The simulation input data should be specified either as an `iddata` object or a numeric matrix.

The `iddata` object contains measurement dat of the inputs and the outputs. The data can be in the time or frequency domain.

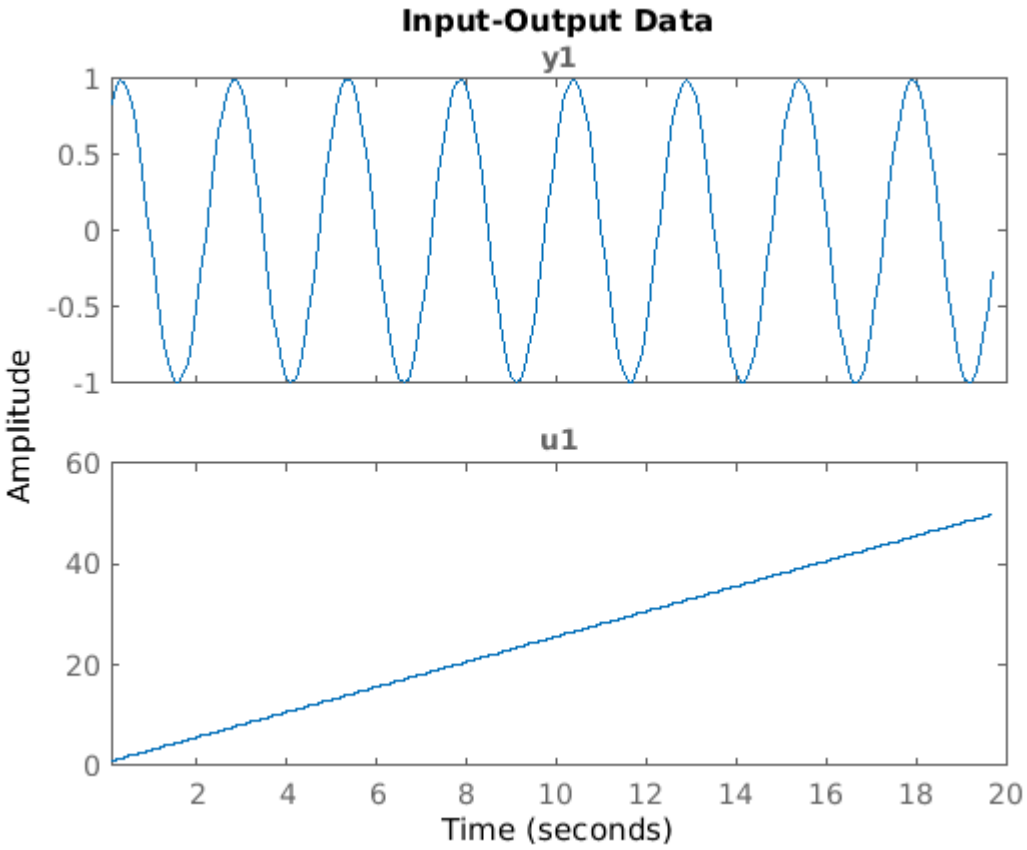
- in the time domain, the data can be uniformly or nonuniformly sampled. To use the `iddata` object for estimation, however, the data must be uniformly sampled, and the input and output data for each experiment must be recorded at the same time instants.
- You can specify data properties, such as the sample time, start time, time points, frequency sample points, and intersample behavior.
- `data=iddata(y,u,Ts)` creates an `iddata` object containing a time-domain output signal `y` and input signal `u`. `Ts` specifies the sample time of the experimental data.

```
u2=1:0.25:50;
y2=sin(u2);
Ts=0.1;
data=iddata(y2',u2',Ts)
```

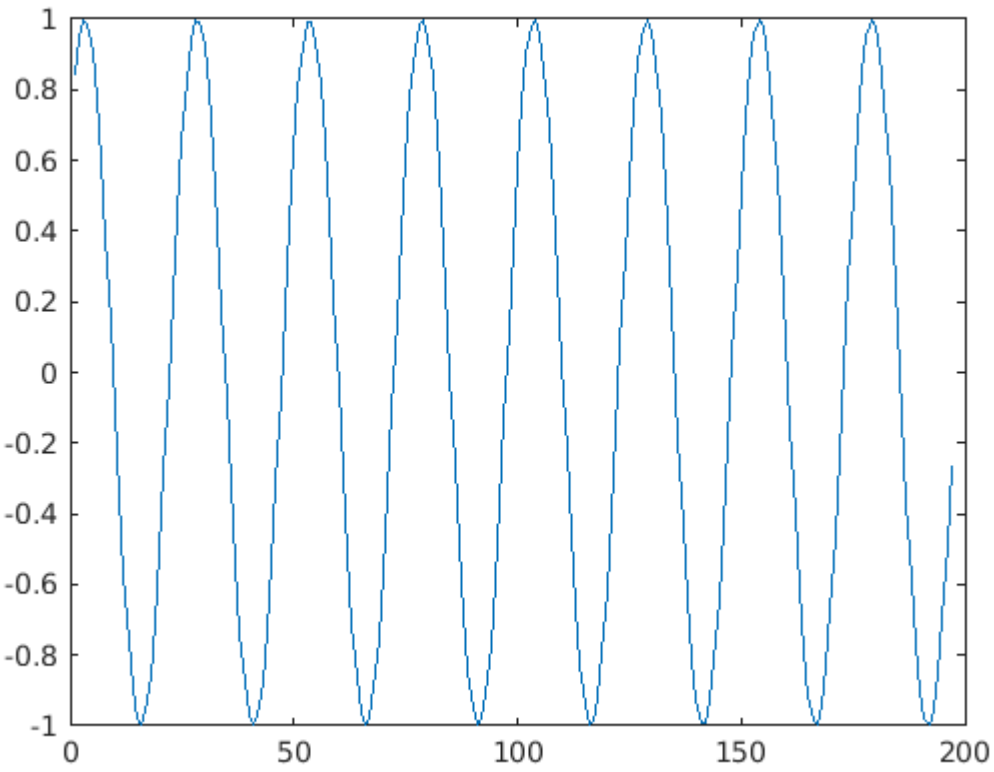
```
data =

Time domain data set with 197 samples.
Sample time: 0.1 seconds

Outputs      Unit (if specified)
   y1
```

```
Inputs       Unit (if specified)
   u1
```

```
plot(data)
```




```
plot(data.y)
```

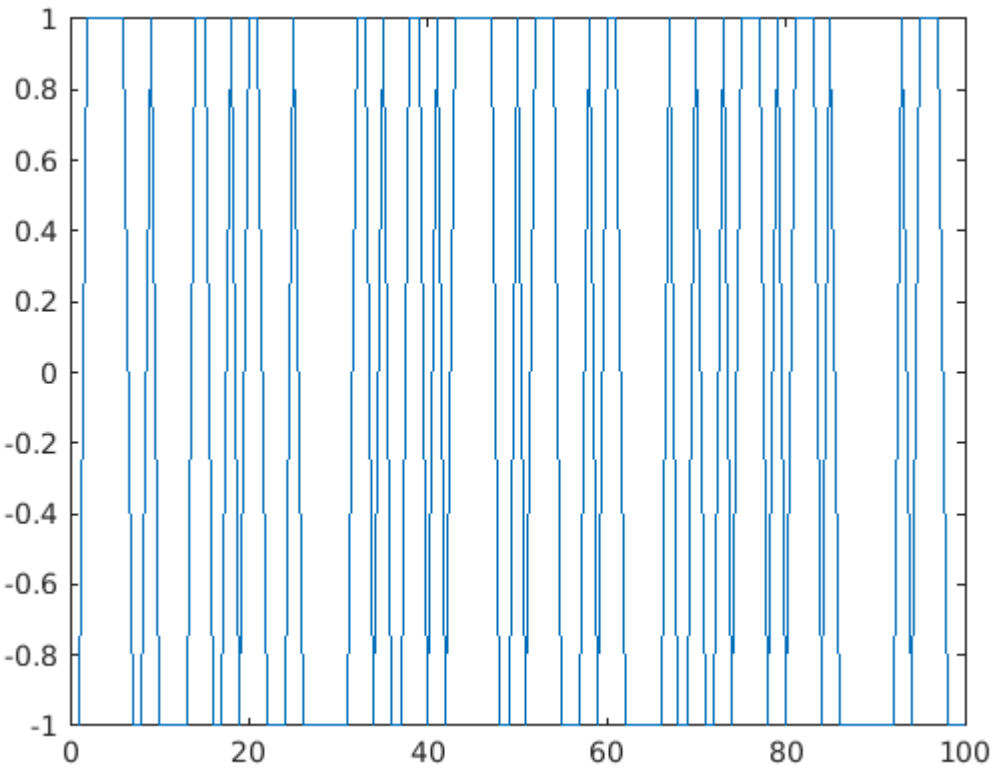


Special input signals can be generated using the `idinput` functione.g. random binary signal (RBS), random gaussian signal, pseudo random binary signal (PRBS), sum of sinusoidal signals

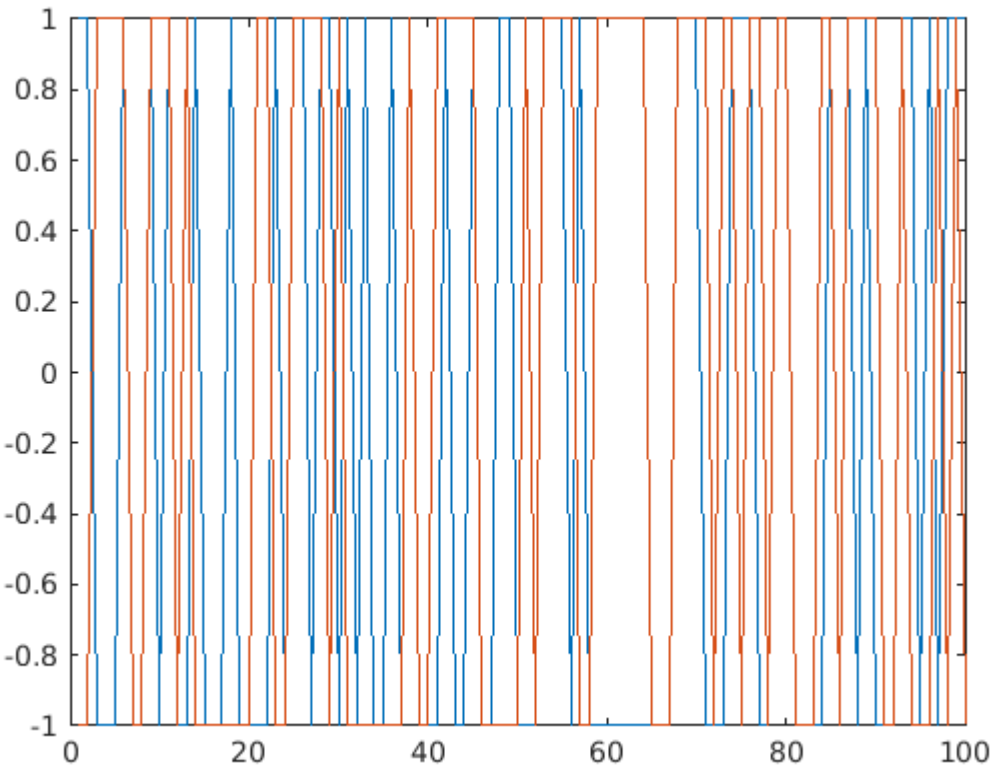
- `idinput(N)` generates a one channel random binary signal with `N` elements. The values can be 1 or -1.
- `idinput([N,Nu])`: returns an `Nu`-channel random binary input signal, where each channel signal has length `N`. The signals in each channel differ from each other.
- `u=idinput(___,Type)` specifies the type of input to be generated as one of the following:
- `'rbs'` — Random binary signal
- `'rgs'` — Random Gaussian signal
- `'prbs'` — Pseudorandom binary signal
- `'sine'` — Sum-of-sinusoids signal
- `u=idinput(___,Type,Band)` specifies the frequency band of the signal. For pseudorandom binary signals (PRBS), `Band` specifies the inverse of the clock period of the signal. In case of PRBS the band should be `[0 B]`.
- `u=idinput(___,Type,Band,Range)` specifies the amplitude-range of the signal.

Examples:

```
u3=idinput(100);
plot(u3)
```



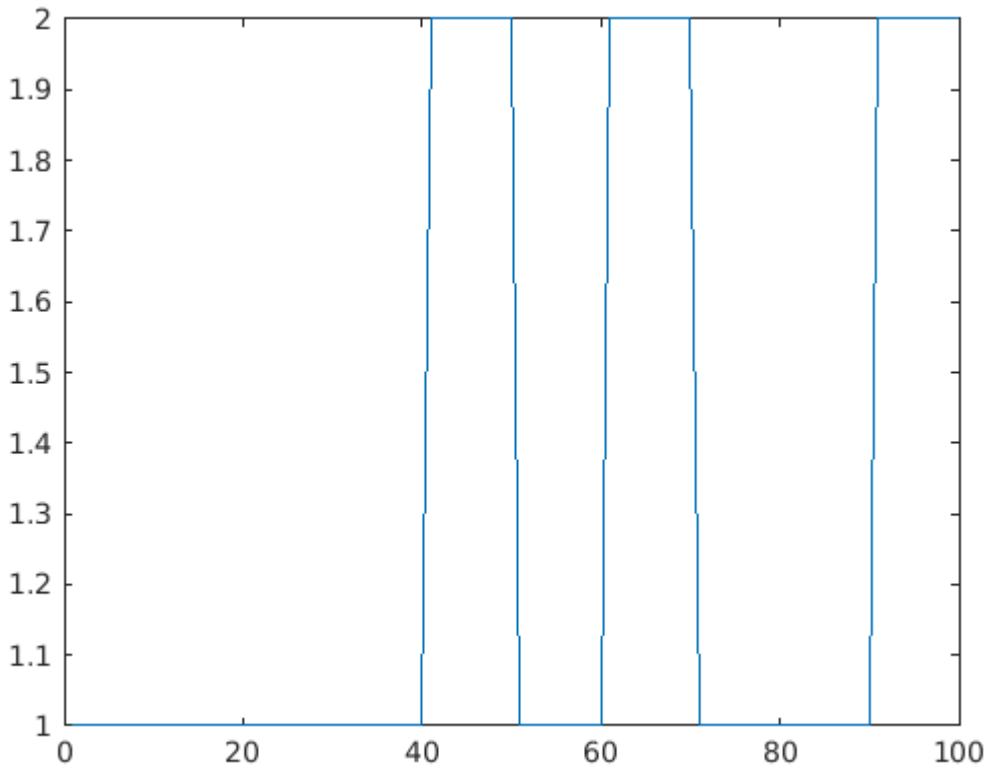

```
u4=idinput([100,2]);
plot(u4)
```




```
u5=idinput(100,'prbs',[0 0.1],[1 2]);
```

```
Warning: The PRBS signal delivered is the 100 first values of a full sequence of length 150.
```

```
plot(u5)
```



Individual task

1. Create a random gaussian input signal (u6) with 500 elements.
2. Simulate the previously created model (`sys2`) with this input and additive noise with 0.2 standart deviation and save the response in the variable `y6`.
3. Create an `iddata` object (`data6`) from the measured input-output data.

Least squares estimation of ARX models

The ARX inout-output models can be written in predictive form:

$$\hat{y}(k|p) = -a_1 y(k-1) - \ldots - a_n y(k-n) + b_0 u(k-d) + \ldots + b_m u(k-d-m) + e(k) =$$
$$= p^T \varphi(k-1) + e(k)$$

where

- $p = [-a_1 \ldots -a_n \, b_0 \ldots b_m]^T$ is the parameter vector
- $\varphi(k-1) = [y(k-1) \ldots y(k-n) \; u(k-d) \ldots u(k-d-m)]^T$ is the regressor

The solution of the LS estimation problem is

$$\hat{p} = \left[\frac{1}{N}\sum_{k=1}^{N} \varphi(k)\varphi^T(k)\right]^{-1} \frac{1}{N}\sum_{k=1}^{N} \varphi(k) y(k)$$

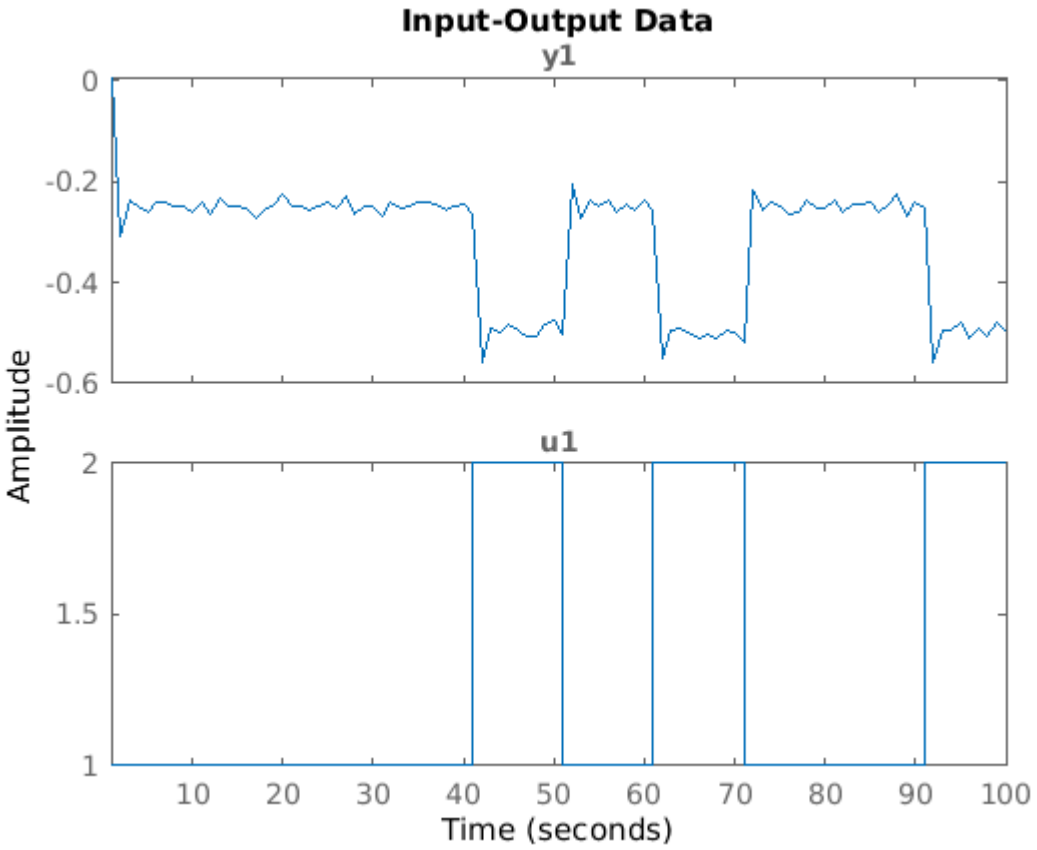
Parameter estimation of ARX models in MATLAB

The `arx` function in the System Identification Toolbox can be used to directly estimate the parameters of an ARX model with available input-output measurement data. In order to estimate the parameters, the model structure need to be defined, namely the orders of the A and B polynomials and the time delay between the input and the output.

- `sys=arx(data,[na nb nk])` estimates the parameter of an ARX model using the measurement data in `data`. The result is an `idpoly` object with the identified parameters.
- `getpvec(sys)` get the parameter vector of the identified model `sys`. Note: getpvec returns the parameters $a_1, \ldots, a_n$ instead of $-a_1, \ldots, -a_n$.
- `getcov(sys)` get the covariance matrix of the estimated parameters of `sys`.

Example:

```
sys3=idpoly([1 0.2],[0 -0.3]);
opt3=simOptions('AddNoise',true,'NoiseData',0.01*randn(size(u5)));
y3=sim(sys3,u5,opt3);
data3=iddata(y3,u5);
plot(data3)
```




```
sys3_est=arx(data3,[1 1 1])
```

```
sys3_est =
Discrete-time ARX model: A(z)y(t) = B(z)u(t) + e(t)
```

```
  A(z) = 1 + 0.202 z^-1

  B(z) = -0.2999 z^-1

Sample time: 1 seconds

Parameterization:
   Polynomial orders:   na=1   nb=1   nk=1
   Number of free coefficients: 2
   Use "polydata", "getpvec", "getcov" for parameters and their uncertainties.

Status:
Estimated using ARX on time domain data "data3".
Fit to estimation data: 91.56% (prediction focus)
FPE: 0.0001097, MSE: 0.0001033
```

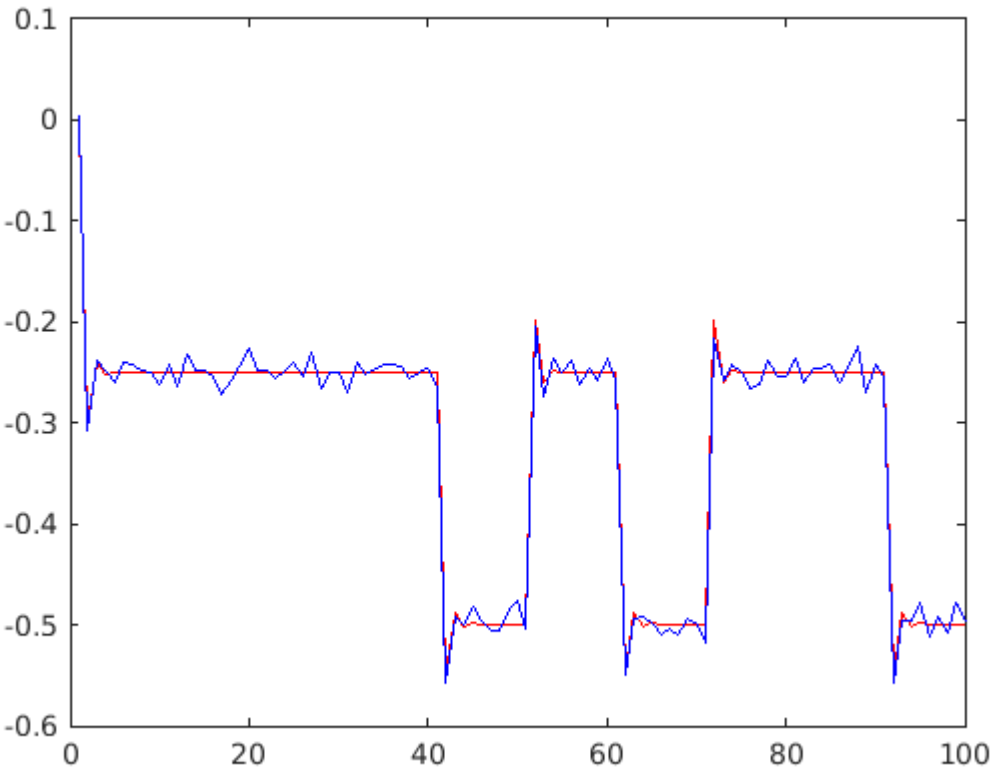
```
p3_est=getpvec(sys3_est)
```

```
p3_est = 2×1
    0.2020
   -0.2999
```

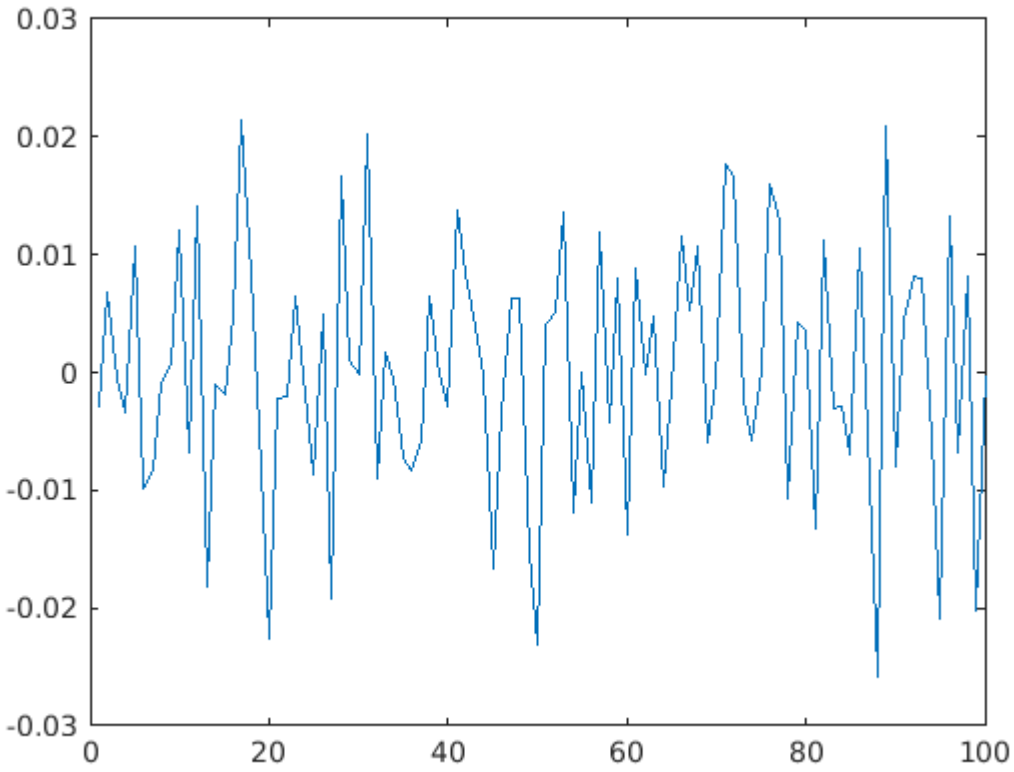
```
cov3=getcov(sys3_est)
```

```
cov3 = 2×2
10^-3 ×
    0.2708   -0.0661
   -0.0661    0.0167
```

```
y3_est=sim(sys3_est,u5);    %simulate response of the identified model
plot(y3_est,'r');
hold on
plot(y3,'b')
```




```
figure
plot(y3_est-y3) % residual sequence
```



Evaluate the quality of the estimated parameters:

- residual sequence should be a white noise process
- covariances should be 'small"

Individual task 1

1. Estimate the parameters of the previously created model `sys2` using the measurement data in `data6`.
2. Plot the response of the identified model and compute the residuals.
3. Evaluate the parameter estimation results (covariance, residuals).

Individual task 2

Let us assume that we have the following ARX model:

$$y(k) + a_1 y(k-1) - a_2 y(k-2) = b_1 u(k-1) + b_2 u(k-2) + e(k)$$

1. Estimate the parameters of the following ARX model, using the data given in `D1` and `D2`.
2. Compare the results of the two estimation:

- Plot the response of the identified model and compute the residuals.
- Evaluate the parameter estimation results (covariance, residuals).

Asymptotic unbiasedness

The unbiasedness of the estimate can be interpreted in an asymptotic sense. It means that the bias becomes closer to zero if the number of samples goes to infinity.

Examine the asymptotic unbiasedness of the estimate of an ARX model.

Create a function (`asympt_LS`) that estimates the parameters of a known ARX system with different number of samples (e.g. 100,200,...,1000) and returns the estimated parameter vectors at each sample size and the mean of the estimated parameters.

1. The inputs of the function are the ARX model as an `idpoly` object (`sys`) and the number of samples (`N`).
2. The output of the function is a matrix (`P_est`), whose columns are the estimated parameter vectors, and the mean vector of the parameters (`m_P`).
3. In the function body, initialize `P_est` as an empty matrix.
4. Create a `for loop` from `i=1` to `N*1000`, with stepsize 100. We will create samples with size 100, 200,...N*1000.
5. In the function body, generate input signals for the ARX model: let's say, PRBS input with `i` samples and with clock period 10 and values between [-1 1]. (`idinput`)
6. Add normally distributed noise to the simulation with 0 mean and 0.01 standard deviation. (`SimOptions`)
7. Simulate the system to get the output data.
8. Create an `iddata` object from the input and output data.
9. Estimate the parameters of the model using the simulated input-output data. To get the model structure (values of `na`, `nb`, `nk`), use `sys.na`, `sys.nb`, `sys.nk`.
10. Append the new estimated parameter vector to the end of the `P_est` matrix.
11. Outside the loop, compute the mean of the `P_est` matrix by rows.

- Call the created function with the `sys2` or the `sys3` model and different sample sizes (e.g. N=5,10,20...)
- You can plot the sequence of estimated paramters. What can you experience when the sample size grows?
- What can we say about the mean of the estimated parameters?

Homework (Deadline 2020. November 4, 10:30)

Estimate the parameters of the ARX model, which is given in the following form:

$$y(k) = -a_1 y(k-1) - a_2 y(k-2) + b_1 u(k-1)$$

The measurement data can be found in the `D5` variable in the `arx_data.mat` file

- Give the estimated parameter vector.
- Give the covariance matrix of the estimation.
- Simulate the estimated model with the original input, and compute the residuals.
- Plot the measured output, the estimated output and the residual sequence.

Send the created script file (NEPTUNKOD-HW3.m) to pozna.anna@virt.uni-pannon.hu!

Solution for the asymptotic unbiasedness function

```
function [P_est,m_P]=asympt_LS(sys,N)
    P_est=[];
for i=100:100:N*1000
    u=idinput(i,'prbs',[0 0.1],[1 2]);
    opt=simOptions('AddNoise',true,'NoiseData',0.01*randn(size(u)));
    y=sim(sys,u,opt);
    data=iddata(y,u);
    na=sys.na;
    nb=sys.nb;
    nk=sys.nk;
    sys_est=arx(data,[na nb nk]);
    P_est=[P_est,getpvec(sys_est)];
end
m_P=mean(P_est,2);
end
```