Sumar

❷ Identificarea modelelor neparametrice

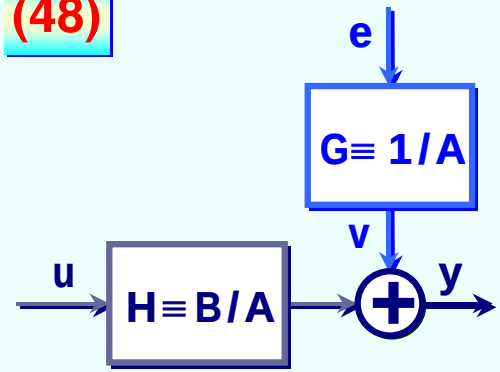
☞ Contextul de lucru

Două modele ARMAX

Auto-Regresiv cu Control eXogen

ARX[na,nb]: $A(q^{-1})y[n] = B(q^{-1})u[n] + e[n]$

(48)



Eroare de Ieşire (Output Error)

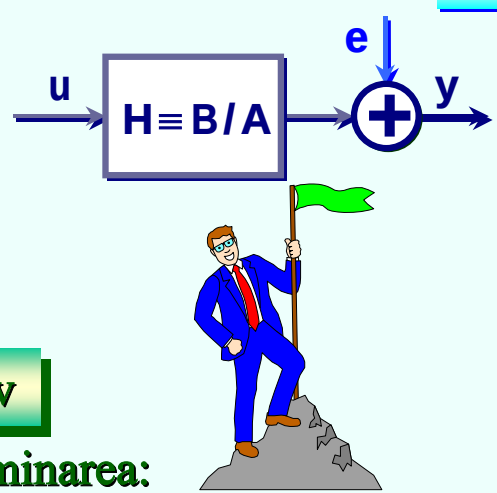
OE[na,nb]: $y[n] = \frac{B(q^{-1})}{A(q^{-1})}u[n] + e[n]$

(49)

$E\{e[n]\} = 0$

$E\{e[n]e[n \pm k]\} = \lambda^2 \delta_0[k], \ \forall k \in \mathbb{Z}$

(zgomot alb)



Cazuri particulare

Ordin I

$A(q^{-1}) = 1 - 0.8q^{-1}$

$B(q^{-1}) = q^{-1}$

(50)

Ordin II

$A(q^{-1}) = 1 - 0.1q^{-1} - 0.56q^{-2}$

$B(q^{-1}) = 0.5q^{-1} + 0.3q^{-2}$

(51)

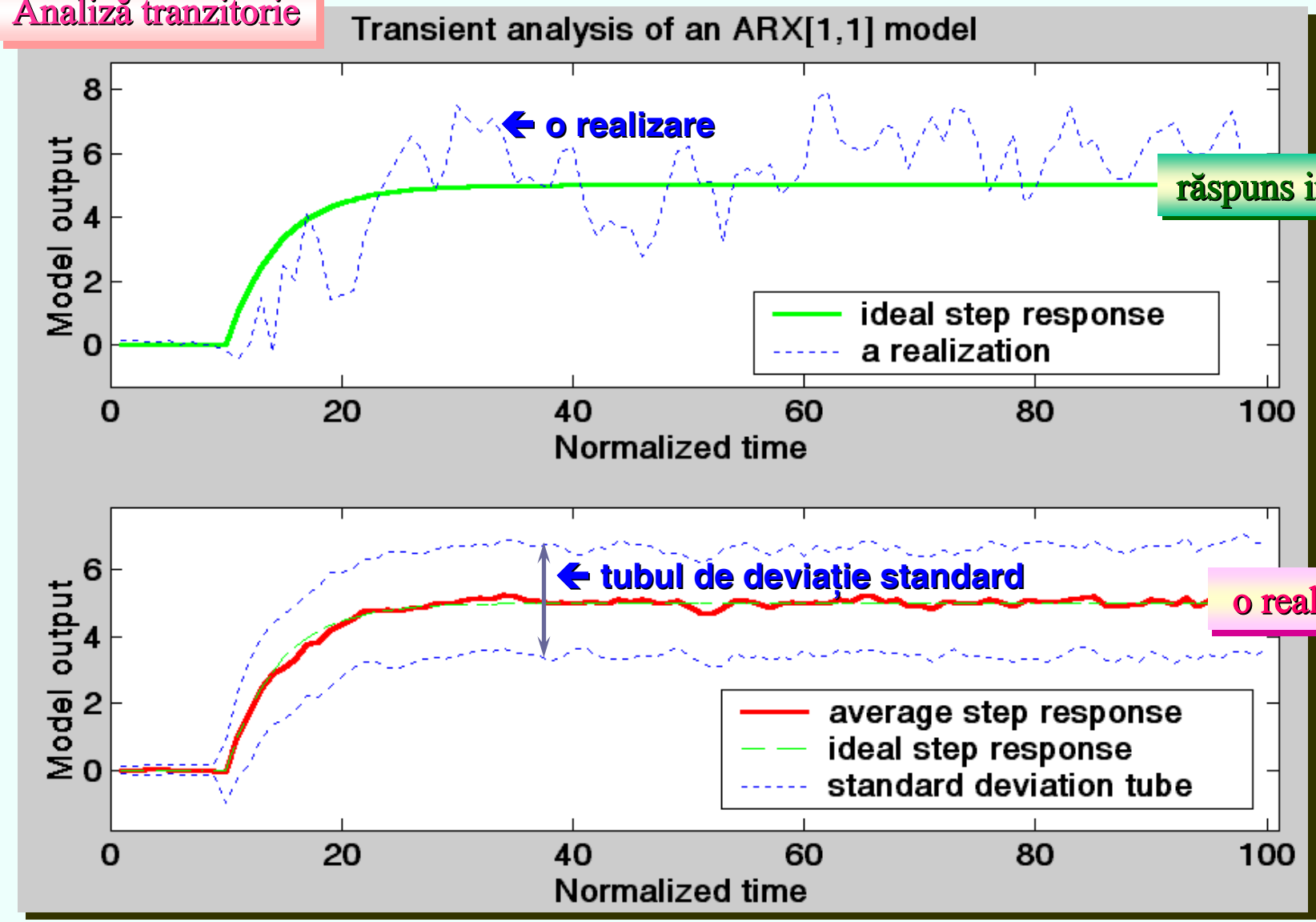
Obiectiv

- Determinarea:
  - ➔ răspunsului indicial
  - ➔ răspunsului la impuls
  - ➔ răspunsului în frecvență

❷ Identificarea modelelor neparametrice

☞ Exemple (model ARX de ordin I)

Analiză tranzitorie

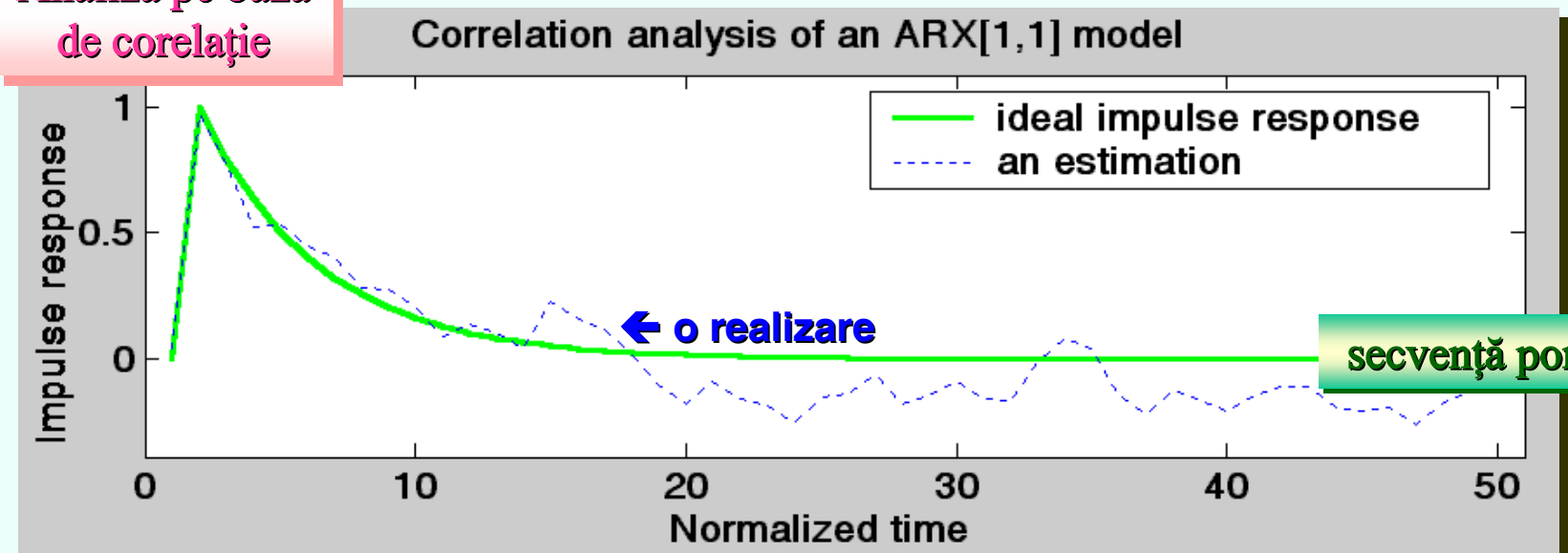


☝ Operaţia de mediere statistică peste un număr de realizări este esenţială.

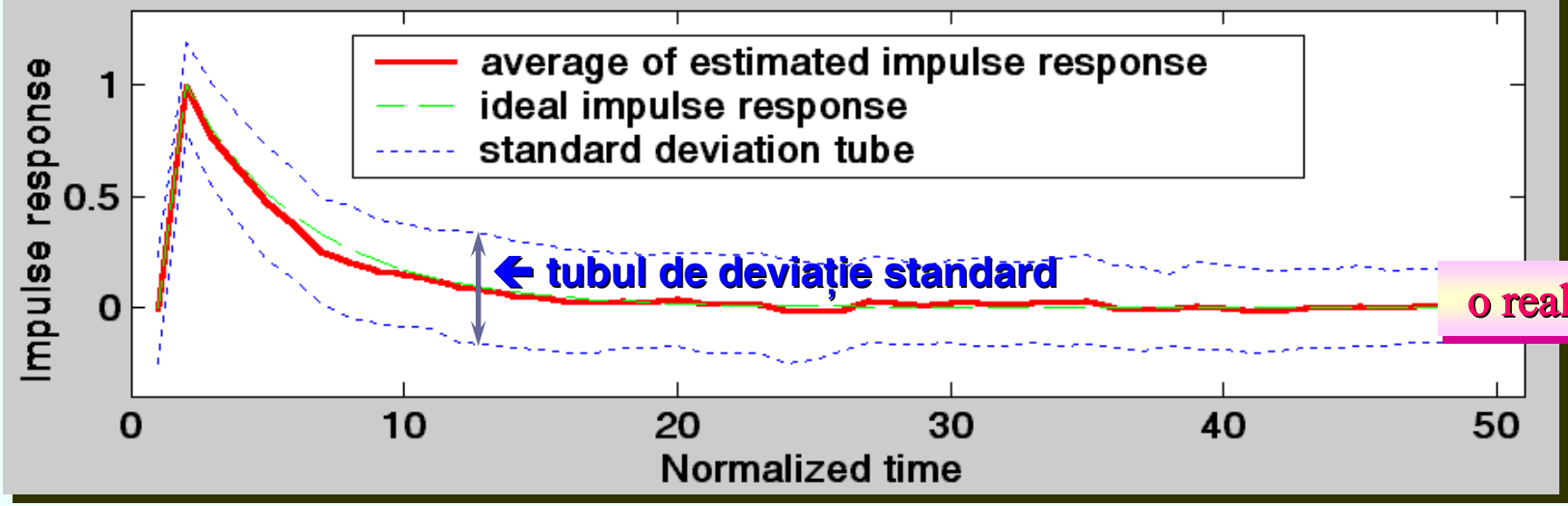
❷ Identificarea modelelor neparametrice

☞ Exemple (model ARX de ordin I)

Analiză pe bază de corelaţie



secvenţă pondere ideală



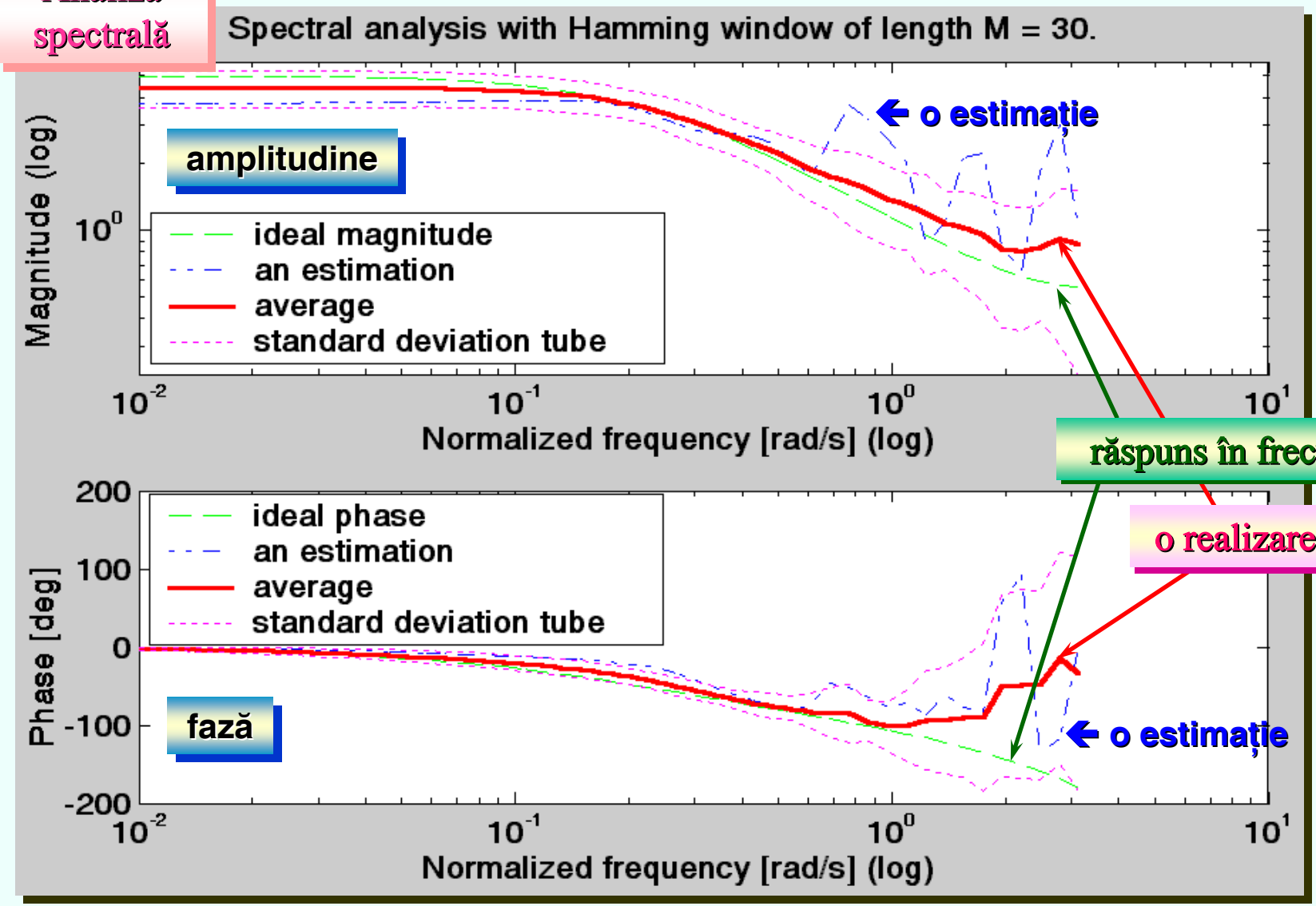
o realizare medie

☝ Ecuaţia Wiener-Hopf este esenţială.

❷ Identificarea modelelor neparametrice

☞ Exemple (model ARX de ordin I)

Analiză spectrală



☝ Ecuația transferului densității spectrale prin sisteme liniare este esențială.

❷ Identificarea modelelor neparametrice

☞ Contextul de lucru

Matrice de auto-covarianţă

$$\mathbf{R}_M(u) \overset{def}{=} \begin{bmatrix} r_u[0] & r_u[1] & r_u[2] & \cdots & r_u[M-1] \\ r_u[1] & r_u[0] & r_u[1] & \cdots & r_u[M-2] \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ r_u[M-2] & \cdots & r_u[1] & r_u[0] & r_u[1] \\ r_u[M-1] & \cdots & r_u[2] & r_u[1] & r_u[0] \end{bmatrix} \geq 0$$

Matrice Toeplitz simetrică

Matrice al cărei element generic depinde numai de modulul diferenţei indicilor săi.

Matricea poate fi construită folosind numai o linie sau o coloană.

$$r_u[k] \overset{def}{=} E\{u[n]u[n-k]\}$$

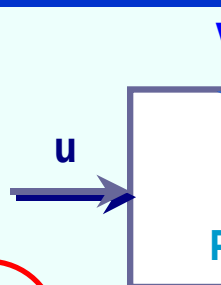
Ipoteza Ergodică

$$r_u[k] \cong \frac{1}{N-k}\sum_{n=k+1}^{N} u[n]u[n-k]$$

$\forall k \in 1 \ldots \lfloor N/4 \rfloor$

☝ pentru a asigura o precizie suficient de mare

Ecuaţia Wiener-Hopf



vectorul parametrilor necunoscuţi

$$\boldsymbol{\theta}^T \overset{def}{=} \begin{bmatrix} h[0] & h[1] & \cdots & h[M-1] \end{bmatrix}$$

Cum s-ar putea determina θ din date I/O măsurate?

E ② $\times u[n-k]$ ①

$y \equiv h * u + v$

$\mathbf{R}_M(u) > 0$

☝ persistenţă

$E\{v[n]u[n-k]\} \equiv 0$

$\mathbf{R}_M(u)\boldsymbol{\theta} = \mathbf{r}_M(y,u)$

$\boldsymbol{\theta} = \mathbf{R}_M^{-1}(u)\mathbf{r}_M(y,u)$

vectorul covarianţelor încrucişate

❷ Identificarea modelelor ne-parametrice

☞ Contextul de lucru

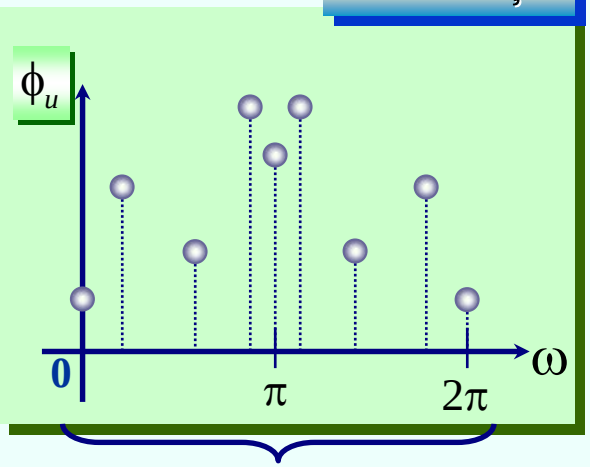
Persistenţă (de ordin M)

Timp

$$\mathbf{R}_M(u) \overset{def}{=} \begin{bmatrix} r_u[0] & r_u[1] & r_u[2] & \cdots & r_u[M-1] \\ r_u[1] & r_u[0] & r_u[1] & \cdots & r_u[M-2] \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ r_u[M-2] & \cdots & r_u[1] & r_u[0] & r_u[1] \\ r_u[M-1] & \cdots & r_u[2] & r_u[1] & r_u[0] \end{bmatrix} > 0$$

☝ inversabilă

Frecvenţă

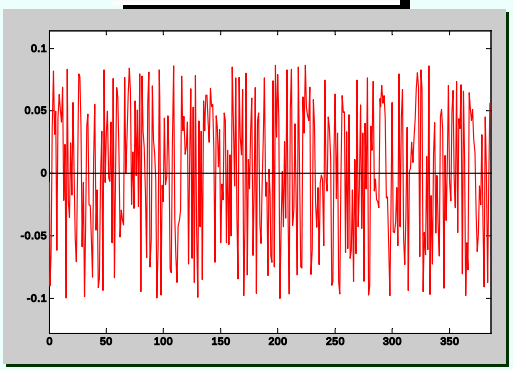


- Se pot determina primele M valori ale secvenţei pondere.

☝ cel puţin M linii spectrale nenule

Semnalul persistent ideal ($M = \infty$)

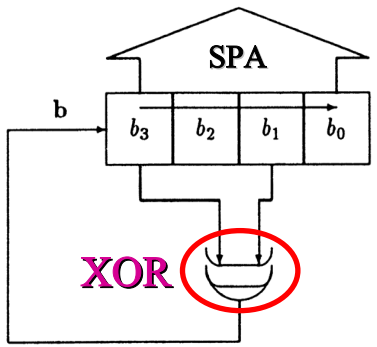
Zgomotul alb



☝ Imposibil de generat.

Semnalul persistent practic

Semnal Pseudo-Aleator (Binar) - SPA(B)



SPAB

b		b_3	b_2	b_1	b_0		SPA
1	↘	0	1	1	1	→	7
0	↘	1	0	1	1	→	11
0	↘	0	1	0	1	→	5
1	↘	0	0	1	0	→	2
1	↘	1	0	0	1	→	9
1	↘	1	1	0	0	→	12
0	↘	1	1	1	0	→	14
1	↘	0	1	1	1	→	7

Ordin de persistenţă egal cu perioada.

❷ Identificarea modelelor neparametrice

☞ Probleme de simulare

9p

Problema 2.1 (analiză tranzitorie) 1p

În cadrul acestei probleme, se va studia analiza tranzitorie. Modelele ARX şi OE vor fi simulate de 100 de ori cu intrarea treaptă:

$$u[n]=\begin{cases}0 & ,\ n\le 9\\ 1 & ,\ n\in\overline{10,100}\end{cases}$$

(timp de cel cel mult 100 de perioade de eşantionare).

0.1 p Să se reprezinte grafic, într-o primă fereastră, răspunsul indicial ideal al modelului ARX de ordin I (adică în absența zgomotului) plus prima realizare a ieşirii. Într-o a doua fereastră, să se traseze media răspunsurilor obținute (în prezența zgomotului), împreună cu răspunsul indicial ideal şi tubul de amplitudine a ieşirii oferit de deviația standard. În acest scop, se vor folosi funcțiile MATLAB: **`filter`**, **`mean`** şi **`std`**. Observați că deviația standard trebuie calculată luînd în considerare ansamblul statistic al realizărilor şi nu media acestor realizări. Ce rol credeți ca are tubul de deviație standard astfel ilustrat? Denumiți mini-simulatorul pe care l-ați proiectat prin **`ISLAB_2A`**.

0.1 p Studiați convergența ieşirii la răspunsul indicial ideal variind numărul de realizări ale procesului ARX în diferite rulări ale mini-simulatorului **`ISLAB_2A`**. Este aparent verificată ipoteza ergodică? (Oferiți toate explicațiile necesare.)

0.1 p Imaginați o tehnică de estimare pe cale grafică a celor 2 parametri a şi b ai modelului ARX de ordin I, folosind realizările ieşirii (mai precis zona tranzitorie a acestora).

0.3 p Reluați simulările pentru modelul OE (proiectați mini-simulatorul **`ISLAB_2B`**) şi comparați rezultatele cu cele ale simulatorului precedent.

0.4 p Generalizați mini-simulatoarele **`ISLAB_2A`** şi **`ISLAB_2B`** pentru cazul unui model ARMAX[na,nb,nc] (adică proiectați mini-simulatorul general **`ISLAB_2C`**) utilizînd aceeaşi intrare treaptă de mai sus şi un zgomot alb de dispersie unitară. Rulați simulatorul în cazul modelelor ARX şi OE de ordin II. Comparați rezultatele celor 2 modele. Pot fi determinați parametrii unui sistem de ordin II (amplificare, supra-reglaj, pulsație de rezonanță), afectat de zgomot, prin analiză tranzitorie, ca în cazul sistemelor de ordin I? Dacă nu, argumentați de ce. Dacă da, explicați în ce constă tehnica de identificare.

Program existent pe CD

`ISLAB_2A`

Programe ce trebuie proiectate

`ISLAB_2B`

`ISLAB_2C`

Înainte de a rula mini-simulatoarele existente, trebuie executate comenzile:

```
> global FIG
> FIG = 1 ;
```

❷ Identificarea modelelor neparametrice

Rutine MATLAB (Problema 2.1)

```
> help function
> helpwin
> doc function
```

filter

Filter data with an infinite impulse response (IIR) or finite impulse response (FIR) filter

Syntax

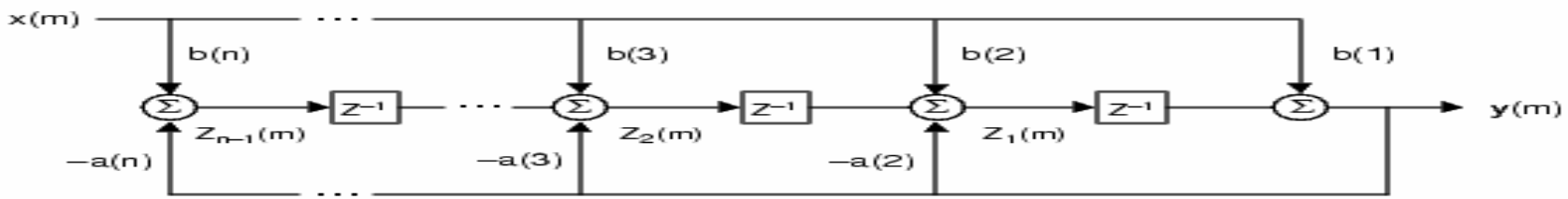
```
y = filter(b,a,X)
[y,zf] = filter(b,a,X)
[y,zf] = filter(b,a,X,zi)
y = filter(b,a,X,zi,dim)
[...] = filter(b,a,X,[],dim)
```

Description

The `filter` function filters a data sequence using a digital filter which works for both real and complex inputs. The filter is a *direct form II transposed* implementation of the standard difference equation (see "Algorithm").

Algorithm

The `filter` function is implemented as a direct form II transposed structure.



or

```
y(n) = b(1)*x(n) + b(2)*x(n-1) + ... + b(nb+1)*x(n-nb)
                 - a(2)*y(n-1) - ... - a(na+1)*y(n-na)
```

where `n-1` is the filter order, and which handles both FIR and IIR filters [1].

$$Y(z) = \frac{b(1) + b(2)z^{-1} + \ldots + b(nb+1)z^{-nb}}{1 + a(2)z^{-1} + \ldots + a(na+1)z^{-na}} X(z)$$

Domeniul Complex — z^{-1}

Domeniul Timp — q^{-1}

❷ Identificarea modelelor neparametrice

Rutine MATLAB (Problema 2.1)

mean

Average or mean value of arrays

Syntax

- ```
  M = mean(A)
  M = mean(A,dim)
  ```

Description

`M = mean(A)` returns the mean values of the elements along different dimensions of an array.

If `A` is a vector, `mean(A)` returns the mean value of `A`.

If `A` is a matrix, `mean(A)` treats the columns of `A` as vectors, returning a row vector of mean values.

If `A` is a multidimensional array, `mean(A)` treats the values along the first non-singleton dimension as vectors, returning an array of mean values.

`M = mean(A,dim)` returns the mean values for elements along the dimension of `A` specified by scalar `dim`.
  ```

Rutine MATLAB (Problema 2.1)

std

Standard deviation

Syntax

```
s = std(X)
s = std(X,flag)
s = std(X,flag,dim)
```

Definition

There are two common textbook definitions for the standard deviation s of a data vector X.

$$(1)\quad s = \left(\frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2 \right)^{\frac{1}{2}}$$

$$(2)\quad s = \left(\frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2 \right)^{\frac{1}{2}}$$

where

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

and n is the number of elements in the sample. The two forms of the equation differ only in $n-1$ versus n in the divisor.

Description

`s = std(X)`, where `X` is a vector, returns the standard deviation using (1) above. If `X` is a random sample of data from a normal distribution, s^2 is the best *unbiased* estimate of its variance.

If `X` is a matrix, `std(X)` returns a row vector containing the standard deviation of the elements of each column of `X`. If `X` is a multidimensional array, `std(X)` is the standard deviation of th elements along the first nonsingleton dimension of `X`.

`s = std(X,flag)` for `flag = 0`, is the same as `std(X)`. For `flag = 1`, `std(X,1)` returns the standard deviation using (2) above, producing the second moment of the sample about its mean.

`s = std(X,flag,dim)` computes the standard deviations along the dimension of `X` specified by scalar `dim`.

❷ Identificarea modelelor neparametrice

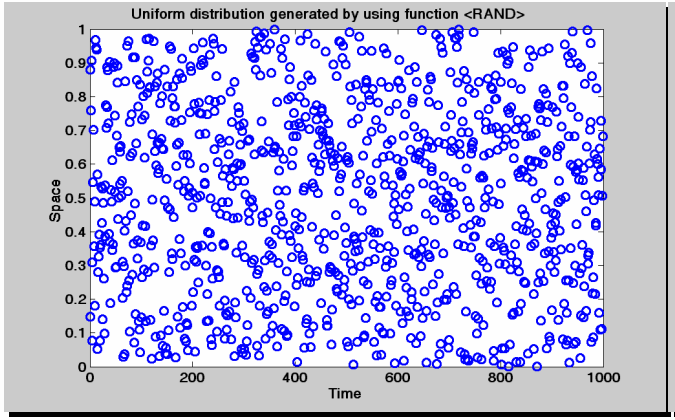
Rutine MATLAB (Problema 2.1)

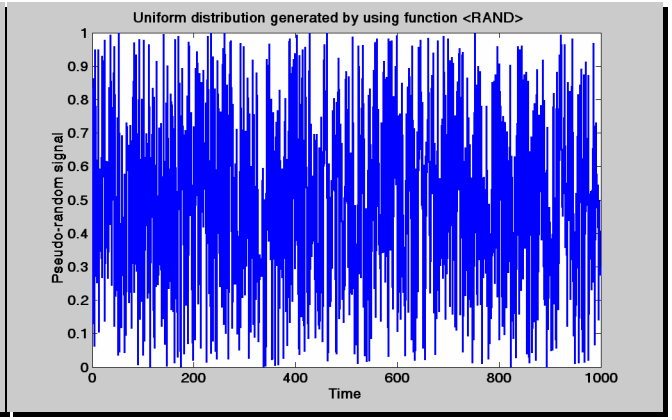
rand

Uniformly distributed random numbers and arrays

Syntax

```
Y = rand(n)
Y = rand(m,n)
Y = rand([m n])
Y = rand(m,n,p,...)
Y = rand([m n p...])
Y = rand(size(A))
rand
s = rand('state')
```





Description

The `rand` function generates arrays of random numbers whose elements are uniformly distributed in the interval (0,1).

`Y = rand(n)` returns an `n`-by-`n` matrix of random entries. An error message appears if `n` is not a scalar.

`Y = rand(m,n)` or `Y = rand([m n])` returns an `m`-by-`n` matrix of random entries.

`Y = rand(m,n,p,...)` or `Y = rand([m n p...])` generates random arrays.

`Y = rand(size(A))` returns an array of random entries that is the same size as `A`.

`rand`, by itself, returns a scalar whose value changes each time it's referenced.
`s = rand('state')` returns a 35-element vector containing the current state of the uniform generator. To change the state of the generator:

`rand('state',s)`	Resets the state to `s`.
`rand('state',0)`	Resets the generator to its initial state.
`rand('state',j)`	For integer `j`, resets the generator to its `j`-th state.
`rand('state',sum(100*clock))`	Resets it to a different state each time.

❷ Identificarea modelelor neparametrice

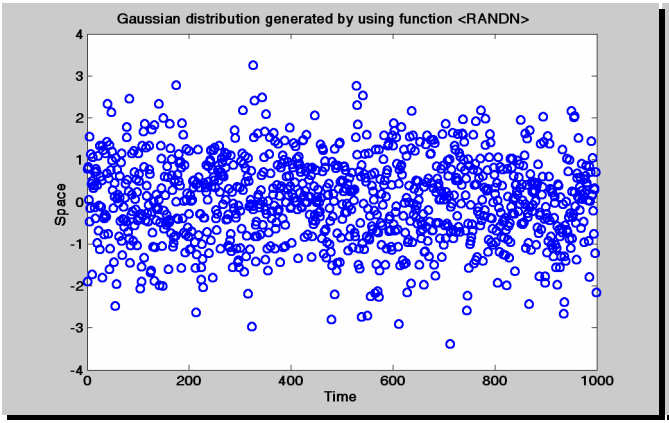
Rutine MATLAB (Problema 2.1)

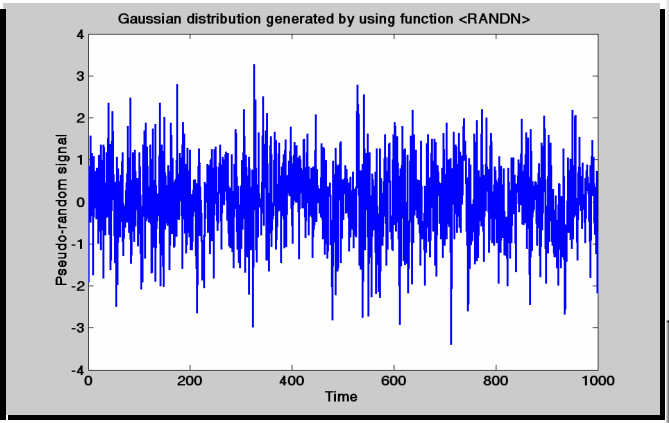
randn

Normally distributed random numbers and arrays

Syntax

```
Y = randn(n)
Y = randn(m,n)
Y = randn([m n])
Y = randn(m,n,p,...)
Y = randn([m n p...])
Y = randn(size(A))
randn
s = randn('state')
```





Description

The `randn` function generates arrays of random numbers whose elements are normally distributed with mean 0, variance $\sigma^2 = 1$, and standard deviation $\sigma = 1$.

`Y = randn(n)` returns an `n`-by-`n` matrix of random entries. An error message appears if `n` is not a scalar.

`Y = randn(m,n)` or `Y = randn([m n])` returns an `m`-by-`n` matrix of random entries.

`Y = randn(m,n,p,...)` or `Y = randn([m n p...])` generates random arrays.

`Y = randn(size(A))` returns an array of random entries that is the same size as `A`.

`randn`, by itself, returns a scalar whose value changes each time it's referenced.

`s = randn('state')` returns a 2-element vector containing the current state of the normal generator. To change the state of the generator:

`randn('state',s)`	Resets the state to `s`.
`randn('state',0)`	Resets the generator to its initial state.
`randn('state',j)`	For integer `j`, resets the generator to its `j`th state.
`randn('state',sum(100*clock))`	Resets it to a different state each time.

Problema 2.2 (analiză pe bază de corelație) 6p

Această problemă se referă la analiza pe bază de corelație. Nucleul acestui tip de analiză îl constituie ecuația Wiener-Hopf. Cu ajutorul acestei ecuații, se poate determina o mulțime finită de valori ale funcției pondere (răspunsul cauzal la impuls) asociate unui model de sistem cu ieşiri corupte de zgomot. Dorim să determinăm primele $M = 50$ de valori ale funcției pondere folosind cele 2 modele ARX şi OE. Acestea vor fi stimulate cu un SPAB bipolar de lungime $N = 100$. Valorile semnalului de intrare sunt doar –1 şi +1. Se vor efectua 100 de experimente.

0 p Să se reprezinte grafic, într-o primă fereastră, funcția pondere ideală a modelului ARX de ordin I (adică în absența zgomotului) plus funcția pondere estimată rezolvînd ecuația Wiener-Hopf, cu ajutorul datelor de intrare-ieşire corespunzătoare primei realizări a procesului. (Folosiți **Exercițiul 2.2** pentru a implementa ecuația generală a funcției pondere ideale.) Într-o a doua fereastră, să se traseze media estimațiilor obținute (în prezența zgomotului), împreună cu secvența pondere ideală şi tubul de deviație standard din jurul mediei Şi în acest caz se vor folosi funcțiile MATLAB: **`filter`**, **`mean`** şi **`std`**. Denumiți mini-simulatorul pe care l-ați proiectat prin **`ISLAB_2D`**.

0.3 p Reluați simulările pentru modelul OE de ordin I (proiectați mini-simulatorul **`ISLAB_2E`**) şi comparați rezultatele cu cele ale simulatorului precedent.

0.7 p Modificați mini-simulatoarele **`ISLAB_2D`** şi **`ISLAB_2E`** astfel încît intrarea de stimul să fie egală cu:

$$u_f[n] \overset{def}{=} \frac{u_0}{1-0.8q^{-1}} u[n], \quad \forall n \in \mathbb{N},$$

unde u_0 este un factor de normare egal cu 0.6, menit să egaleze variațele lui u (semnalul SPAB original) şi u_f (versiunea filtrată a semnalului SPAB). Denumiți noile mini-simulatoare prin **`ISLAB_2F`** şi **`ISLAB_2G`**, respectiv. Observați că, de această dată, estimația secvenței pondere pare a fi deviată în ambele cazuri. Care credeți că este cauza acestei proprietăți nedorite?

Problema 2.2 (analiză pe bază de corelație – continuare)

d. Pentru a diminua deviația estimației secvenței pondere, se pot aplica 2 tehnici de bază: pre-albire de date sau idealizarea matricii de auto-covarianță a intrării.

2 p

➢ *Pre-albirea datelor*. Funcția MATLAB **cra** efectuează analiza be bază de corelație însoțită de pre-albirea datelor, dacă utilizatorul o doreşte. Această operație constă în filtrarea datelor de intrare-ieşire cu ajutorul unui filtru IIR de tip AR[na]. Implicit, ordinul filtrului este $na = 10$, dar utilizatorul poate specifica propria sa opțiune în acest scop. Albirea datelor (mai ales de intrare) conduce la diminuarea deviației estimației. În general, această tehnică este utilizată atunci cînd nu se cunosc suficiente informații despre maniera în care a fost generată intrarea.

Apelul tipic al funcției **cra** este:

```
[ir,R,cl] = cra(data,M,na,plot) ;
```

unde:

- **data** este blocul de date măsurate (2 coloane: **[y u]**);
- **M** este numărul de valori ale secvenței pondere ce trebuie estimate (implicit: **M=20**);
- **na** este ordinul filtrului de albire IIR-AR (implicit: **na=10**); dacă nu se doreşte pre-albirea datelor, se poate seta **na=0**;
- **plot** este un parametru de afişare grafică; implicit: **plot=1**, care indică trasarea graficului funcției pondere estimate; alte opțiuni recunoscute sunt: **plot=0** (trasarea de grafice este inhibată) şi **plot=2** (se trasează graficele tuturor funcțiilor de corelație implicate);
- **ir** este răspunsul cauzal la impuls (funcția pondere) estimat(ă);
- **R** este o matrice care conține următoarele informații de corelație: pe prima coloană se află pivoții funcțiilor de covarianță; pe coloana a doua se află valorile secvenței de covarianță a ieşirii (după pre-albire, dacă a fost cazul); pe coloana a treia se află valorile secvenței de covarianță a intrării (după pre-albire, dacă este cazul); aceste secvențe pot fi şi direct trasate grafic prin apelul: **cra(R)**;
- **cl** este nivelul de încredere al estimației funcției pondere.

Problema 2.2 (analiză pe bază de corelație – continuare)

3 p

➢ *Idealizarea matricii de auto-covarianță a intrării*. Dacă utilizatorul este la curent cu metoda de generare a intrării şi poate evalua secvența sa de auto-covarianță, atunci matricea ecuației Wiener-Hopf poate fi implementată direct. În acest context, rezolvarea ecuației (care presupune totuşi estimarea corelației încrucişate dintre intrare şi ieşire) conduce la estimații cu deviație diminuată.

Folosind fiecare dintre cele 2 tehnici anterioare, să se modifice mini-simulatoarele **ISLAB_2F** şi **ISLAB_2G** pentru a testa diminuarea deviației estimației în cazul intrării filtrate u_f. În cazul celei de-a doua tehnici, se va evalua întîi secvența de auto-covarianță a intrării în formă completă. Pentru a construi matricea de auto-covarianță a intrării, se poate folosi funcția MATLAB **toeplitz** (avînd în vedere că această matrice este de tip Toeplitz simetrică). Denumiți mini-simulatarele obținute prin **ISLAB_2H** şi **ISLAB_2I** (pentru modelul ARX) şi **ISLAB_2J** şi **ISLAB_2K** (pentru modelul OE).

Program existent pe CD
ISLAB_2D

Programe ce trebuie proiectate	
ISLAB_2E	ISLAB_2I
ISLAB_2F	ISLAB_2J
ISLAB_2G	ISLAB_2K
ISLAB_2H	

Înainte de a rula mini-simulatoarele existente, trebuie executate comenzile:

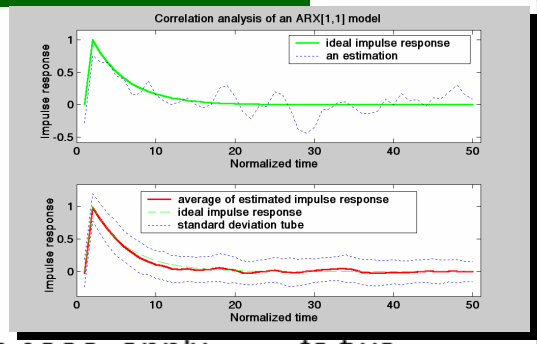
```
> global FIG
> FIG = 1 ;
```

❷ Identificarea modelelor neparametrice

Rutine MATLAB - System Identification toolbox (Problema 2.2)

cra

Perform prewhitening based correlation analysis and estimate impulse response.



Syntax

```
cra(data);
[ir,R,cl] = cra(data,M,na,plot);
cra(R);
```

Description

`data` is the output-input data given as an `iddata` object.

The routine only handles single-input-single-output data pairs. (For the multivariate case, apply `cra` to two signals at a time, or use `impulse`.) `cra` prewhitens the input sequence, i.e., filters `u` through a filter chosen so that the result is as uncorrelated (white) as possible. The output `y` is subjected to the same filter, and then the covariance functions of the filtered `y` and `u` are computed and graphed. The cross correlation function between (prewhitened) input and output is also computed and graphed. Positive values of the lag variable then corresponds to an influence from `u` to later values of `y`. In other words, significant correlation for negative lags is an indication of feedback from `y` to `u` in the data.

A properly scaled version of this correlation function is also an estimate of the system's impulse response `ir`. This is also graphed along with 99% confidence levels. The output argument `ir` is this impulse response estimate, so that its first entry corresponds to lag zero. (Negative lags are excluded in `ir`.) In the plot, the impulse response is scaled, so that it corresponds to an impulse of height $1/T$ and duration T, where T is the sampling interval of the data.

The output argument `R` contains the covariance/correlation information as follows: The first column of `R` contains the lag indices. The second column contains the covariance function of the (possibly filtered) output. The third column contains the covariance function of the (possibly prewhitened) input, and the fourth column contains the correlation function. The plots can be redisplayed by `cra(R)`.

The output argument `cl` is the 99% confidence level for the impulse response estimate.

The optional argument `M` defines the number of lags for which the covariance/correlation functions are computed. These are from `-M` to `M`, so that the length of `R` is `2M+1`. The impulse response is computed from 0 to `M`. The default value of `M` is 20.

For the prewhitening, the input is fitted to an AR model of order `na`. The third argument of `cra` can change this order from its default value `na = 10`. With `na = 0` the covariance and correlation functions of the original data sequences are obtained.

`plot`: `plot = 0` gives no plots. `plot = 1` (default) gives a plot of the estimated impulse response together with a 99% confidence region. `plot = 2` gives a plot of all the covariance functions.

An often better alternative to `cra` are the functions `impulse` and `step`, which use a high order FIR model to estimate the impulse response.

Rutine MATLAB - System Identification toolbox (Problema 2.2)

toeplitz

Toeplitz matrix

Syntax

```
T = toeplitz(c,r)
T = toeplitz(r)
```

$$R_M(u) \overset{def}{=} \begin{bmatrix} r_u[0] & r_u[1] & r_u[2] & \cdots & r_u[M-1] \\ r_u[1] & r_u[0] & r_u[1] & \cdots & r_u[M-2] \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ r_u[M-2] & \cdots & r_u[1] & r_u[0] & r_u[1] \\ r_u[M-1] & \cdots & r_u[2] & r_u[1] & r_u[0] \end{bmatrix}$$

☝ toate elementele matricii se regăsesc pe o coloană (sau o linie)

Description

A *Toeplitz* matrix is defined by one row and one column. A *symmetric Toeplitz* matrix is defined by just one row. `toeplitz` generates Toeplitz matrices given just the row or row and column description.

`T = toeplitz(c,r)` returns a nonsymmetric Toeplitz matrix `T` having `c` as its first column and `r` as its first row. If the first elements of `c` and `r` are different, a message is printed and the column element is used.

`T = toeplitz(r)` returns the symmetric or Hermitian Toeplitz matrix formed from vector `r`, where `r` defines the first row of the matrix.

❷ Identificarea modelelor neparametrice

Problema 2.3 (analiză spectrală) 2p

Ultimul tip de analiză, cea spectrală, va fi ilustrat în contextul acestei probleme. Prin analiza spectrală se urmăreşte estimarea răspunsului în frecvență al unui proces furnizor de date, folosind ecuația transferului densității spectrale încrucişate prin sisteme liniare. Ecuația poate fi rezolvată dacă se estimează mai întîi densitatea spectrală a intrării (ϕ_u) şi densitatea spectrală încrucişată a intrării şi ieşirii (ϕ_{uy}). Funcția MATLAB care efectuează analiza spectrală plecînd de la date măsurate este **spa**. Apelul tipic al acesteia este:

```
H = spa(data,M,w) ;
```

unde: **data** este blocul de date măsurate (2 coloane: **[y u]**);

M este dimensiunea ferestrei Hamming aplicate datelor (implicit: **M = min(length(data)/10,30)**);

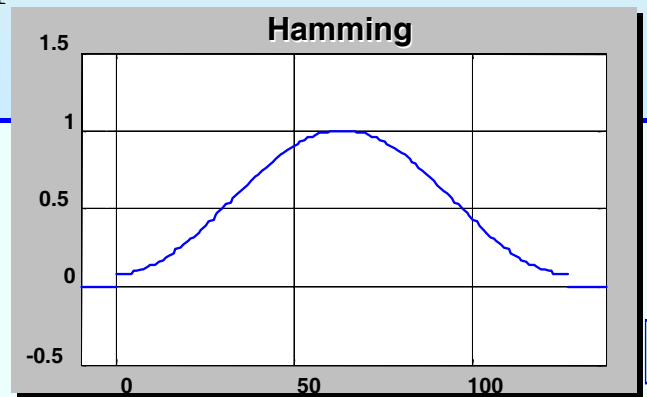
w este vectorul nodurilor de frecvență unde se doreşte estimat răspunsul în frecvență al sistemului;

H este răspunsul în frecvență al sistemului.

De notat că fereastra Hamming este una dintre cele mai convenabile pentru estimarea spectrală, avînd expresia:

$$W[n] = 0.54 - 0.46 \cos \frac{2n\pi}{M-1}, \quad \forall\, n \in \mathbb{N},$$

unde M este deschiderea ferestrei.



$$\phi_{uy}(\omega) = H(e^{j\omega})\phi_u(\omega)$$

❷ Identificarea modelelor neparametrice

Problema 2.3 (analiză spectrală – continuare)

Mini-simulatorul **ISLAB_2L** (al cărui listing se află înregistrat pe CD) a fost proiectat pentru efectuarea analizei spectrale a modelului ARX de ordin I, în cazul în care sistemul este stimulat cu intrarea filtrată u_f (din **Problema 2.2**). Diagrama Bode afişată de funcţia **ISLAB_2L** este comparată cu răspunsul în frecvenţă ideal dedus direct din ecuaţia modelului (vezi **Exerciţiul 2.4**).

0.1 p a. Efectuaţi cîteva simulări cu diferite valori ale deschiderii ferestrei (M) pentru a observa influenţa acestui parametru asupra calităţii estimaţiei şi a propune o valoare rezonabilă a lui.

0.4 p b. Înlocuiţi semnalul de stimul u_f din mini-simulatorul **ISLAB_2L** cu un zgomot alb (adică un SPAB). Denumiţi noul simulator prin **ISLAB_2M** şi repetaţi experimentul de la punctul precedent. Comparaţi rezultatele celor 2 mini- simulatoare **ISLAB_2L** şi **ISLAB_2M** pentru cele mai bune valori ale deschiderii ferestrei Hamming găsite în fiecare caz.

0.5 p c. Proiectaţi mini-simulatoarele **ISLAB_2N** şi **ISLAB_2O** inspirate de cele 2 mini-simulatoare anterioare, dar pentru modelul OE de ordin I. Efectuaţi din nou o analiză comparativă.

1 p d. Proiectaţi mini-simulatoarele **ISLAB_2P**, **ISLAB_2Q**, **ISLAB_2R** şi **ISLAB_2S** inspirate de cele 4 mini-simulatoare anterioare, dar pentru modelele ARX şi OE de ordin II. Repetaţi analiza comparativă.

Program existent pe CD	Programe ce trebuie proiectate	
ISLAB_2L	ISLAB_2M	ISLAB_2Q
	ISLAB_2N	ISLAB_2R
	ISLAB_2O	ISLAB_2S
	ISLAB_2P	

Înainte de a rula mini-simulatoarele existente, trebuie executate comenzile:

```
> global FIG
> FIG = 1 ;
```

❷ Identificarea modelelor neparametrice

Despre biblioteca MATLAB de IS - System Identification toolbox

- Proiectată în tehnologia Programării Orientate Obiect (POO).
- Obiectul principal ⇨ **IDDATA**

D Structură de date intrare-ieşire generate folosind ecuaţiile modelului ales. Este creată cu ajutorul funcţiei (metodei) constructor **iddata** asociată obiectului **IDDATA** (date de identificare). Cele 2 matrici de date **u** (de intrare) şi **y** (de ieşire) au dimensiunile identice: **N** linii şi **nr** coloane. Fiecare din cele **nr** experimente (realizări) ocupă cîte o coloană cu **N** perechi de date intrare-ieşire). Datele se regăsesc în cîmpurile **D.u** şi **D.y** ale structurii **D**. Structura este mult mai complexă şi se compune din următoarele cîmpuri:

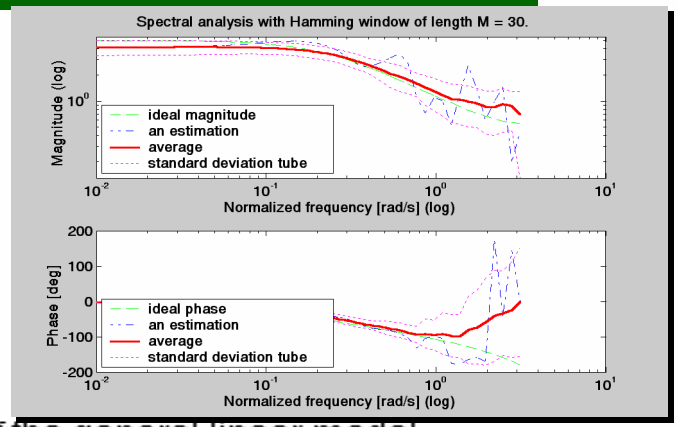
```
                Domain: ''Time'/'Frequency''
                  Name: 'String'
            OutputData: [1x37 char]
                     y: 'Same as OutputData'   ← date de ieşire
            OutputName: 'Ny-by-1 cell array of strings'
            OutputUnit: 'Ny-by-1 cell array of strings'
             InputData: [1x36 char]
                     u: 'Same as InputData'    ← date de intrare
             InputName: 'Nu-by-1 cell array of strings'
             InputUnit: 'Nu-by-1 cell array of strings'
                Period: [1x51 char]
           InterSample: [1x36 char]
                    Ts: [1x54 char]            ← perioadă de eşantionare
                Tstart: 'Scalar  (Starting time)'
      SamplingInstants: [1x51 char]
              TimeUnit: 'String'
        ExperimentName: [1x43 char]
                 Notes: 'Cell array of strings'
              UserData: 'Arbitrary'
```

❷ Identificarea modelelor neparametrice

Rutine MATLAB - System Identification toolbox (Problema 2.3)

spa

Estimate frequency response and spectrum by spectral analysis.



Syntax

```
g = spa(data)
g = spa(data,M,w,maxsize)
[g,phi,spe] = spa(data)
```

Description

spa estimates the transfer function g and the noise spectrum Φ_v of the general linear model

$$y(t) = G(q)u(t) + v(t)$$

where $\Phi_v(\omega)$ is the spectrum of $v(t)$.

data contains the output-input data as an iddata object. The data may be complex-valued.

g is returned as an idfrd object (see idfrd) with the estimate of $G(e^{i\omega})$ at the frequencies ω specified by row vector w. The default value of w is

```
w = [1:128]/128*pi/Ts
```

Here Ts is the sampling interval of data.

g also includes information about the spectrum estimate of $\Phi_v(\omega)$ at the same frequencies. Both outputs are returned with estimated covariances, included in g. See idfrd.

M is the length of the lag window used in the calculations. The default value is

```
M = min(30,length(data)/10)
```

Rutine MATLAB - System Identification toolbox (Problema 2.3)

spa

Estimate frequency response and spectrum by spectral analysis.

Changing the value of `M` exchanges bias for variance in the spectral estimate. As `M` is increased, the estimated functions show more detail, but are more corrupted by noise. The sharper peaks a true frequency function has, the higher `M` it needs. See `etfe` as an alternative for narrowband signals and systems. See also Estimating Spectra and Frequency Functions in the "Tutorial".

`maxsize` controls the memory-speed trade-off (see `Algorithm Properties`).

For time series, where data contains no input channels, `g` is returned with the estimated output spectrum and its estimated standard deviation.

When `spa` is called with two or three output arguments:

- `g` is returned as an `idfrd` model with just the estimated frequency response from input to output and its uncertainty.
- `phi` is returned as an `idfrd` model, containing just the spectrum data for the output spectrum $\Phi_v(\omega)$ and its uncertainty.
- `spe` is returned as an `idfrd` model containing spectrum data for all output-input channels in data. That is if `z = [data.OutputData, data.InputData]`, `spe` contains as spectrum data the matrix-valued power spectrum of `z`.

$$S = \sum_{m=-M}^{M} Ez(t+m)z(t)'\exp(-iWmT)win(m)$$

Here *win(m)* is weight at lag `m` of an M-size Hamming window and *W* is the frequency value i rad/s. Note that ' denotes complex-conjugate transpose.

The normalization of the spectrum differs from the one used by `spectrum` in the Signal Processing Toolbox. See Spectrum Normalization and the Sampling Interval in the "Tutorial" for a more precise definition.

❷ Identificarea modelelor neparametrice

Ce returnează funcția spa

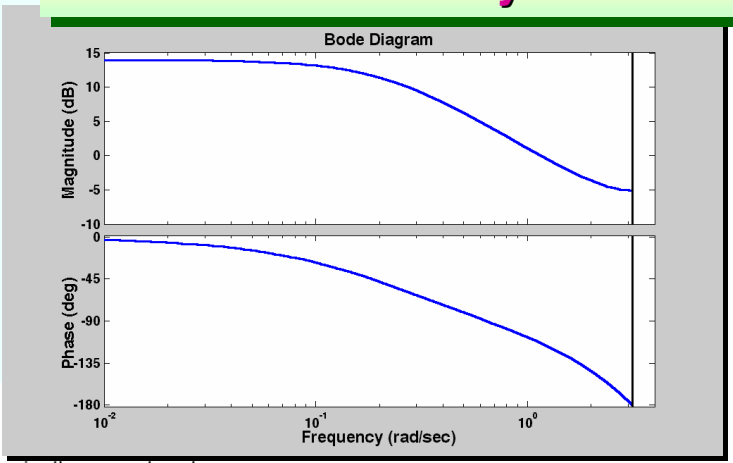
- **Obiect** → **FREQH**

H Structură de date reprezentînd răspunsul în frecvență estimat folosind funcția MATLAB **spa**. Răspunsurile în frecvență estimate folosind datele **D** se regăsesc în blocul tri-dimensional **H.ResponseData**, astfel: prima estimație se află în vectorul **H.ResponseData(1,1,:)**, a doua estimație se află în vectorul **H.ResponseData(2,2,:)**, etc. Sunt **nr** estimații în total. Fiecare estimație conține **K** date cu valori complexe (partea reală şi partea imaginară a răspunsului în frecvență). Numărul **K** figurează printre argumentele funcției **ISLAB_2L** şi reprezintă numărul de noduri (echidistante) de frecvență în care se doresc estimate valorile răspunsului în frecvență. Ca şi în cazul structurii de date, structura răspunsului în frecvență este mai complexă, incluzînd următoarele cîmpuri (din care se constată că funcția specializată **spa** poate oferi numeroase informații spectrale sau de corelație referitoare la datele măsurate şi zgomot):

```
                Name: 'string'
           Frequency: [1x48 char]   ← noduri de frecvenţă
        ResponseData: [1x40 char]   ← răspuns în frecvenţă
        SpectrumData: [1x38 char]   ← spectru
      CovarianceData: [1x62 char]   ← secvenţă de (auto)covarianţă
    NoiseCovariance: [1x57 char]
               Units: '['rad/s'|'Hz']'
                  Ts: 'scalar'      ← perioadă de eşantionare
          InputDelay: 'Nu-by-1 vector'
      EstimationInfo: 'structure'
           InputName: 'Nu-by-1 cell array of strings'
          OutputName: 'Ny-by-1 cell array of strings'
           InputUnit: 'Nu-by-1 cell array of strings'
          OutputUnit: 'Ny-by-1 cell array of strings'
               Notes: 'Array or cell array of strings'
            UserData: 'Arbitrary'
             Version: 'Internal Use'
             Utility: 'Internal Use'
```

❷ Identificarea modelelor neparametrice

Rutine MATLAB - System Identification toolbox (Problema 2.3)



dbode Bode frequency response for discrete-time linear systems.

dbode(A,B,C,D,Ts,IU) produces a Bode plot from the single input IU to all the outputs of the discrete state-space system (A,B,C,D). IU is an index into the inputs of the system and specifies which input to use for the Bode response. Ts is the sample period. The frequency range and number of points are chosen automatically.

dbode(NUM,DEN,Ts) produces the Bode plot for the polynomial transfer function G(z) = NUM(z)/DEN(z) where NUM and DEN contain the polynomial coefficients in descending powers of z.

dbode(A,B,C,D,Ts,IU,W) or **dbode**(NUM,DEN,Ts,W) uses the user-supplied freq. vector W which must contain the frequencies, in radians/sec, at which the Bode response is to be evaluated. Aliasing will occur at frequencies greater than the Nyquist frequency (pi/Ts). See LOGSPACE to generate log. spaced frequency vectors. With left hand arguments,

[MAG,PHASE,W] = **dbode**(A,B,C,D,Ts,...)

[MAG,PHASE,W] = **dbode**(NUM,DEN,Ts,...)

returns the frequency vector W and matrices MAG and PHASE (in degrees) with as many columns as outputs and length(W) rows. No plot is drawn on the screen.

logspace

Generate logarithmically spaced vectors

Syntax

```
y = logspace(a,b)
y = logspace(a,b,n)
y = logspace(a,pi)
```

Description

The `logspace` function generates logarithmically spaced vectors. Especially useful for creating frequency vectors, it is a logarithmic equivalent of `linspace` and the ":" or colon operator.

`y = logspace(a,b)` generates a row vector `y` of 50 logarithmically spaced points between decades `10^a` and `10^b`.

`y = logspace(a,b,n)` generates `n` points between decades `10^a` and `10^b`.

`y = logspace(a,pi)` generates the points between `10^a` and `pi`, which is useful for digital signal processing where frequencies over this interval go around the unit circle.

Remarks

All the arguments to `logspace` must be scalars.