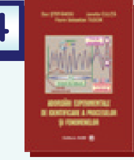


❸ Identificare parametrică prin MCMMP

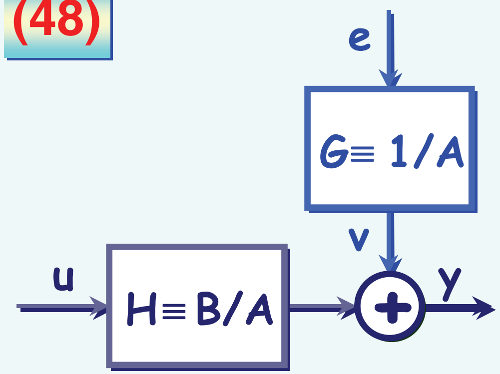
☞ Contextul de lucru

7.5 p

Două modele ARMAX

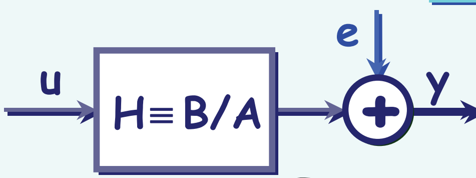
Auto-Regresiv cu Control eXogen

$$\text{ARX[na,nb]}: \quad A(q^{-1})y[n] = B(q^{-1})u[n] + e[n] \qquad (48)$$



Eroare de Ieşire (Output Error)

$$\text{OE[na,nb]}: \quad y[n] = \frac{B(q^{-1})}{A(q^{-1})}u[n] + e[n] \qquad (49)$$



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

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

(zgomot alb)

Cazuri particulare

Ordin I

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

$$B(q^{-1}) = q^{-1} \qquad (50)$$

Ordin II

$$A(q^{-1}) = 1 - 0.4q^{-1} - 0.32q^{-2}$$

$$B(q^{-1}) = 0.5q^{-1} + 0.03q^{-2} \qquad (55)$$

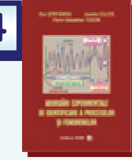
☝ Posedă rădăcini parazite.

- 2 tipuri de intrări:
  - u (SPAB bipolar, de regulă)
  - $u_f[n] \overset{def}{=} \dfrac{0.6}{1-0.8q^{-1}}u[n]$ (filtrat)

Obiectiv

- Identificarea:
  - ➔ parametrilor modelelor ARX
  - ➔ parametrilor modelelor OE

folosind MCMMP.



❸ Identificare parametrică prin MCMMP

☞ Contextul de lucru

Forma de regresie liniară a unui model ARMAX ⇒ $y[n] = \varphi^T[n]\,\boldsymbol{\theta} + e[n]$ $\quad \forall n \in \mathbb{N}$

vectorul regresorilor — vectorul parametrilor necunoscuți

$$\varphi^T[n] \overset{def}{=} \big[-y[n-1]\; -y[n-2] \cdots -y[n-na] \;\big|\; u[n-1]\; u[n-2] \cdots u[n-nb] \;\big|\; \ldots$$

☝ componentă nemăsurabilă (zgomot alb) ➔ $\ldots\; e[n-1]\; e[n-2] \cdots e[n-nc]\big]$

$$\boldsymbol{\theta}^T \overset{def}{=} \big[a_1\; a_2 \cdots a_{na} \;\big|\; b_1\; b_2 \cdots b_{nb} \;\big|\; c_1\; c_2 \cdots c_{nc}\big] \qquad \forall n \in \mathbb{N}$$

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

Metoda Celor Mai Mici Pătrate (MCMMP)

☝ Centrate pe medii.

- Dacă s-ar dispune de o infinitate de realizări:

E ② $\quad \varphi[n]\times$ ① $\quad y[n] = \varphi^T[n]\boldsymbol{\theta} + e[n]$

$$\boldsymbol{\theta}^* = \left(E\{\varphi[n]\varphi^T[n]\}\right)^{-1}\left(E\{\varphi[n]y[n]\} - E\{\varphi[n]e[n]\}\right)$$

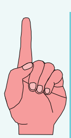
$$(\lambda^*)^2 = E\left\{\left(y[n] - \varphi^T[n]\boldsymbol{\theta}^*\right)^2\right\}$$

Relații teoretice de estimare

- Pentru o singură realizare finită:

$$\hat{\boldsymbol{\theta}}_N \overset{def}{=} \underbrace{\left(\frac{1}{N}\sum_{n=1}^{N}\tilde{\varphi}[n]\tilde{\varphi}^T[n]\right)^{-1}}_{\mathbf{R}_N^{-1}} \underbrace{\left(\frac{1}{N}\sum_{n=1}^{N}\tilde{\varphi}[n]\tilde{y}[n]\right)}_{\mathbf{r}_N}$$

$$\hat{\lambda}_N^2 \overset{def}{=} \frac{1}{N}\sum_{n=1}^{N}\left(y[n] - \varphi^T[n]\hat{\boldsymbol{\theta}}_N\right)^2$$



$$\hat{\boldsymbol{\theta}}_N = \underset{\boldsymbol{\theta}\in\mathcal{S}}{\operatorname{argmin}}\, \mathcal{V}_N(\boldsymbol{\theta}) = \underset{\boldsymbol{\theta}\in\mathcal{S}}{\operatorname{argmin}} \sum_{n=1}^{N}\left(y[n] - \varphi[n]\boldsymbol{\theta}\right)^2$$

Relații practice de estimare

❸ Identificare parametrică prin MCMMP

☞ Probleme de simulare

Problema 4.1 (MCMMP pentru modelele ARX)

Se studiază influența semnalului de intrare asupra calității estimației oferite de MCMMP pentru modelele ARX. Aceste modele vor fi stimulate de cîte 100 de ori cu fiecare din cele 2 intrări (adică u – un SPAB bipolar de lungime 100, avînd doar valorile –1 sau +1 şi u_f – un semnal de joasă frecvență generat prin filtrarea semnalului u). După achiziția datelor de intrare-ieşire, se vor implementa relațiile de calcul ale estimațiilor parametrilor necunoscuți din **Exercițiile 4.1** şi **4.2**. Estimațiile parametrilor vor fi mediate peste ansamblul celor 100 de realizări şi li se vor calcula deviațiile standard. Cele 4 mini-simulatoare obținute vor fi denumite prin: **ISLAB_4A** (model ARX[1,1] & intrare u), **ISLAB_4B** (model ARX[1,1] & intrare u_f), **ISLAB_4C** (model ARX[2,2] & intrare u) şi **ISLAB_4D** (model ARX[2,2] & intrare u_f). Mini-simulatorul **ISLAB_4D** a fost deja implementat, ca model de lucru.

a. Pentru fiecare mini-simulator, să se reprezinte grafic, în cadrul a două ferestre, erorile de estimare a răspunsului în frecvență după cum urmează:

- În partea superioară a primei fereastre, va fi trasat graficul erorii de estimare a amplitudinii răspunsului în frecvență, adică media amplitudinii diferenței dintre răspunsul în frecvență ideal (în absența zgomotului) şi răspunsurile în frecvență obținute din cele 100 de realizări (după estimarea parametrilor necunoscuți). Tubul de dispersie a amplitudinii se va evalua ca în problemele din capitolul precedent pentru fiecare eroare de estimare şi se va trasa pe acelaşi grafic.
- În partea inferioară a ferestrei, va fi trasat graficul erorii de estimare a fazei răspunsului în frecvență, adică media fazei diferenței dintre răspunsul în frecvență ideal (în absența zgomotului) şi răspunsurile în frecvență obținute din cele 100 de realizări (după estimarea parametrilor necunoscuți). Se va evalua tubul de dispersie a fazei pentru fiecare eroare de estimare şi se va trasa pe acelaşi grafic.
- Într-o a doua fereastră, vor fi trasate graficul dispersiei estimate a zgomotului (obținută pentru fiecare realizare a procesului) şi graficul valorii adevărate a dispersiei zgomotului ($\lambda^2 = 1$). În cadrul figurii, se specifică valorile parametrilor adevărați şi media valorilor parametrilor estimați (calculată peste ansamblul realizărilor).

❸ Identificare parametrică prin MCMMP

Problema 4.1 (MCMMP pentru modelele ARX)

Pentru determinarea răspunsurilor în frecvență se va utiliza funcția MATLAB **dbode**. Nu va fi în nici un caz utilizată funcția de analiză spectrală **spa**, deoarece răspunsul în frecvență estimat trebuie obținut prin combinația dintre MCMMP şi **dbode**. De asemenea, în cazul modelului ARX[2,2], funcțiile de covarianță implicate de relațiile de estimare ale MCMMP pot fi evaluate cu precizie ridicată folosind funcția MATLAB **xcov**, dacă este utilizată cu atenție. După implementarea mini-simulatoarelor, se va testa funcționarea lor, prin cîteva rulări succesive.

1p

b. Comentați rezultatele obținute la punctul precedent. Observați influența tipului de intrare asupra estimării rădăcinilor parazite din modelul particular ARX[2,2]. Dacă acest proces nu va putea fi stimulat decît cu intrări de joasă frecvență, cum credeți că s-ar putea estima (fie şi imprecis) rădăcinile parazite?

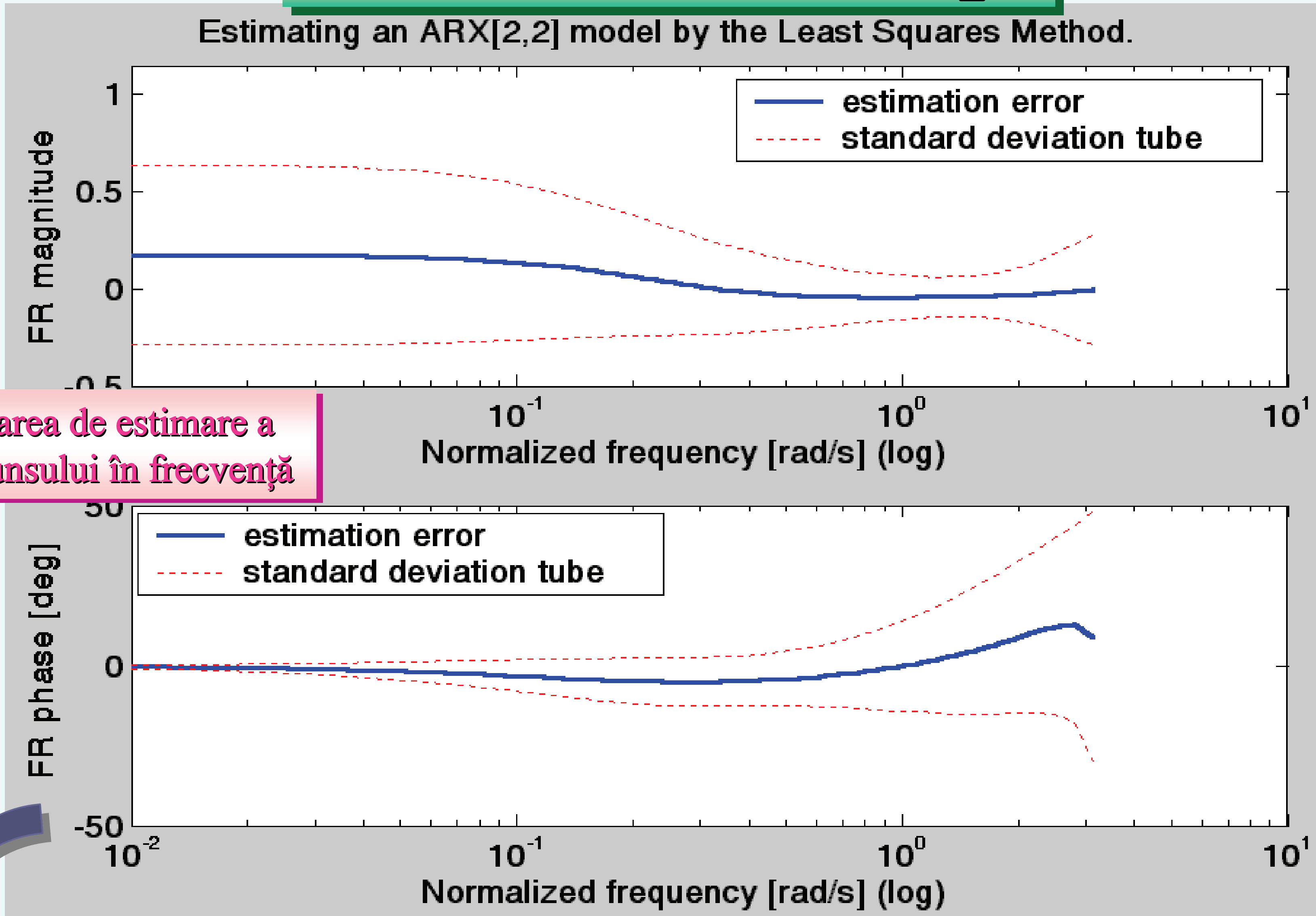
Program existent	Programe ce trebuie proiectate	
ISLAB_4D	ISLAB_4A	0.75p
	ISLAB_4B	0.25p
	ISLAB_4C	0.25p

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

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

❸ Identificare parametrică prin MCMMP

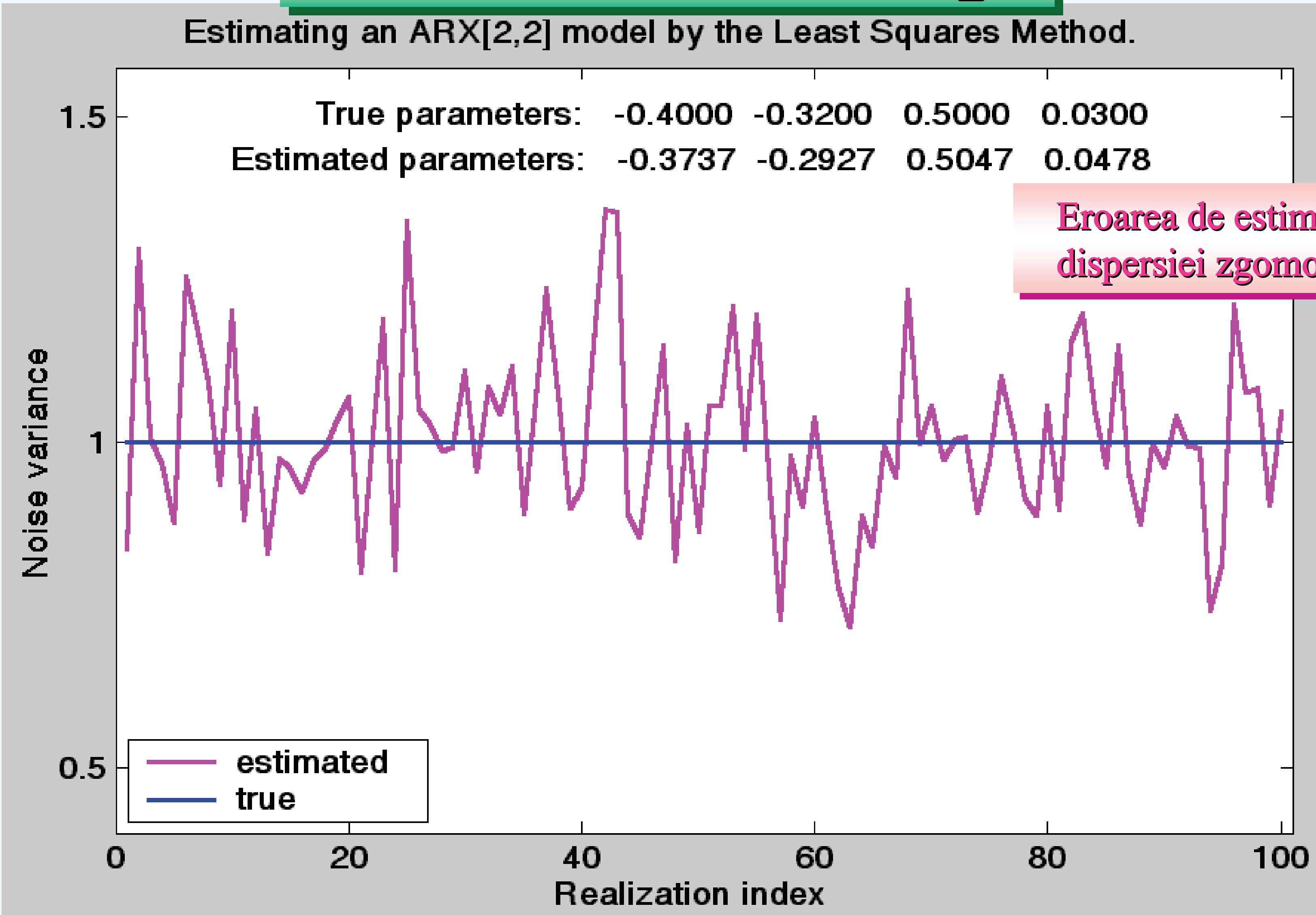
Ce afişează mini-simulatorul ISLAB_4D



Eroarea de estimare a răspunsului în frecvenţă

❸ Identificare parametrică prin MCMMP

Ce afişează mini-simulatorul ISLAB_4D



Eroarea de estimare a dispersiei zgomotului

❸ Identificare parametrică prin MCMMP

Rutine MATLAB (Problema 4.1)

xcov

Estimate the cross-covariance function (mean-removed cross-correlation)

Syntax

```
v = xcov(x,y)
v = xcov(x)
v = xcov(x,'option')
[c,lags] = xcov(x,y,maxlags)
[c,lags] = xcov(x,maxlags)
[c,lags] = xcov(x,y,maxlags,'option')
```

Description

xcov estimates the cross-covariance sequence of random processes. Autocovariance is handled as a special case.

The true cross-covariance sequence is the cross-correlation of mean-removed sequences

- $$\phi_{xy}(\mu) = E\{(x_{n+m} - \mu_x)(y_n - \mu_y)^*\}$$

where μ_x and μ_y are the mean values of the two stationary random processes, and $E\{\cdot\}$ is the expected value operator. xcov estimates the sequence because, in practice, access is available to only a finite segment of the infinite-length random process.

`v = xcov(x,y)` returns the cross-covariance sequence in a length $2N$-1 vector, where x and y are length N vectors. For information on how arrays are processed with xcov, see Multiple Channels.

`v = xcov(x)` is the autocovariance sequence for the vector x. Where x is an N-by-P array, v = xcov(x) returns an array with $2N$-1 rows whose P^2 columns contain the cross-covariance sequences for all combinations of the columns of x.

By default, xcov computes raw covariances with no normalization. For a length N vector

- $$c_{xy}(m) = \begin{cases} \displaystyle\sum_{n=0}^{N-|m|-1} \left(x(n+m) - \frac{1}{N}\sum_{i=0}^{N-1} x_i\right)\left(y_n^* - \frac{1}{N}\sum_{i=0}^{N-1} y_i^*\right) & m \geq 0 \\ c_{yx}^*(-m) & m < 0 \end{cases}$$

❸ Identificare parametrică prin MCMMP

Rutine MATLAB (Problema 4.1)

xcov

Estimate the cross-covariance function (mean-removed cross-correlation)

The output vector `c` has elements given by `c(m)` = $c_{xy}(m\text{-}N)$, $m = 1, ..., 2N\text{-}1$.

The covariance function requires normalization to estimate the function properly.

`v = xcov(x,'option')` specifies a scaling option, where `'option'` is

- `'biased'`, for biased estimates of the cross-covariance function
- `'unbiased'`, for unbiased estimates of the cross-covariance function
- `'coeff'`, to normalize the sequence so the auto-covariances at zero lag are identically 1.0
- `'none'`, to use the raw, unscaled cross-covariances (default)

`[c,lags] = xcov(x,y,maxlags)` where `x` and `y` are length `m` vectors, returns the cross-covariance sequence in a length `2*maxlags+1` vector `c`. `lags` is a vector of the lag indices where `c` was estimated, that is, `[-maxlags:maxlags]`.

`[c,lags] = xcov(x,maxlags)` is the autocovariance sequence over the range of lags `[-maxlags:maxlags]`.

`[c,lags] = xcov(x,maxlags)` where `x` is an `m`-by-`p` array, returns array `c` with `2*maxlags+1` rows whose P^2 columns contain the cross-covariance sequences for all combinations of the columns of `x`.

`[c,lags] = xcov(x,y,maxlags,'option')` specifies a scaling option, where `'option'` is the last input argument.

In all cases, `xcov` gives an output such that the zeroth lag of the covariance vector is in the middle of the sequence at element or row `maxlag+1` or at `m`.

Algorithm

`xcov` computes the mean of its inputs, subtracts the mean, and then calls `xcorr`. For more information on estimating covariance and correlation functions, see [1].

---

❸ Identificare parametrică prin MCMMP

Problema 4.2 (MCMMP pentru modelele OE)

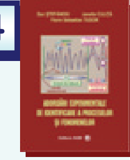
Dacă ați ajuns la concluzia că modelele OE (49) & (50), respectiv (49) & (55) ar putea fi identificate cu ajutorul MCMMP, reluați **Problema 4.1** pentru cazul acestor modele. Denumiți mini-simulatoarele obținute prin **ISLAB_4E** (model OE[1,1] & intrare u), **ISLAB_4F** (model OE[1,1] & intrare u_f), **ISLAB_4G** (model OE[2,2] & intrare u) şi **ISLAB_4H** (model OE[2,2] & intrare u_f).

Programe ce trebuie proiectate

ISLAB_4E	0.75p
ISLAB_4F	0.25p
ISLAB_4G	0.5p
ISLAB_4H	0.25p
Comentarii	1.5p

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

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



❸ Identificare parametrică prin MCMMP

Problema 4.3 (Generalizare)

Generalizați mini-simulatoarele anterioare şi denumiți noile rutine prin **ISLAB_4I** (pentru modele ARX[na,nb]) şi, dacă este cazul, **ISLAB_4J** (pentru modele OE[na,nb]). În acest scop, se poate utiliza funcția de bibliotecă IS MATLAB numită **arx**. Apelul tipic al acestei rutine este următorul:

```
theta = arx(D,si) ;
```

unde: **D** este structura datelor de intrare-ieşire, de regulă creată cu ajutorul funcției (metodei) constructor asociată obiectului **IDDATA** (vezi comentariile privind proiectarea mini-simulatorului **ISLAB_3L** din finalul **Capitolului 3**);

si este vectorul indicilor structurali şi al întîrzierii modelului:

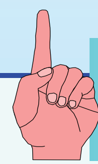
si = [na nb nk],

unde **na** este ordinul componentei AR, iar **nb+nk** este ordinul componentei X; practic, **nk** este numărul de coeficienți nuli ai polinomului B, pînă la primul coeficient nenul de grad minim (adică întîrzierea intrinsecă a modelului); urmează cei **nb** coeficienți nenuli.

Argumentul de ieşire **theta** este la rîndul său un obiect de tip **IDPOLY** (*polinom de identificare* – în cazul modelelor SISO) sau **IDMODEL** (*model general de identificare* în cazul modelelor SISO sau MISO).

Programe ce trebuie proiectate | **Comentarii**

ISLAB_4I 1p | **ISLAB_4J** 0.5p | Comentarii 0.5p



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

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

❸ Identificare parametrică prin MCMMP

Rutine MATLAB - System Identification toolbox (Problema 4.3)

arx

Estimate the parameters of an ARX or AR model.

Syntax

```
m = arx(data,orders)
m = arx(data,'na',na,'nb',nb,'nk',nk)
m= arx(data,orders,'Property1',Value1,...,'PropertyN',ValueN)
```

Description

The parameters of the ARX model structure

- $A(q)y(t) = B(q)u(t-nk) + e(t)$

are estimated using the least-squares method.

data is an iddata object that contains the output-input data. orders is given as

```
orders = [na nb nk]
```

defining the orders and delay of the ARX model. Specifically,

$$na: \quad A(q) = 1 + a_1 q^{-1} + \ldots + a_{na} q^{-na}$$

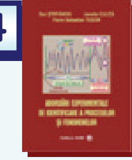
$$nb: \quad B(q) = b_1 + b_2 q^{-1} + \ldots + b_{nb} q^{-nb+1}$$

See Polynomial Representation of Transfer Functions in the "Tutorial" for more information. The model orders can also be defined by explicit pairs (...,'na',na,'nb',nb,'nk',nk,...).

m is returned as the least-squares estimates of the parameters. For single-output data this is an idpoly object, otherwise an idarx object.

For a time series, data contains no input channels and orders = na. Then an AR model of order na for y is computed.

- $A(q)y(t) = e(t)$



❸ Identificare parametrică prin MCMMP

Rutine MATLAB - System Identification toolbox (Problema 4.3)

arx

Estimate the parameters of an ARX or AR model.

Models with several inputs

- $A(q)y(t) = B_1(q)u_1(t-nk_1) + \ldots B_{nu}u_{nu}(t-nk_{nu}) + e(t)$

are handled by allowing `nb` and `nk` to be row vectors defining the orders and delays associated with each input.

Models with several inputs and several outputs are handled by allowing `na`, `nb`, and `nk` to contain one row for each output number. See Multivariable ARX Models: The idarx Model in the "Tutorial" for exact definitions.

The algorithm and model structure are affected by the property name/property value list in the input argument.

Useful options are reached by the properties `'Focus'`, `'InputDelay'`, and `'MaxSize'`.

See `Algorithm Properties` for details of these properties and possible values

When the true noise term $e(t)$ in the ARX model structure is not white noise and `na` is nonzero, the estimate does not give a correct model. It is then better to use `armax`, `bj`, `iv4`, or `oe`.

Examples

Here is an example that generates data and estimates an ARX model.

-

```
A = [1  -1.5  0.7]; B = [0 1 0.5];
m0 = idpoly(A,B);
u = iddata([],idinput(300,'rbs'));
e = iddata([],randn(300,1));
y = sim(m0, [u e]);
z = [y,u];
m = arx(z,[2 2 1]);
```

Algorithm

The least squares estimation problem is an overdetermined set of linear equations that is solved using QR-factorization.

The regression matrix is formed so that only measured quantities are used (no fill-out with zeros[illegible]hen the regression matrix is larger than `MaxSize`, the QR-factorization is performed in a `for`-loop.