7. FILTRE ADAPTIVE BAZATE PE METODA CELOR MAI MICI PĂTRATE

7.1 Metoda recursivă a celor mai mici pătrate (RLS)

7.1 Metoda recursivă a celor mai mici pătrate (RLS)

7.1.1 Algoritmul RLS

Funcţia cost: eroare pătratică ponderată

$$J(n)=\sum_{i=1}^{n}\beta(n,i)|e(i)|^2$$

în care

$$e(i)=d(i)-y(i)=d(i)-\mathbf{w}^H(n)\mathbf{x}(i);$$
$$\mathbf{x}(i)=[x(i),x(i-1),\cdots,x(i-N+1)]^T$$

Factorul de pondere

$$\beta(n,i)=\lambda^{n-i},\quad i=1,\cdots,n,\quad 0<\lambda\le 1$$

$$J(n)=\sum_{i=1}^{n}\lambda^{n-i}|e(i)|^2$$

7.1 Metoda recursivă a celor mai mici pătrate (RLS)

7.1.1 Algoritmul RLS

$$J(n)=\sum_{i=1}^{n}\lambda^{n-i}\left(d(i)-\mathbf{w}^{H}(n)\mathbf{x}(i)\right)\left(d^{*}(i)-\mathbf{x}^{H}(i)\mathbf{w}(n)\right)$$

$$J(n)=\sum_{i=1}^{n}\lambda^{n-i}\left|d(i)\right|^{2}-\mathbf{w}^{H}(n)\sum_{i=1}^{n}\lambda^{n-i}\mathbf{x}(i)d^{*}(i)-$$

$$-\sum_{i=1}^{n}\lambda^{n-i}\mathbf{x}^{H}(i)d(i)\mathbf{w}(n)+\mathbf{w}^{H}(n)\sum_{i=1}^{n}\lambda^{n-i}\mathbf{x}(i)\mathbf{x}^{H}(i)\mathbf{w}(n)$$

Notăm

$$\mathbf{\Phi}(n)=\sum_{i=1}^{n}\lambda^{n-i}\mathbf{x}(i)\mathbf{x}^{H}(i)$$

$$\boldsymbol{\theta}(n)=\sum_{i=1}^{n}\lambda^{n-i}\mathbf{x}(i)d^{*}(i)$$

$$E_{d}(n)=\sum_{i=1}^{n}\lambda^{n-i}\left|d(i)\right|^{2}$$

7.1 Metoda recursivă a celor mai mici pătrate (RLS)

7.1.1 Algoritmul RLS

$$\mathbf{\Phi}(n) = \sum_{i=1}^{n} \lambda^{n-i} \mathbf{x}(i)\mathbf{x}^{H}(i)$$

$$\boldsymbol{\theta}(n) = \sum_{i=1}^{n} \lambda^{n-i} \mathbf{x}(i)d^{*}(i)$$

$$E_d(n) = \sum_{i=1}^{n} \lambda^{n-i} \left|d(i)\right|^2$$

$\mathbf{\Phi}(n)$, $\boldsymbol{\theta}(n)$, $E_d(n)$ reprezintă estimatori deplasaţi ai matricei de autocorelaţie, vectorului corelaţiei între semnalul dorit şi secvenţa de intrare şi respectiv puterii semnalului dorit.

$$J(n) = E_d - \mathbf{w}^{H}(n)\boldsymbol{\theta}(n) - \boldsymbol{\theta}^{H}(n)\mathbf{w}(n) + \mathbf{w}^{H}(n)\mathbf{\Phi}(n)\mathbf{w}(n)$$

Anulând gradientul funcţiei cost se obţine ecuaţia normală.

7.1 Metoda recursivă a celor mai mici pătrate (RLS)

7.1.1 Algoritmul RLS

Ecuaţia normală. Minimizarea funcţiei cost se obţine tot pentru

$$\boldsymbol{\Phi}(n)\mathbf{w}(n)=\boldsymbol{\theta}(n)$$

Presupunând problema rezolvată pentru n-1, deci cunoscând:

$$\mathbf{w}(n-1)=\boldsymbol{\Phi}^{-1}(n-1)\boldsymbol{\theta}(n-1)$$

ne propunem să găsim o metodă pentru a evalua

$$\mathbf{w}(n)=\boldsymbol{\Phi}^{-1}(n)\boldsymbol{\theta}(n)$$

7.1 Metoda recursivă a celor mai mici pătrate (RLS)

7.1.1 Algoritmul RLS

Relaţii de recurenţă pentru $\mathbf{\Phi}(n)$ ***şi*** $\boldsymbol{\theta}(n)$:

$$\mathbf{\Phi}(n) = \lambda \sum_{i=1}^{n-1} \lambda^{n-1-i} \mathbf{x}(i)\mathbf{x}^H(i) + \mathbf{x}(n)\mathbf{x}^H(n)$$

$$\mathbf{\Phi}(n) = \lambda \mathbf{\Phi}(n-1) + \mathbf{x}(n)\mathbf{x}^H(n)$$

$$\boldsymbol{\theta}(n) = \lambda \sum_{i=1}^{n-1} \lambda^{n-1-i} \mathbf{x}(i)d^*(i) + \mathbf{x}(n)d^*(n)$$

$$\boldsymbol{\theta}(n) = \lambda \boldsymbol{\theta}(n-1) + \mathbf{x}(n)d^*(n)$$

7.1 Metoda recursivă a celor mai mici pătrate (RLS)

7.1.1 Algoritmul RLS

Este necesară o metodă de exprimare a inversei matricei $\mathbf{\Phi}(n)$, pornind de la relaţia de recurenţă.

Lema inversării matricei: Fiind date matricele $\mathbf{A}(N\times N)$, $\mathbf{B}(N\times N)$, $\mathbf{C}(N\times L)$, $\mathbf{D}(L\times L)$, în care **A**, **B**, **D**, sunt nesingulare şi satisfac relaţia:

$$\mathbf{A}=\mathbf{B}+\mathbf{C}\mathbf{D}\mathbf{C}^{H}$$

inversa matricei **A** este dată de

$$\mathbf{A}^{-1}=\mathbf{B}^{-1}-\mathbf{B}^{-1}\mathbf{C}\left(\mathbf{D}^{-1}+\mathbf{C}^{H}\mathbf{B}^{-1}\mathbf{C}\right)^{-1}\mathbf{C}^{H}\mathbf{B}^{-1}$$

Vom aplica această lemă pentru:

$$\mathbf{A}=\mathbf{\Phi}(n),\quad \mathbf{B}=\lambda\mathbf{\Phi}(n-1),\quad \mathbf{C}=\mathbf{x}(n),\quad \mathbf{D}=\mathbf{I}=1$$

$$\mathbf{\Phi}^{-1}(n)=\lambda^{-1}\mathbf{\Phi}^{-1}(n-1)-$$

$$-\lambda^{-1}\mathbf{\Phi}^{-1}(n-1)\mathbf{x}(n)\left(1+\mathbf{x}^{H}(n)\lambda^{-1}\mathbf{\Phi}^{-1}(n-1)\mathbf{x}(n)\right)^{-1}\mathbf{x}^{H}(n)\lambda^{-1}\mathbf{\Phi}^{-1}(n-1)$$

7.1 Metoda recursivă a celor mai mici pătrate (RLS)

7.1.1 Algoritmul RLS

$$\boldsymbol{\Phi}^{-1}(n) = \lambda^{-1}\boldsymbol{\Phi}^{-1}(n-1) -$$

$$-\lambda^{-1}\boldsymbol{\Phi}^{-1}(n-1)\mathbf{x}(n)\left(1+\mathbf{x}^{H}(n)\lambda^{-1}\boldsymbol{\Phi}^{-1}(n-1)\mathbf{x}(n)\right)^{-1}\mathbf{x}^{H}(n)\lambda^{-1}\boldsymbol{\Phi}^{-1}(n-1)$$

$$\boldsymbol{\Phi}^{-1}(n) = \frac{1}{\lambda}\boldsymbol{\Phi}^{-1}(n-1) - \frac{1}{\lambda^{2}}\frac{\boldsymbol{\Phi}^{-1}(n-1)\mathbf{x}(n)\mathbf{x}^{H}(n)\boldsymbol{\Phi}^{-1}(n-1)}{1+\frac{1}{\lambda}\mathbf{x}^{H}(n)\boldsymbol{\Phi}^{-1}(n-1)\mathbf{x}(n)}$$

$$\mathbf{P}(n) \hat{=} \boldsymbol{\Phi}^{-1}(n);$$

$$\mathbf{k}(n) \hat{=} \frac{1}{\lambda}\frac{\mathbf{P}(n-1)\mathbf{x}(n)}{1+\frac{1}{\lambda}\mathbf{x}^{H}(n)\mathbf{P}(n-1)\mathbf{x}(n)}$$

$$\mathbf{P}(n) = \frac{1}{\lambda}\mathbf{P}(n-1) - \frac{1}{\lambda}\mathbf{k}(n)\mathbf{x}^{H}(n)\mathbf{P}(n-1)$$

7.1 Metoda recursivă a celor mai mici pătrate (RLS)

7.1.1 Algoritmul RLS

$\mathbf{P}(n)$ este o matrice pătrată NxN;
$\mathbf{k}(n)$ este un vector Nx1, numit *vectorul câştig* (*Kalman*).
Ecuaţia de mai sus este o ecuaţie de tip *Riccati*.

$$\mathbf{k}(n)=\frac{1}{\lambda}\mathbf{P}(n-1)\mathbf{x}(n)-\frac{1}{\lambda}\mathbf{k}(n)\mathbf{x}^{H}(n)\mathbf{P}(n-1)\mathbf{x}(n)=$$

$$=\left(\frac{1}{\lambda}\mathbf{P}(n-1)-\frac{1}{\lambda}\mathbf{k}(n)\mathbf{x}^{H}(n)\mathbf{P}(n-1)\right)\mathbf{x}(n)$$

deci

$$\mathbf{k}(n)=\mathbf{P}(n)\mathbf{x}(n)$$
$$\mathbf{\Phi}(n)\mathbf{k}(n)=\mathbf{x}(n)$$

7.1 Metoda recursivă a celor mai mici pătrate (RLS)

7.1.1 Algoritmul RLS

Avem acum toate elementele pentru exprimarea lui $\mathbf{w}(n)$.

$$\mathbf{w}(n) = \mathbf{\Phi}^{-1}(n)\boldsymbol{\theta}(n) = \mathbf{P}(n)\boldsymbol{\theta}(n) = \lambda\mathbf{P}(n)\boldsymbol{\theta}(n-1) + \mathbf{P}(n)\mathbf{x}(n)d^*(n)$$

$$\mathbf{w}(n) = \mathbf{P}(n-1)\boldsymbol{\theta}(n-1) - \mathbf{k}(n)\mathbf{x}^H(n)\mathbf{P}(n-1)\boldsymbol{\theta}(n-1) + \mathbf{P}(n)\mathbf{x}(n)d^*(n) =$$

$$= \mathbf{\Phi}^{-1}(n-1)\boldsymbol{\theta}(n-1) - \mathbf{k}(n)\mathbf{x}^H(n)\mathbf{\Phi}^{-1}(n-1)\boldsymbol{\theta}(n-1) + \mathbf{k}(n)d^*(n)$$

$$\mathbf{w}(n) = \mathbf{w}(n-1) + \mathbf{k}(n)\left(d^*(n) - \mathbf{x}^H(n)\mathbf{w}(n-1)\right) = \mathbf{w}(n-1) + \mathbf{k}(n)\alpha^*(n)$$

unde

$$\mathbf{w}(n) = \mathbf{w}(n-1) + \mathbf{k}(n)\alpha^*(n)$$
$$\alpha(n) = d(n) - \mathbf{w}^H(n-1)\mathbf{x}(n)$$

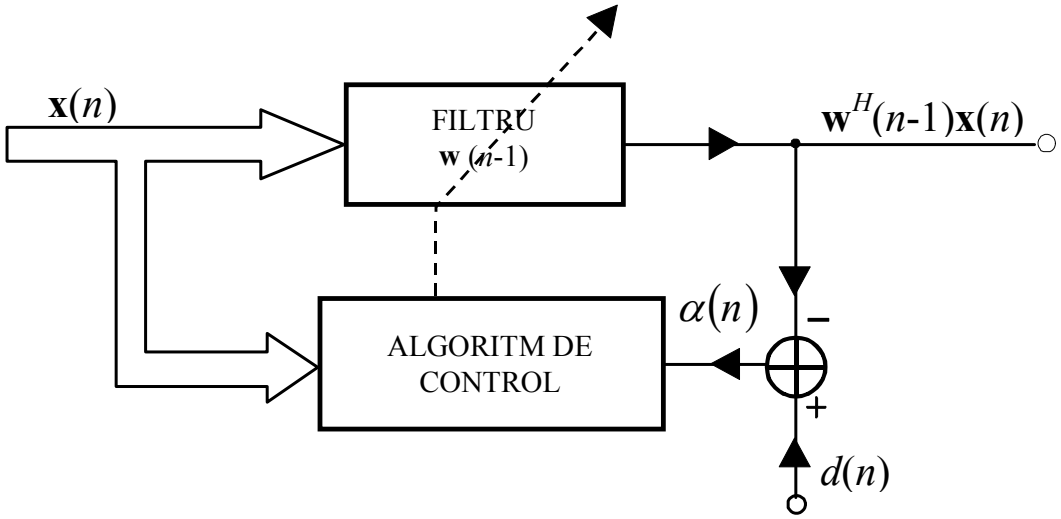
reprezintă eroarea estimării obţinute utilizând vechile valori $\mathbf{w}(n\text{-}1)$ ale coeficienţilor pentru noile date, deci *eroarea de estimare apriori*, numită şi *inovaţie*. Ea nu coincide cu eroarea aposteriori

$$e(n) = d(n) - \mathbf{w}^H(n)\mathbf{x}(n)$$

7.1 Metoda recursivă a celor mai mici pătrate (RLS)

7.1.1 Algoritmul RLS

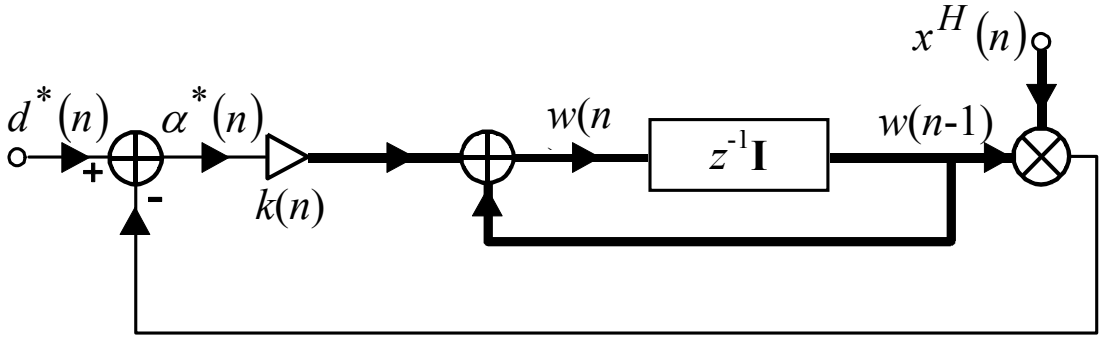
$$\mathbf{w}(n) = \mathbf{w}(n-1) + \mathbf{k}(n)\alpha^*(n)$$
$$\alpha(n) = d(n) - \mathbf{w}^H(n-1)\mathbf{x}(n)$$



7.1 Metoda recursivă a celor mai mici pătrate (RLS)

7.1.1 Algoritmul RLS

$$\mathbf{w}(n) = \mathbf{w}(n-1) + \mathbf{k}(n)\alpha^*(n)$$
$$\alpha(n) = d(n) - \mathbf{w}^H(n-1)\mathbf{x}(n)$$



7.1 Metoda recursivă a celor mai mici pătrate (RLS)

7.1.1 Algoritmul RLS

Relaţiile de ortogonalitate

$$\sum_{i=1}^{n} \lambda^{n-i} \mathbf{x}(i) e_{\min}^{*}(i) = \mathbf{0}$$

$$\sum_{i=1}^{n} \lambda^{n-i} y_0(i) e_{\min}^{*}(i) = 0$$

Consecinţă

$$E_{\min}(n) = J_{\min}(n) = E_d(n) - E_y(n)$$

În cazul de faţă:

$$E_y(n) = \sum_{i=1}^{n} \lambda^{n-i} y_o(i) y_o^{*}(i) = \sum_{i=1}^{n} \lambda^{n-i} \mathbf{w}^H(n) \mathbf{x}(i) \mathbf{x}^H(i) \mathbf{w}(n) =$$

$$= \mathbf{w}^H(n) \boldsymbol{\Phi}(n) \mathbf{w}(n) = \boldsymbol{\theta}^H(n) \mathbf{w}(n)$$

aşa încât rezultă o primă expresie utilă:

$$J_{\min}(n) = E_d(n) - \boldsymbol{\theta}^H(n) \mathbf{w}(n)$$

7.1 Metoda recursivă a celor mai mici pătrate (RLS)

7.1.1 Algoritmul RLS

$$E_d(n)=\sum_{i=1}^{n}\lambda^{n-i}|d(i)|^2=\lambda\sum_{i=1}^{n-1}\lambda^{n-1-i}|d(i)|^2+|d(n)|^2=\lambda E_d(n-1)+|d(n)|^2$$

Ţinând seama şi de relaţiile de recurenţă pentru $\boldsymbol{\theta}(n)$ şi $\mathbf{w}(n)$, rezultă

$$J_{\min}(n)=\lambda E_d(n-1)+|d(n)|^2-\left(\lambda\boldsymbol{\theta}^H(n-1)+d(n)\mathbf{x}^H(n)\right)\left(\mathbf{w}(n-1)+\mathbf{k}(n)\alpha^*(n)\right)=$$

$$=\lambda\left(E_d(n-1)-\boldsymbol{\theta}^H(n-1)\mathbf{w}(n-1)\right)+$$

$$+d(n)\left(d^*(n)-\mathbf{x}^H(n)\mathbf{w}(n-1)\right)-\left(\lambda\boldsymbol{\theta}^H(n-1)+d(n)\mathbf{x}^H(n)\right)\mathbf{k}(n)\alpha^*(n)=$$

$$=\lambda J_{\min}(n-1)+d(n)\alpha^*(n)-\boldsymbol{\theta}^H(n)\mathbf{k}(n)\alpha^*(n)$$

Ultimul termen al sumei se mai scrie

$$\boldsymbol{\theta}^H(n)\mathbf{k}(n)\alpha^*(n)=\boldsymbol{\theta}^H(n)\boldsymbol{\Phi}^{-1}(n)\mathbf{x}(n)\alpha^*(n)=\mathbf{w}^H(n)\mathbf{x}(n)\alpha^*(n)$$

Revenind la $J_{\min}(n)$ se obţine:

$$J_{\min}(n)=\lambda J_{\min}(n-1)+\alpha^*(n)\left(d(n)-\mathbf{w}^H(n)\mathbf{x}(n)\right)$$

$$J_{\min}(n)=\lambda J_{\min}(n-1)+\alpha^*(n)e(n)$$

7.1 Metoda recursivă a celor mai mici pătrate (RLS)

7.1.1 Algoritmul RLS

Iniţializare

$$\mathbf{P}(0)=\delta^{-1}\mathbf{I},\quad \mathbf{w}(n)=\mathbf{0}$$

for n=1,2,…

$$\mathbf{z}(n)=\mathbf{x}^{H}(n)\mathbf{P}(n-1)$$

(N^2 înmulţiri şi N (N-1) adunări)

$$\mathbf{k}(n)=\frac{1}{\lambda+\mathbf{z}(n)\mathbf{x}(n)}\mathbf{z}^{H}(n)$$

($2N$ înmulţiri şi N adunări)

$$\alpha(n)=d(n)-\mathbf{w}^{H}(n-1)\mathbf{x}(n)$$

(N înmulţiri şi N adunări)

$$\mathbf{w}(n)=\mathbf{w}(n-1)+\mathbf{k}(n)\alpha^{*}(n)$$

(N înmulţiri şi N adunări)

$$\mathbf{P}(n)=\frac{1}{\lambda}\left(\mathbf{P}(n-1)-\mathbf{k}(n)\mathbf{z}(n)\right)$$

($2N^2$ înmulţiri şi N^2 adunări)

end

7.1 Metoda recursivă a celor mai mici pătrate (RLS)

7.1.1 Algoritmul RLS

Complexitatea aritmetică a algoritmului

Iniţializarea algoritmului constă în impunerea valorilor **P**[0] şi **w**[0]. Uzual se ia **w**[0]=**0** şi

$$\mathbf{P}(0) = \delta^{-1}\mathbf{I}$$

unde δ este o valoare constantă mică, astfel încât matricea de autocorelaţie să nu rezulte singulară.

Complexitatea aritmetică este de $3N^2 + 4N$ înmulţiri şi $2N^2 + 2N$ adunări, deci numărul de operaţii este de forma O(N^2), (creşte cu N^2).

7.1 Metoda recursivă a celor mai mici pătrate (RLS)
7.1.2 Convergenţa algoritmului RLS

S-a arătat că matricea $\mathbf{\Phi}$ satisface ecuaţia cu diferenţe finite:

$$\mathbf{\Phi}(n) = \lambda \mathbf{\Phi}(n-1) + \mathbf{x}(n)\mathbf{x}^H(n)$$

cu condiţia iniţială

$$\mathbf{\Phi}(0) = \delta \mathbf{I}$$

Ca urmare, soluţia ecuaţiei cu diferenţe finite va fi în realitate:

$$\mathbf{\Phi}(n) = \lambda^n \mathbf{\Phi}(0) + \mathbf{\Phi}_0(n)$$

unde $\mathbf{\Phi}_0(n)$ este o soluţie particulară a ecuaţiei neomogene.
O asemenea soluţie este:

$$\mathbf{\Phi}_0(n) = \sum_{i=1}^{n} \lambda^{n-i} \mathbf{x}(i)\mathbf{x}^H(i)$$

şi rezultă

$$\mathbf{\Phi}(n) = \sum_{i=1}^{n} \lambda^{n-i} \mathbf{x}(i)\mathbf{x}^H(i) + \delta\lambda^n \mathbf{I}$$

7.1 Metoda recursivă a celor mai mici pătrate (RLS)
7.1.2 Convergenţa algoritmului RLS

Semnalul dorit este generat de un model liniar

$$d(i) = \mathbf{w}_0^H \mathbf{x}(i) + e_0(i)$$

unde $e_0(n)$reprezintă un zgomot alb cu valoare medie nulă şi varianţă σ^2.

Coeficienţii generaţi de filtrul adaptiv vor rezulta din:

$$\boldsymbol{\theta}(n) = \boldsymbol{\Phi}(n)\mathbf{w}(n)$$

care se poate scrie

$$\sum_{i=1}^{n} \lambda^{n-i}\mathbf{x}(i)d^*(i) = \sum_{i=1}^{n} \lambda^{n-i}\mathbf{x}(i)\mathbf{x}^H(i)\mathbf{w}_0 + \sum_{i=1}^{n} \lambda^{n-i}\mathbf{x}(i)e_0^*(i) =$$

$$= \sum_{i=1}^{n} \lambda^{n-i}\mathbf{x}(i)\mathbf{x}^H(i)\mathbf{w}(n) + \lambda^n \delta\mathbf{w}(n)$$

sau

$$\boldsymbol{\Phi}_0(n)(\mathbf{w}(n) - \mathbf{w}_0) = \sum_{i=1}^{n} \lambda^{n-i}\mathbf{x}(i)e_0^*(i) - \lambda^n \delta\mathbf{w}(n)$$

introducând vectorul eroare a coeficienţilor

$$\mathbf{c}(n) = \mathbf{w}(n) - \mathbf{w}_0$$

$$\mathbf{\Phi}_0(n)\mathbf{c}(n) = \sum_{i=1}^{n} \lambda^{n-i}\mathbf{x}(i)e_0^*(i) - \lambda^n \delta \mathbf{c}(n) - \lambda^n \delta \mathbf{w}_0$$

Se mediază relația de mai sus, observând că

$$\mathrm{E}\left\{\sum_{i=1}^{n} \lambda^{n-i}\mathbf{x}(i)e_0^*(i)\right\} = \sum_{i=1}^{n} \lambda^{n-i}\,\mathrm{E}\left\{\mathbf{x}(i)e_0^*(i)\right\} = \mathbf{0}$$

Admițând ipotezele de independență

$$\mathrm{E}\left\{\mathbf{\Phi}_0(n)\mathbf{c}(n)\right\} = \mathrm{E}\left\{\mathbf{\Phi}_0(n)\right\}\mathrm{E}\left\{\mathbf{c}(n)\right\}$$

$$\mathrm{E}\left\{\mathbf{\Phi}_0(n)\right\} = \sum_{i=1}^{n} \lambda^{n-i}\,\mathrm{E}\left\{\mathbf{x}(i)\mathbf{x}^H(i)\right\} = \mathbf{R}\sum_{i=1}^{n} \lambda^{n-i} = \frac{1-\lambda^n}{1-\lambda}\mathbf{R}$$

$$\left[\frac{1-\lambda^n}{1-\lambda}\mathbf{R}+\lambda^n\delta\mathbf{I}\right]\mathrm{E}\left\{\mathbf{c}(n)\right\}=-\lambda^n\delta\mathbf{w}_0$$

Pentru $\lambda\to 1$,

$$\frac{1-\lambda^n}{1-\lambda}\to n$$

$$\left[n\mathbf{R}+\lambda^n\delta\mathbf{I}\right]\mathrm{E}\left\{\mathbf{c}(n)\right\}=-\lambda^n\delta\mathbf{w}_0$$

Dacă

$$\delta \ll nr_{xx}(0)=nP_x$$

$$\mathrm{E}\left\{\mathbf{c}(n)\right\}\cong-\frac{1}{n}\delta\mathbf{R}^{-1}\mathbf{w}_0$$

Pentru $\lambda<1$ și n suficient de mare,

$$\left[\frac{1}{1-\lambda}\mathbf{R}+\lambda^n\delta\mathbf{I}\right]\mathrm{E}\{\mathbf{c}(n)\}=-\lambda^n\delta\mathbf{w}_0$$

$$\text{și} \qquad \delta \ll \lambda^{-n}P_x$$

$$\mathrm{E}\{\mathbf{c}(n)\}\cong-\lambda^n(1-\lambda)\delta\mathbf{R}^{-1}\mathbf{w}_0$$

În ambele cazuri

$$\mathrm{E}\{\mathbf{c}(n)\}\to\mathbf{0} \quad \text{cand} \quad n\to\infty$$

Observații

- Viteza de convergență depinde numai de λ.
- O valoare apropiată de 1 conduce la o viteză mică.
- Nu depinde de proprietățile statistice ale semnalelor implicate.

7.1 Metoda recursivă a celor mai mici pătrate (RLS)
7.1.2 Convergenţa algoritmului RLS

Convergenţa în medie pătratică a erorii apriori

Pornind de la definiţia acestei erori, pentru $d(n)$ dat de modelul autoregresiv:

$$\alpha(n) = e_0(n) - (\mathbf{w}(n-1) - \mathbf{w}_0)^H \mathbf{x}(n) = e_0(n) - \mathbf{c}^H(n-1)\mathbf{x}(n)$$

Vom evalua eroarea medie pătratică:

$$J'(n) = \mathrm{E}\left\{|\alpha(n)|^2\right\} = \mathrm{E}\left\{|e_0(n)|^2\right\} - \mathrm{E}\left\{\mathbf{x}^H(n)\mathbf{c}(n-1)e_0(n)\right\} -$$
$$- \mathrm{E}\left\{e_0^*(n)\mathbf{c}^H(n-1)\mathbf{x}(n)\right\} + \mathrm{E}\left\{\mathbf{x}^H(n)\mathbf{c}(n-1)\mathbf{c}^H(n-1)\mathbf{x}(n)\right\}$$

Pentru evaluarea mediilor ce intervin în această expresie vom face apel la ipotezele de independenţă: $\mathbf{w}(n\text{-}1)$ şi deci şi $\mathbf{c}(n\text{-}1)$ se presupun independente de $\mathbf{x}(n)$ sau de $e_o(n)$ aşa încât:

$$\mathrm{E}\left\{\mathbf{x}^H(n)\mathbf{c}(n-1)\mathbf{c}^H(n-1)\mathbf{x}(n)\right\} = \mathrm{E}\left\{\mathrm{tr}\left\{\mathbf{x}^H(n)\mathbf{c}(n-1)\mathbf{c}^H(n-1)\mathbf{x}(n)\right\}\right\} =$$
$$= \mathrm{E}\left\{\mathrm{tr}\left\{\mathbf{x}(n)\mathbf{x}^H(n)\mathbf{c}(n-1)\mathbf{c}^H(n-1)\right\}\right\} = \mathrm{tr}\left\{\mathrm{E}\left\{\mathbf{x}(n)\mathbf{x}^H(n)\mathbf{c}(n-1)\mathbf{c}^H(n-1)\right\}\right\} =$$
$$= \mathrm{tr}\left\{\mathrm{E}\left\{\mathbf{x}(n)\mathbf{x}^H(n)\right\}\mathrm{E}\left\{\mathbf{c}(n-1)\mathbf{c}^H(n-1)\right\}\right\} = \mathrm{tr}\{\mathbf{RC}(n-1)\}$$

7.1 Metoda recursivă a celor mai mici pătrate (RLS)
7.1.2 Convergenţa algoritmului RLS

Din aceleaşi motive şi având în vedere în plus principiul ortogonalităţii:

$$\mathrm{E}\left\{\mathbf{x}^H(n)\mathbf{c}(n-1)e_0(n)\right\} = \mathrm{E}\left\{e_0(n)\mathbf{x}^H(n)\right\}\mathrm{E}\left\{\mathbf{c}(n-1)\right\} = 0$$

$$\mathrm{E}\left\{e_0^*(n)\mathbf{c}^H(n-1)\mathbf{x}(n)\right\} = \mathrm{E}\left\{\mathbf{c}^H(n-1)\right\}\mathrm{E}\left\{e_0^*(n)\mathbf{x}(n)\right\} = 0$$

$$J'(n) = \sigma^2 + \mathrm{tr}\{\mathbf{RC}(n-1)\}$$

$$J'(n) \cong \sigma^2 + \frac{\sigma^2}{n-1}\mathrm{tr}\left\{\mathbf{RR}^{-1}\right\} = \sigma^2 + \frac{N\sigma^2}{n-1}$$

Evident,

$$\lim_{n\to\infty} J'(n) = \sigma^2 = \mathrm{E}\left\{\left|e_0(n)\right|^2\right\}$$

deci acest algoritm nu produce eroare medie pătratică în exces (dezadaptare) când operează cu un semnal staţionar. În general, convergenţa este mai rapidă decât în cazul algoritmilor de gradient, şi mai puţin dependentă de numărul condiţional al matricei de autocorelaţie, deci de caracterul datelor. Această îmbunătăţire a vitezei de convergenţă are loc numai dacă eroarea $e_o(n)$ este mică în comparaţie cu răspunsul dorit (raportul semnal/zgomot este mare). Preţul plătit constă într-o complexitate aritmetică mai ridicată, numerele de adunări şi înmulţiri crescând cu N^2.

7.2 Algoritm recursiv rapid pentru structuri transversale (FRLS)

FRLS (Fast Recursive Least Squares), sau FTF (fast transversal filter), pentru structuri transversale;
LFRLS (Lattice Fast Recursive Least Squares) sau LSL (least squares lattice).

7.2 Algoritm recursiv rapid pentru structuri transversale (FRLS)

7.2.1 Predicţia liniară directă în sensul LS

$$e_N^f(i) = x(i) - \mathbf{w}^H(n)\mathbf{x}_N(i-1)$$

$$\mathbf{x}_N^T(i-1) = \left[x(i-1), x(i-2), \cdots x(i-N)\right]$$

sau

$$\mathbf{a}_N(n) = \begin{bmatrix} 1 \\ -\mathbf{w}^*(n) \end{bmatrix}, \quad \mathbf{x}_{N+1}(i) = \begin{bmatrix} x(i) \\ \mathbf{x}_N(i-1) \end{bmatrix};$$

$$e_N^f(i) = \mathbf{a}_N^T(n)\mathbf{x}_{N+1}(i), \quad i = 1, \cdots, n$$

Funcţia cost:

$$J_N^f(n) = \sum_{i=1}^{n} \lambda^{n-i} \left| e_N^f(i) \right|^2$$

7.2 Algoritm recursiv rapid pentru structuri transversale (FRLS)

7.2.1 Predicţia liniară directă în sensul LS

Notăm:

$$\mathbf{\Phi}_N(n-1)=\sum_{i=1}^{n}\lambda^{n-i}\mathbf{x}_N(i-1)\mathbf{x}_N^H(i-1)=\sum_{i=1}^{n-1}\lambda^{n-1-i}\mathbf{x}_N(i)\mathbf{x}_N^H(i)$$

$$\boldsymbol{\theta}^f(n)=\sum_{i=1}^{n}\lambda^{n-i}\mathbf{x}_N(i-1)x^*(i)$$

în care în afara intervalului 1,...,*n*, datele s-au presupus nule, deci $\mathbf{x}(0)=\mathbf{0}$. Minimizarea funcţiei cost se obţine pentru:

$$\mathbf{\Phi}_N(n-1)\mathbf{w}(n)=\boldsymbol{\theta}^f(n)$$

Relaţiile

$$E_{\min}(n)=E_d(n)-\boldsymbol{\theta}^H(n)\mathbf{w}(n);\quad E_d(n)=\sum_{i=1}^{n}\lambda^{n-i}|d(i)|^2$$

se înlocuiesc cu:

$$J_{N\min}^f(n)=E_x^f(n)-\boldsymbol{\theta}^{fH}(n)\mathbf{w}(n);\quad E_x^f(n)=\sum_{i=1}^{n}\lambda^{n-i}|x(i)|^2$$

7.2 Algoritm recursiv rapid pentru structuri transversale (FRLS)

7.2.1 Predicţia liniară directă în sensul LS

Rezultă o ecuaţie normală extinsă, sub forma:

$$\mathbf{\Phi}_{N+1}(n)\mathbf{a}_N^*(n)=\begin{bmatrix} J_{N\min}^f \\ \mathbf{0} \end{bmatrix} \quad \text{unde} \quad \mathbf{\Phi}_{N+1}(n)=\begin{bmatrix} E_x^f(n) & \boldsymbol{\theta}^{fH}(n) \\ \boldsymbol{\theta}^f(n) & \mathbf{\Phi}_N(n-1) \end{bmatrix}$$

Deoarece faţă de formulele din paragraful precedent $\mathbf{x}(n)$ şi $\mathbf{\Phi}(n)$ se înlocuiesc cu $\mathbf{x}(n-1)$ şi $\mathbf{\Phi}_N(n-1)$, rezultă că şi ecuaţia ce permite reactualizarea coeficienţilor se poate scrie:

$$\mathbf{w}(n)=\mathbf{w}(n-1)+\mathbf{k}_N(n-1)\alpha_N^{f*}(n),$$

unde s-au introdus *vectorul câştig* (*Kalman*) pentru predicţia liniară directă

$$\mathbf{k}_N(n-1)=\mathbf{\Phi}_N^{-1}(n-1)\mathbf{x}_N(n-1)$$

şi eroarea apriori a predicţiei directe de ordinul N,

$$\alpha_N^f(n)=x(n)-\mathbf{w}^H(n-1)\mathbf{x}_N(n-1)=\left[1,-\mathbf{w}^H(n-1)\right]\begin{bmatrix} x(n) \\ \mathbf{x}_N(n-1) \end{bmatrix}=$$

$$=\mathbf{a}_N^T(n-1)\mathbf{x}_{N+1}(n)$$

7.2 Algoritm recursiv rapid pentru structuri transversale (FRLS)

7.2.1 Predicţia liniară directă în sensul LS

Pentru coeficienţii $\mathbf{a}_N(n)$ vom avea:

$$\mathbf{a}_N(n) = \mathbf{a}_N(n-1) - \begin{bmatrix} 0 \\ \mathbf{k}_N^*(n-1) \end{bmatrix} \alpha_N^f(n)$$

În fine, relaţia de recurenţă pentru evaluarea sumei ponderate a erorilor în cazul minimizării, va deveni:

$$J_{N\min}^f(n) = \lambda J_{N\min}^f(n-1) + \alpha_N^f(n) e_N^{f*}(n)$$

7.2 Algoritm recursiv rapid pentru structuri transversale (FRLS)

7.2.2 Predicţia liniară inversă în sensul LS

Reamintim că în acest caz eroarea era definită prin:

$$e_N^b(i) = x(i-N) - \mathbf{g}^H(n)\mathbf{x}_N(i)$$

deci faţă de modelul considerat în paragraful precedent, locul semnalului dorit este luat de $x(i-N)$. Vom defini vectorul $\mathbf{c}_N(n)$ format cu coeficienţii filtrului erorii de predicţie inversă,

$$\mathbf{c}_N(n) = \begin{bmatrix} -\mathbf{g}^*(n) \\ 1 \end{bmatrix}; \quad \mathbf{x}_{N+1}(i) = \begin{bmatrix} \mathbf{x}_N(i) \\ x(i-N) \end{bmatrix};$$

$$e_N^b(i) = \mathbf{c}_N^T(i)\mathbf{x}_{N+1}(i), \quad i = 1, \cdots, n$$

Funcţia cost ce trebuie minimizată va fi:

$$J_N^b(n) = \sum_{i=1}^{n} \lambda^{n-i} \left| e_N^b(i) \right|^2$$

Matricele $\mathbf{\Phi}$ şi $\boldsymbol{\theta}$ vor avea formele:

$$\mathbf{\Phi}_N(n) = \sum_{i=1}^{n} \lambda^{n-i} \mathbf{x}_N(i)\mathbf{x}_N^H(i); \quad \boldsymbol{\theta}^b(n) = \sum_{i=1}^{n} \lambda^{n-i} \mathbf{x}_N(i)x^*(i-N)$$

7.2 Algoritm recursiv rapid pentru structuri transversale (FRLS)

7.2.2 Predicţia liniară inversă în sensul LS

Coeficienţii optimi sunt daţi de ecuaţia normală

$$\mathbf{\Phi}_N(n)\mathbf{g}(n) = \boldsymbol{\theta}^b(n)$$

$$J^b_{N\,\min}(n) = E^b_x(n) - \boldsymbol{\theta}^{bH}(n)\mathbf{g}(n)$$

unde

$$E^b_x(n) = \sum_{i=1}^{n} \lambda^{n-i} \left| x(i-N) \right|^2 = \sum_{i=1}^{n-N} \lambda^{n-N-i} \left| x(i) \right|^2$$

în care s-a avut în vedere ipoteza că $x(i) = 0$ pentru $i \le 0$.

Ecuaţia normală extinsă:

$$\mathbf{\Phi}_{N+1}(n)\mathbf{c}^*_N(n) = \begin{bmatrix} 0 \\ J^b_{N\,\min}(n) \end{bmatrix}$$

unde s-a introdus

$$\mathbf{\Phi}_{N+1}(n) = \begin{bmatrix} \mathbf{\Phi}_N(n) & \boldsymbol{\theta}^b(n) \\ \boldsymbol{\theta}^{bH}(n) & E^b_x(n) \end{bmatrix}$$

7.2 Algoritm recursiv rapid pentru structuri transversale (FRLS)
7.2.2 Predicţia liniară inversă în sensul LS

Reactualizarea coeficienţilor se va face deci conform relaţiei:

$$\mathbf{g}(n) = \mathbf{g}(n-1) + \mathbf{k}_N(n)\alpha_N^{b*}(n)$$

în care vectorul câştig este:

$$\mathbf{k}_N(n) = \mathbf{\Phi}_N^{-1}(n)\mathbf{x}_N(n)$$

definiţie care concordă formal cu cea obţinută în cazul predicţiei directe.

7.2 Algoritm recursiv rapid pentru structuri transversale (FRLS)
7.2.2 Predicţia liniară inversă în sensul LS

Eroarea apriori a predicţiei liniare inverse de ordinul N este:

$$\alpha_N^b(n) = x(n-N) - \mathbf{g}^H(n-1)\mathbf{x}_N(n) = \left[-\mathbf{g}^H(n-1), 1\right]\begin{bmatrix}\mathbf{x}_N(n)\\ x(n-N)\end{bmatrix} = \mathbf{c}_N^T(n-1)\mathbf{x}_{N+1}(n)$$

Pentru coeficienţii filtrului erorii de predicţie, relaţia de recurenţă va fi deci:

$$\mathbf{c}_N(n) = \mathbf{c}_N(n-1) - \begin{bmatrix}\mathbf{k}_N^*(n)\\ 0\end{bmatrix}\alpha_N^b(n)$$

Valoarea minimizată a funcţiei cost se va putea calcula cu relaţia de recurenţă:

$$J_{N\min}^b(n) = \lambda J_{N\min}^b(n-1) + \alpha_N^b(n)e_N^{b*}(n)$$

7.2 Algoritm recursiv rapid pentru structuri transversale (FRLS)
7.2.3 Relaţii de recurenţă pentru vectorul câştig extins

Reamintim:

$$\mathbf{k}_N(n) = \boldsymbol{\Phi}_N^{-1}(n)\mathbf{x}_N(n)$$

Vom defini un ***vector câştig extins*** prin relaţia:

$$\mathbf{k}_{N+1}(n) = \boldsymbol{\Phi}_{N+1}^{-1}(n)\mathbf{x}_{N+1}(n)$$

Să arătăm mai întâi că inversa matricei $\boldsymbol{\Phi}$ extinse este dată de:

$$\boldsymbol{\Phi}_{N+1}^{-1}(n) = \begin{bmatrix} 0 & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Phi}_N^{-1}(n-1) \end{bmatrix} + \frac{1}{J_{N\min}^{f}(n)} \mathbf{a}_N^{*}(n)\mathbf{a}_N^{T}(n)$$

7.2 Algoritm recursiv rapid pentru structuri transversale (FRLS)
7.2.3 Relaţii de recurenţă pentru vectorul câştig extins

$$\mathbf{\Phi}_{N+1}^{-1}(n)=\begin{bmatrix}0 & \mathbf{0}\\ \mathbf{0} & \mathbf{\Phi}_N^{-1}(n-1)\end{bmatrix}+\frac{1}{J_{N\min}^{f}(n)}\mathbf{a}_N^{*}(n)\mathbf{a}_N^{T}(n)$$

Vom porni de la forma partiţionată a matricei respective

$$\mathbf{\Phi}_{N+1}(n)\begin{bmatrix}0 & \mathbf{0}\\ \mathbf{0} & \mathbf{\Phi}_N^{-1}(n-1)\end{bmatrix}=\begin{bmatrix}E_x^{f}(n) & \boldsymbol{\theta}^{fH}(n)\\ \boldsymbol{\theta}^{f}(n) & \mathbf{\Phi}_N(n-1)\end{bmatrix}\begin{bmatrix}0 & \mathbf{0}\\ \mathbf{0} & \mathbf{\Phi}_N^{-1}(n-1)\end{bmatrix}=$$

$$=\begin{bmatrix}0 & \boldsymbol{\theta}^{fH}(n)\mathbf{\Phi}_N^{-1}(n-1)\\ \mathbf{0} & \mathbf{I}\end{bmatrix}=\begin{bmatrix}0 & \mathbf{w}^{H}(n)\\ \mathbf{0} & \mathbf{I}\end{bmatrix}$$

7.2 Algoritm recursiv rapid pentru structuri transversale (FRLS)

7.2.3 Relaţii de recurenţă pentru vectorul câştig extins

$$\mathbf{\Phi}_{N+1}^{-1}(n)=\begin{bmatrix}0 & \mathbf{0}\\ \mathbf{0} & \mathbf{\Phi}_N^{-1}(n-1)\end{bmatrix}+\frac{1}{J_{N\min}^{f}(n)}\mathbf{a}_N^{*}(n)\mathbf{a}_N^{T}(n)$$

$$\mathbf{\Phi}_{N+1}(n)\frac{1}{J_{N\min}^{f}(n)}\mathbf{a}_N^{*}(n)\mathbf{a}_N^{T}(n)=\frac{1}{J_{N\min}^{f}(n)}\begin{bmatrix}J_{N\min}^{f}(n)\\ \mathbf{0}\end{bmatrix}\mathbf{a}_N^{T}(n)=$$

$$=\begin{bmatrix}1\\ \mathbf{0}\end{bmatrix}\left[1,-\mathbf{w}^{H}(n)\right]=\begin{bmatrix}1 & -\mathbf{w}^{H}(n)\\ \mathbf{0} & \mathbf{0}\end{bmatrix}$$

şi se adună membru cu membru cele două egalităţi obţinute. Se obţine matricea unitate.

7.2 Algoritm recursiv rapid pentru structuri transversale (FRLS)
7.2.3 Relaţii de recurenţă pentru vectorul câştig extins

Consecinţă:

$$\mathbf{k}_{N+1}(n)=\begin{bmatrix}0 & \mathbf{0}\\ \mathbf{0} & \mathbf{\Phi}_{N+1}^{-1}(n-1)\end{bmatrix}\begin{bmatrix}x(n)\\ \mathbf{x}_N(n-1)\end{bmatrix}+\frac{1}{J_{N\min}^{f}(n)}\mathbf{a}_N^{*}(n)\mathbf{a}_N^{T}(n)\mathbf{x}_{N+1}(n)=$$

$$=\begin{bmatrix}0\\ \mathbf{k}_N(n-1)\end{bmatrix}+\mathbf{a}_N^{*}(n)\frac{e_N^{f}(n)}{J_{N\min}^{f}(n)}$$

În mod absolut asemănător

$$\mathbf{\Phi}_{N+1}^{-1}(n)=\begin{bmatrix}\mathbf{\Phi}_N^{-1}(n) & \mathbf{0}\\ \mathbf{0} & 0\end{bmatrix}+\frac{1}{J_{N\min}^{b}(n)}\mathbf{c}_N^{*}(n)\mathbf{c}_N^{T}(n)$$

şi

$$\mathbf{k}_{N+1}(n)=\begin{bmatrix}\mathbf{k}_N(n)\\ 0\end{bmatrix}+\mathbf{c}_N^{*}(n)\frac{e_N^{b}(n)}{J_{N\min}^{b}(n)}$$

7.2 Algoritm recursiv rapid pentru structuri transversale (FRLS)
7.2.4 Factorul de conversie

Din definiţia vectorului câştig

$$\mathbf{k}_N(n) = \mathbf{\Phi}_N^{-1}(n)\mathbf{x}_N(n)$$

rezultă că acesta poate fi privit drept setul de coeficienţi optimi, pentru un filtru transversal de ordinul *N*, având drept intrare secvenţa $x(1), x(1), \cdots, x(n)$ şi drept semnal dorit,

$$d(i) = \begin{cases} 1, & i = n \\ 0, & i = 1, \cdots, n-1 \end{cases}$$

Într-adevăr, ecuaţia este de forma:

$$\mathbf{k}_N(n) = \mathbf{\Phi}_N^{-1}(n)\boldsymbol{\theta}(n), \quad \text{unde} \quad \boldsymbol{\theta}(n) = \sum_{i=1}^{n} \lambda^{n-i}\mathbf{x}_N(i)d^*(i) = \mathbf{x}_N(n)$$

7.2 Algoritm recursiv rapid pentru structuri transversale (FRLS)
7.2.4 Factorul de conversie

Eroarea de estimare, numită şi ***factor de conversie***,

$$\gamma_N(n) = d(n) - \mathbf{k}_N^H(n)\mathbf{x}_N(n) = 1 - \mathbf{x}_N^H(n)\boldsymbol{\Phi}_N^{-1}(n)\mathbf{x}_N(n)$$

reprezintă ieşirea filtrului din figură. Această eroare este evident reală şi având în vedere definiţia vectorului câştig,

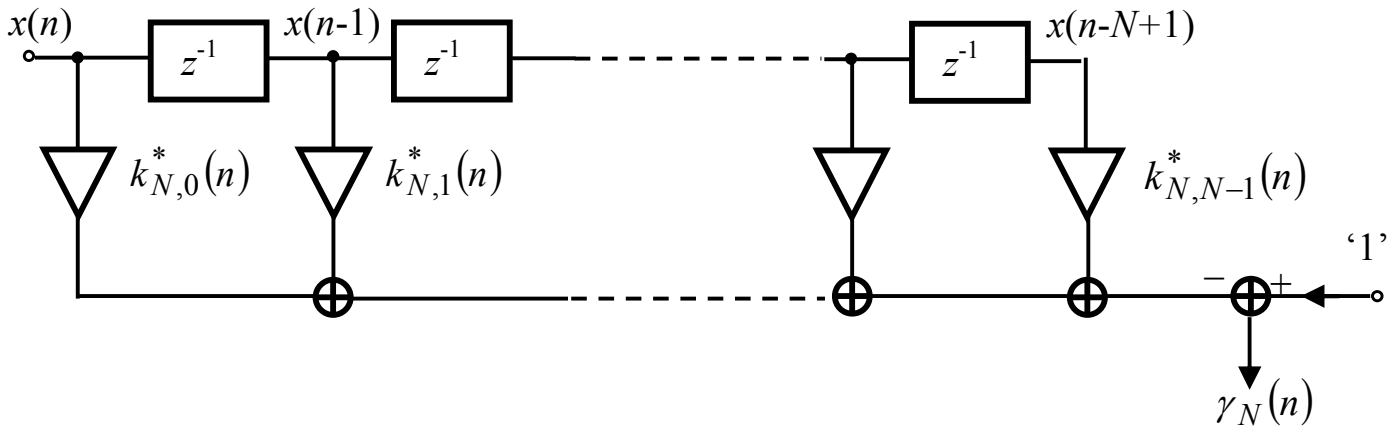
$$\mathbf{P}(n) \triangleq \boldsymbol{\Phi}^{-1}(n);$$

$$\mathbf{k}(n) \triangleq \frac{1}{\lambda}\frac{\mathbf{P}(n-1)\mathbf{x}(n)}{1+\frac{1}{\lambda}\mathbf{x}^H(n)\mathbf{P}(n-1)\mathbf{x}(n)}$$

mai poate fi scrisă:

$$\gamma_N(n) = \frac{1}{1+\lambda^{-1}\mathbf{x}_N^H(n)\Phi_N^{-1}(n-1)\mathbf{x}_N(n)}$$

7.2 Algoritm recursiv rapid pentru structuri transversale (FRLS)
7.2.4 Factorul de conversie



Evident,

$$\mathbf{x}_N^H(n)\boldsymbol{\Phi}_N^{-1}(n-1)\mathbf{x}(n) \geq 0$$

$$0 \leq \gamma_N(n) \leq 1$$

7.2 Algoritm recursiv rapid pentru structuri transversale (FRLS)
7.2.4 Factorul de conversie

Sunt posibile trei exprimări utile pentru $\gamma_N(n)$.

a) Pentru estimarea recursivă în sensul celor mai mici pătrate,

$$\gamma_N(n) = \frac{e_N(n)}{\alpha_N(n)}$$

Într-adevăr, înmulţind relaţia

$$\mathbf{w}^H(n) = \mathbf{w}^H(n-1) + \alpha_N(n)\mathbf{k}_N^H(n)$$

la dreapta cu $\mathbf{x}_N(n)$, se obţin succesiv:

$$\mathbf{w}^H(n)\mathbf{x}_N(n) = \mathbf{w}^H(n-1)\mathbf{x}_N(n) + \alpha_N(n)\mathbf{k}_N^H(n)\mathbf{x}_N(n);$$

$$d(n) - e_N(n) = d(n) - \alpha_N(n) + \alpha_N(n)(1 - \gamma_N(n));$$

$$e_N(n) = \alpha_N(n)\gamma_N(n)$$

7.2 Algoritm recursiv rapid pentru structuri transversale (FRLS)
7.2.4 Factorul de conversie

a) În cazul predicţiei directe:

$$\gamma_N(n-1)=\frac{e_N^f(n)}{\alpha_N^f(n)}$$

Pentru demonstraţie vom porni de la

$$\mathbf{a}_N(n)=\mathbf{a}_N(n-1)-\begin{bmatrix}0\\ \mathbf{k}_N^*(n-1)\end{bmatrix}\alpha_N^f(n) \qquad (3)$$

pe care o transpunem şi o înmulţim la dreapta cu $\mathbf{x}_{N+1}(n)$, obţinând succesiv:

$$\mathbf{a}_N^T(n)\mathbf{x}_{N+1}(n)=\mathbf{a}_N^T(n-1)\mathbf{x}_{N+1}(n)-\alpha_N^f(n)\left[0,\mathbf{k}_N^H(n-1)\right]\mathbf{x}_{N+1}(n);$$

$$e_N^f(n)=\alpha_N^f(n)-\alpha_N^f(n)\mathbf{k}_N^H(n-1)\mathbf{x}_N(n-1)=$$

$$=\alpha_N^f(n)-\alpha_N^f(n)(1-\gamma_N(n-1))=\alpha_N^f(n)\gamma_N(n-1)$$

7.2 Algoritm recursiv rapid pentru structuri transversale (FRLS)
7.2.4 Factorul de conversie

c). În cazul predicţiei inverse:

Demonstraţia se face în mod asemănător cazului b), pornind de la

$$\gamma_N(n) = \frac{e_N^b(n)}{\alpha_N^b(n)}$$

$$\mathbf{c}_N(n) = \mathbf{c}_N(n-1) - \begin{bmatrix} \mathbf{k}_N^*(n) \\ 0 \end{bmatrix} \alpha_N^b(n) \qquad (5)$$

$$\mathbf{c}_N^T(n)\mathbf{x}_{N+1}(n) = \mathbf{c}_N^T(n-1)\mathbf{x}_{N+1}(n) - \alpha_N^b(n)\left[\mathbf{k}_N^H(n), 0\right]\mathbf{x}_{N+1}(n);$$

$$e_N^b(n) = \alpha_N^b(n)\mathbf{k}_N^H(n)\mathbf{x}_N(n) =$$

$$= \alpha_N^b(n) - \alpha_N^b(n)(1 - \gamma_N(n)) = \alpha_N^b(n)\gamma_N(n)$$

7.2 Algoritm recursiv rapid pentru structuri transversale (FRLS)

7.2.5 Algoritm adaptiv rapid pentru filtre transversale (FTF)

Algoritmul ce va fi prezentat în continuare are drept punct de plecare formulele care au fost demonstrate mai înainte şi este cunoscut în literatură prin denumirea prescurtată FTF (Fast Transversal Filter).
Filtrul adaptiv se consideră a fi un ansamblu format din 4 filtre:

- un filtru cu coeficienţii $a_N(n)$ pentru calculul erorii predicţiei directe, $e_N^f(n)$;
- un filtru cu coeficienţii $c_N(n)$ pentru calculul erorii predicţiei inverse, $e_N^b(n)$;
- un filtru cu coeficienţii $k_N(n)$ ce generează factorul de conversie, $\gamma_N(n)$;
- filtrul propriu-zis cu coeficienţii $w_n^*(n-1)$.

7.3 ALGORITMI RECURSIVI BAZAŢI PE DESCOMPUNEREA QR. STRUCTURI SISTOLICE

7.3.1 Formularea recursivă a descompunerii QR

Se urmăreşte reactualizarea matricelor **Q** şi **R** la fiecare iteraţie, în funcţie de valorile din iteraţia precedentă şi de noile date.

Vom considera că datele sunt nule pentru $n<0$, (metoda ferestrei anterioare) aşa încât matricea de date, de dimensiuni $N \times n$, este:

$$\mathbf{A}^H = [\mathbf{x}(1), \mathbf{x}(2), \cdots, \mathbf{x}(N), \cdots \mathbf{x}(n)] =$$

$$= \begin{bmatrix} x(1) & x(2) & \cdots & x(N) & \cdots & x(n) \\ 0 & x(1) & \cdots & x(N-1) & \cdots & x(n-1) \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & x(1) & \cdots & x(n-N+1) \end{bmatrix}$$

7.3 ALGORITMI RECURSIVI BAZAŢI PE DESCOMPUNEREA QR. STRUCTURI SISTOLICE

7.3.1 Formularea recursivă a descompunerii QR

Introducând vectorii erorilor şi semnalului dorit:

$$\mathbf{e}^H(n) = [e(1), e(2), \cdots, e(n)]$$

$$\mathbf{d}^H(n) = [d(1), d(2), \cdots, d(n)]$$

$$\mathbf{e}(n) = \mathbf{d}(n) - \mathbf{A}(n)\mathbf{w}(n)$$

eroarea pătratică ponderată poate fi exprimată prin:

$$J(n) = \sum_{i=1}^{n} \lambda^{n-i} |e(i)|^2 = \mathbf{e}^H(n)\mathbf{\Lambda}(n)\mathbf{e}(n) = \left\| \mathbf{\Lambda}^{1/2}(n)\mathbf{e}(n) \right\|^2$$

$$\mathbf{\Lambda}(n) = \text{diag}\left\{\lambda^{n-1}, \lambda^{n-2}, \cdots, \lambda, 1\right\}$$

Fie $\mathbf{Q}(n)$ o matrice unitară.

$$J(n) = \left\| \mathbf{Q}(n)\mathbf{\Lambda}^{1/2}(n)\mathbf{e}(n) \right\|^2 =$$

$$= \left\| \mathbf{Q}(n)\mathbf{\Lambda}^{1/2}(n)\mathbf{d}(n) - \mathbf{Q}(n)\mathbf{\Lambda}^{1/2}(n)\mathbf{A}(n)\mathbf{w}(n) \right\|^2$$

7.3 ALGORITMI RECURSIVI BAZAŢI PE DESCOMPUNEREA QR. STRUCTURI SISTOLICE

7.3.1 Formularea recursivă a descompunerii QR

Pentru orice $n>N$, se poate alege matricea $\mathbf{Q}(n)$ de dimensiuni $n \times n$ aşa încât:

$$\mathbf{Q}(n)\boldsymbol{\Lambda}^{1/2}(n)\mathbf{A}(n) = \begin{bmatrix} \mathbf{R}(n) \\ \mathbf{0} \end{bmatrix}, \quad n > N$$

unde $\mathbf{R}(n)$ este o matrice NxN superior triunghiulară. Vom presupune matricea $\mathbf{Q}$ partiţionată sub forma:

$$\mathbf{Q}(n) = \begin{bmatrix} \mathbf{F}(n) \\ \mathbf{S}(n) \end{bmatrix}, \quad n > N$$

unde matricea $\mathbf{F}(n)$ este formată cu primele N linii şi vom nota:

7.3 ALGORITMI RECURSIVI BAZAŢI PE DESCOMPUNEREA QR. STRUCTURI SISTOLICE

7.3.1 Formularea recursivă a descompunerii QR

$$\mathbf{p}(n) = \mathbf{F}(n)\mathbf{\Lambda}^{1/2}(n)\mathbf{d}(n); \quad \mathbf{v}(n) = \mathbf{S}(n)\mathbf{\Lambda}^{1/2}(n)\mathbf{d}(n), \quad n > N$$

aşa încât:

$$\mathbf{Q}(n)\mathbf{\Lambda}^{1/2}(n)\mathbf{e}(n) = \begin{bmatrix} \mathbf{p}(n) - \mathbf{R}(n)\mathbf{w}(n) \\ \mathbf{v}(n) \end{bmatrix}$$

$\mathbf{w}(n)$ va trebui astfel ales încât să minimizeze norma vectorului de mai sus, deci rezultă:

$$\mathbf{R}(n)\mathbf{w}(n) = \mathbf{p}(n)$$

şi rămâne:

$$J_{\min}(n) = \|\mathbf{v}(n)\|^2$$

7.3 ALGORITMI RECURSIVI BAZAŢI PE DESCOMPUNEREA QR. STRUCTURI SISTOLICE

7.3.1 Formularea recursivă a descompunerii QR

Ecuaţia normală devine

$$\mathbf{\Phi}(n)\mathbf{w}(n) = \boldsymbol{\theta}(n) \rightarrow \ \mathbf{R}(n)\mathbf{w}(n) = \mathbf{p}(n)$$

având însă avantajul că este un sistem de N ecuaţii cu N necunoscute, cu matricea $\mathbf{R}(n)$ triunghiulară, ceea ce permite calculul simplu al coeficienţilor $\mathbf{w}(n)$ prin substituţii succesive.

Să presupunem că s-a realizat descompunerea QR pentru n-1:

$$\mathbf{Q}(n-1)\mathbf{\Lambda}^{1/2}(n-1)\mathbf{A}(n-1) = \begin{bmatrix} \mathbf{R}(n-1) \\ \mathbf{0} \end{bmatrix}$$

7.3 ALGORITMI RECURSIVI BAZAŢI PE DESCOMPUNEREA QR. STRUCTURI SISTOLICE

7.3.1 Formularea recursivă a descompunerii QR

La momentul n, la matricea $\mathbf{A}(n\text{-}1)$ se mai adaugă o linie:

$$\mathbf{A}(n)=\begin{bmatrix}\mathbf{A}(n-1)\\ \mathbf{x}^H(n)\end{bmatrix},\quad \mathbf{x}^T(n)=\left[x(n),x(n-1),\cdots,x(n-N+1)\right]$$

şi am dori să determinăm $\mathbf{Q}(n)$ pornind de la $\mathbf{Q}(n\text{-}1)$ şi $\mathbf{x}(n)$. Să introducem mai întâi o matrice de dimensiuni nxn,

$$\mathbf{Q}'(n-1)=\begin{bmatrix}\mathbf{Q}(n-1) & \mathbf{0}_{n-1}\\ \mathbf{0}_{n-1}^T & 1\end{bmatrix}$$

Evident:

$$\mathbf{\Lambda}(n)=\begin{bmatrix}\lambda\mathbf{\Lambda}(n-1) & \mathbf{0}_{n-1}\\ \mathbf{0}_{n-1}^T & 1\end{bmatrix}$$

7.3 ALGORITMI RECURSIVI BAZAŢI PE DESCOMPUNEREA QR. STRUCTURI SISTOLICE

7.3.1 Formularea recursivă a descompunerii QR

$$\mathbf{Q}'(n-1)\boldsymbol{\Lambda}^{1/2}(n)=\begin{bmatrix}\mathbf{Q}(n-1) & \mathbf{0}_{n-1}\\ \mathbf{0}_{n-1}^T & 1\end{bmatrix}\begin{bmatrix}\lambda^{1/2}\boldsymbol{\Lambda}^{1/2}(n-1) & \mathbf{0}_{n-1}\\ \mathbf{0}_{n-1}^T & 1\end{bmatrix}=$$

$$=\begin{bmatrix}\lambda^{1/2}\mathbf{Q}(n-1)\boldsymbol{\Lambda}^{1/2}(n-1) & 0\\ 0 & 1\end{bmatrix}$$

Se efectuează produsul:

$$\mathbf{Q}'(n-1)\boldsymbol{\Lambda}^{1/2}(n)\mathbf{A}(n)=\begin{bmatrix}\lambda^{1/2}\mathbf{Q}(n-1)\boldsymbol{\Lambda}^{1/2}(n-1)\mathbf{A}(n-1)\\ \mathbf{x}^H(n)\end{bmatrix}=\begin{bmatrix}\lambda^{1/2}\mathbf{R}(n-1)\\ \mathbf{0}\\ \mathbf{x}^H(n)\end{bmatrix}$$

În membrul drept s-a obţinut o matrice de dimensiune nxN, $n>N$ care are proprietăţile:

- primele N linii formează o matrice superior triunghiulară, $\mathbf{R}(n\text{-}1)$;
- următoarele n-N-1 linii conţin numai zerouri;
- ultima linie conţine elemente în general nenule.

Vom numi o asemenea matrice *parţial triunghiulară*.

7.3 ALGORITMI RECURSIVI BAZAŢI PE DESCOMPUNEREA QR. STRUCTURI SISTOLICE

7.3.1 Formularea recursivă a descompunerii QR

Utilizând oricare din procedeele de triunghiularizare cunoscute se pot anula, unul câte unul, elementele ultimei linii

$$\mathbf{T}(n)\mathbf{Q}'(n-1)\mathbf{\Lambda}^{1/2}(n)\mathbf{A}(n) = \mathbf{Q}(n)\mathbf{\Lambda}^{1/2}(n)\mathbf{A}(n) = \begin{bmatrix} \mathbf{R}(n) \\ \mathbf{0} \end{bmatrix}$$

de unde

$$\begin{bmatrix} \mathbf{R}(n) \\ \mathbf{0} \end{bmatrix} = \mathbf{T}(n) \begin{bmatrix} \lambda^{1/2}\mathbf{R}(n-1) \\ \mathbf{0} \\ \mathbf{x}^H(n) \end{bmatrix}$$

Noua valoare a lui $\mathbf{Q}(n)$ este deci:

$$\mathbf{Q}(n) = \mathbf{T}(n)\mathbf{Q}'(n-1)$$

7.3 ALGORITMI RECURSIVI BAZAŢI PE DESCOMPUNEREA QR. STRUCTURI SISTOLICE

7.3.1 Formularea recursivă a descompunerii QR

Pe de altă parte:

$$\mathbf{Q}(n)\boldsymbol{\Lambda}^{1/2}(n)\mathbf{d}(n)=\mathbf{T}(n)\begin{bmatrix}\mathbf{Q}(n-1) & \mathbf{0}_{n-1}\\ \mathbf{0}_{n-1}^{T} & 1\end{bmatrix}\begin{bmatrix}\lambda^{1/2}\boldsymbol{\Lambda}^{1/2}(n-1) & \mathbf{0}_{n-1}\\ \mathbf{0}_{n-1}^{T} & 1\end{bmatrix}\begin{bmatrix}\mathbf{d}(n-1)\\ d^{*}(n)\end{bmatrix}=$$

$$=\mathbf{T}(n)\begin{bmatrix}\lambda^{1/2}\mathbf{Q}(n-1)\boldsymbol{\Lambda}^{1/2}(n-1)\mathbf{d}(n-1)\\ d^{*}(n)\end{bmatrix}$$

sau:

$$\begin{bmatrix}\mathbf{p}(n)\\ \mathbf{v}(n)\end{bmatrix}=\mathbf{T}(n)\begin{bmatrix}\lambda^{1/2}\mathbf{p}(n-1)\\ \lambda^{1/2}\mathbf{v}(n-1)\\ d^{*}(n)\end{bmatrix}$$

deci $\mathbf{p}(n)$, $\mathbf{v}(n)$ se pot reactualiza aplicând aceiaşi transformare $\mathbf{T}(n)$ ca şi pentru reactualizarea lui $\mathbf{R}(n)$.

7.3 ALGORITMI RECURSIVI BAZAŢI PE DESCOMPUNEREA QR. STRUCTURI SISTOLICE

7.3.1 Formularea recursivă a descompunerii QR

Algoritmul va începe cu o etapă de iniţializare, pentru n=0,1,...,N pornind de la $\mathbf{R}(0)=\mathbf{0}$ şi $\mathbf{p}(0)=\mathbf{0}$ şi având în vedere caracterul triunghiular al matricei de date **A** în acest caz. Apoi, cu formulele deduse se reactualizează $\mathbf{R}(n)$, $\mathbf{p}(n)$ şi $\mathbf{v}(n)$. Anularea ultimei linii a matricei parţial triunghiulare ce apare în procesul recursiv se poate realiza utilizând una din metodele de triangularizare prezentate în capitolul precedent. Se rezolvă sistemul

$$\mathbf{R}(n)\mathbf{w}(n) = \mathbf{p}(n)$$

prin substituţii, având în vedere forma triunghiulară a matricei $\mathbf{R}(n)$, şi se calculează eroarea:

$$J_{\min}(n) = \|\mathbf{v}(n)\|^2$$

7.3 ALGORITMI RECURSIVI BAZAŢI PE DESCOMPUNEREA QR. STRUCTURI SISTOLICE

7.3.2 Algoritm recursiv bazat pe rotaţii Givens

Fie o matrice parţial triunghiulară **Y** de dimensiune $n \times N$. Ne propunem ca prin înmulţiri cu matrice unitare să eliminăm elementele ultimei linii fără a afecta primele două proprietăţi ale unei asemenea matrice. Să considerăm un element k al ultimei linii şi să introducem matricea de dimensiuni nxn partiţionată:

$$\mathbf{T}^{(k)}(n) = \begin{bmatrix} \mathbf{I}_{k-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{D} \end{bmatrix}; \quad \mathbf{D} = \begin{bmatrix} c & 0 & \cdots & 0 & s^* \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ -s & 0 & \cdots & 0 & c \end{bmatrix}$$

unde $\mathbf{I}_{k-1}$ este matricea unitate de dimensiune k-1 iar **D** este o matrice $(n-k+1)\times(n-k+1)$. Parametrii c şi s sunt de forma:

$$c = \cos\theta, \quad s = \mathrm{e}^{\mathrm{j}\varphi}\sin\theta$$

aşa încât:

$$c^2 + |s|^2 = 1$$

7.3 ALGORITMI RECURSIVI BAZAŢI PE DESCOMPUNEREA QR. STRUCTURI SISTOLICE

7.3.2 Algoritm recursiv bazat pe rotaţii Givens

Se poate verifica faptul că $\mathbf{T}^{(k)}(n)$ este o matrice unitară. Prin înmulţirea la stânga a lui $\mathbf{Y}$ cu $\mathbf{T}^{(k)}(n)$, elementele y_{kk} şi y_{nk} se transformă după cum urmează:

$$y_{kk} \rightarrow c y_{kk} + s^{*} y_{nk}$$

$$y_{nk} \rightarrow -s y_{kk} + c y_{nk}$$

$$-s y_{kk} + c y_{nk} = 0$$

rezultă:

$$c = \frac{|y_{kk}|}{\sqrt{|y_{kk}|^2 + |y_{nk}|^2}}; \quad s = \frac{y_{nk}}{y_{kk}} c$$

7.3 ALGORITMI RECURSIVI BAZAŢI PE DESCOMPUNEREA QR. STRUCTURI SISTOLICE

7.3.2 Algoritm recursiv bazat pe rotaţii Givens

Observaţii

- Operaţia respectivă modifică numai linia k şi ultima linie a matricei **Y.**
- Să presupunem că în paşii precedenţi s-au anulat primele k-1 elemente ale ultimei linii, iar matricea este parţial triunghiulară, deci

$$y_{ki} = 0; \quad y_{ni} = 0, \quad \text{pentru} \quad i < k$$

Rezultă că după transformare:

$$\begin{aligned} y_{ki} &\to c y_{ki} + s^{*} y_{ni} = 0 \\ y_{ni} &\to -s y_{ki} + c y_{ni} = 0 \end{aligned} \quad \text{pentru} \quad i < k$$

În cazul unei iniţializări adecvate, y_{kk} a avea valori reale pozitive

$$y_{kk} \to \frac{y_{kk}}{c} = \sqrt{|y_{kk}|^2 + |y_{nk}|^2}$$

7.3 ALGORITMI RECURSIVI BAZAŢI PE DESCOMPUNEREA QR. STRUCTURI SISTOLICE

7.3.2 Algoritm recursiv bazat pe rotaţii Givens

Triunghiularizarea matricei parţial triunghiulare **Y** se poate deci realiza aplicând în mod repetat acest procedeu, începând cu k=1, până la k=N, obţinând în final:

$$\mathbf{T}^{(N)}(n)\mathbf{T}^{(N-1)}(n)\cdots\mathbf{T}^{(1)}(n)\mathbf{Y}=\begin{bmatrix}\mathbf{R}\\ \mathbf{0}\end{bmatrix}=\mathbf{T}(n)\mathbf{Y}$$

Vom putea neglija aria centrală de zerouri, aşa încât reţinând numai liniile semnificative, la iteraţia n-1vom avea

$$\mathbf{R}(n-1)\mathbf{w}(n-1)=\mathbf{p}(n-1)$$

7.3 ALGORITMI RECURSIVI BAZAŢI PE DESCOMPUNEREA QR. STRUCTURI SISTOLICE

7.3.2 Algoritm recursiv bazat pe rotaţii Givens

$$\begin{bmatrix} r_{11}(n-1) & r_{12}(n-1) & . & . & . & r_{1N}(n-1) \\ 0 & r_{22}(n-1) & . & . & . & r_{2N}(n-1) \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ 0 & 0 & 0 & 0 & 0 & r_{NN}(n-1) \end{bmatrix} \begin{bmatrix} w_1(n-1) \\ w_2(n-1) \\ \\ \\ \\ w_N(n-1) \end{bmatrix} = \begin{bmatrix} p_1(n-1) \\ p_2(n-1) \\ \\ \\ \\ p_N(n-1) \end{bmatrix}$$

$$\begin{bmatrix} r_{11}(n-1) & r_{12}(n-1) & . & . & . & r_{1N}(n-1) & p_1(n-1) \\ 0 & r_{22}(n-1) & . & . & . & r_{2N}(n-1) & p_2(n-1) \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ 0 & 0 & 0 & 0 & 0 & r_{NN}(n-1) & p_N(n-1) \end{bmatrix} \begin{bmatrix} w_1(n-1) \\ w_2(n-1) \\ . \\ . \\ w_N(n-1) \\ -1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ . \\ . \\ . \\ 0 \end{bmatrix}$$

7.3 ALGORITMI RECURSIVI BAZAŢI PE DESCOMPUNEREA QR. STRUCTURI SISTOLICE

7.3.2 Algoritm recursiv bazat pe rotaţii Givens

La următoarea iteraţie, se adaugă o nouă linie la matricea din stânga, formată cu elementele

$$x_i^{(1)}(n) = x^*(n-i+1), \;\; i = 1,\ldots,N, \qquad d^{(1)}(n) = d^*(n)$$

şi se înmulţeşte cu matricea Givens construită astfel încât să se anuleze primul element al ultimei linii:

$$\begin{bmatrix} c_1 & \mathbf{0}^T & s_1^* \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \\ -s_1 & \mathbf{0}^T & c_1 \end{bmatrix} \begin{bmatrix} \sqrt{\lambda} r_{11}(n-1) & \sqrt{\lambda} r_{12}(n-1) & \ldots & \sqrt{\lambda} r_{1N}(n-1) & \sqrt{\lambda} p_1(n-1) \\ 0 & \sqrt{\lambda} r_{22}(n-1) & \ldots & \sqrt{\lambda} r_{2N}(n-1) & \sqrt{\lambda} p_2(n-1) \\ \vdots & \vdots & \ldots & \vdots & \vdots \\ 0 & 0 & \ldots & \sqrt{\lambda} r_{NN}(n-1) & \sqrt{\lambda} p_N(n-1) \\ x_1^{(1)}(n) & x_2^{(1)}(n) & \ldots & x_N^{(1)}(n) & d^{(1)}(n) \end{bmatrix}$$

în care

$$\sqrt{\lambda r_{11}^2(n-1) + \left| x_1^{(1)}(n) \right|^2} = r_{11}(n), \quad c_1 = \frac{\sqrt{\lambda} r_{11}(n-1)}{r_{11}(n)}, \quad s_1 = \frac{x_1^{(1)}(n)}{r_{11}(n)}$$

7.3 ALGORITMI RECURSIVI BAZAŢI PE DESCOMPUNEREA QR. STRUCTURI SISTOLICE

7.3.2 Algoritm recursiv bazat pe rotaţii Givens

Această operaţie va afecta numai prima şi ultima linie a matricei, care devine

$$\begin{bmatrix} r_{11}(n) & r_{12}(n) & \cdots & r_{1N}(n) & p_1(n) \\ 0 & \sqrt{\lambda} r_{22}(n-1) & \cdots & \sqrt{\lambda} r_{2N}(n-1) & \sqrt{\lambda} p_2(n-1) \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ 0 & 0 & \cdots & \sqrt{\lambda} r_{NN}(n-1) & \sqrt{\lambda} p_N(n-1) \\ 0 & x_2^{(2)}(n) & \cdots & x_N^{(2)}(n) & d^{(2)}(n) \end{bmatrix}$$

7.3 ALGORITMI RECURSIVI BAZAŢI PE DESCOMPUNEREA QR. STRUCTURI SISTOLICE

7.3.2 Algoritm recursiv bazat pe rotaţii Givens

În expresia anterioară, elementele primei linii reprezintă deci valorile finale corespunzătoare iteraţiei n şi au fost notate în consecinţă. Matricea Givens corespunzătoare pasului al doilea este

$$\mathbf{T}^{(2)}(n)=\begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & c_2 & \mathbf{0}^T & s_2^* \\ \vdots & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ 0 & -s_2 & \mathbf{0}^T & c_2 \end{bmatrix}$$

în care

$$\sqrt{\lambda r_{22}^2(n-1)+\left|x_2^{(2)}(n)\right|^2}=r_{22}(n),\quad c_2=\frac{\sqrt{\lambda}r_{22}(n-1)}{r_{22}(n)},\quad s_2=\frac{x_2^{(2)}(n)}{r_{22}(n)}$$

şi va conduce la anularea celui de-al doilea element al ultimei linii. După efectuarea a N operaţii, se obţine

$$\mathbf{T}^{(N)}(n)\mathbf{T}^{(N-1)}(n)\cdots\mathbf{T}^{(1)}(n)\begin{bmatrix} \sqrt{\lambda}\mathbf{R}(n-1) & \sqrt{\lambda}\mathbf{p}(n-1) \\ \mathbf{0}^T & d^*(n) \end{bmatrix}=\begin{bmatrix} \mathbf{R}(n) & \mathbf{p}(n) \\ \mathbf{0}^T & d^{(N+1)}(n) \end{bmatrix}$$

$$\text{for } i = 1 : N$$

$$\text{for } j = i : N$$

$$r_{ij}(0) = 0$$

$$\text{end}$$

$$\text{end}$$

$$\text{for } n = 1,2,\cdots$$

$$d^{(1)}(n) = d^{*}(n)$$

$$\text{for } \; i = 1 : 1 : N$$

$$x_i^{(1)}(n) = x^{*}(n - i - 1)$$

$$\text{end}$$

$$\text{for } \; i = 1 : 1 : \min(N, n)$$

$$r_{ii}(n) = \sqrt{\lambda r_{ii}^2(n-1) + \left| x_i^{(i)}(n) \right|^2}$$

$$c_i(n) = \frac{\sqrt{\lambda} r_{ii}(n-1)}{r_{ii}(n)}$$

$$s_i(n) = \frac{x_i^{(i)}(n)}{r_{ii}(n)}$$

$$\text{for } \; j = i + 1 : 1 : N$$

$$x_j^{(i+1)}(n) = c_i x_j^{(i)}(n) - s_i r_{ij}(n-1)$$

$$r_{ij}(n) = c_i r_{ij}(n-1) + s_i^{*} x_j^{(i)}(n)$$

$$\text{end}$$

$$d^{(i+1)}(n) = c_i d^{(i)}(n) - s_i p_i(n-1)$$

$$p_i(n) = c_i p_i(n-1) + s_i^{*} d^{(i)}(n)$$

$$\text{end}$$

$$\text{end}$$

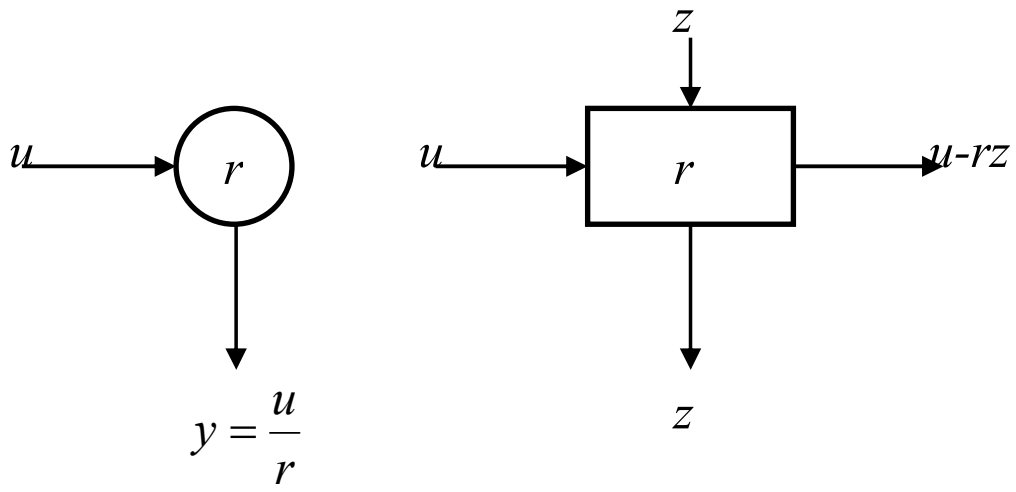
7.3 ALGORITMI RECURSIVI BAZAŢI PE DESCOMPUNEREA QR. STRUCTURI SISTOLICE

7.3.3 Implementarea algoritmilor de filtrare adaptivă cu ajutorul ariilor sistolice

O arie sistolică este o reţea de procesori care calculează şi transferă ritmic date în sistem. Întreaga reţea funcţionează cu un tact unic. În reţea are loc astfel un transfer regulat de date. Datorită gradului ridicat de paralelism, se pot realiza viteze foarte mari de prelucrare. Reţeaua conţine două tipuri de celule: *celule marginale* şi *celule interne*. Să considerăm ca exemplu rezolvarea sistemului

$$\mathbf{Rw} = \mathbf{p}$$

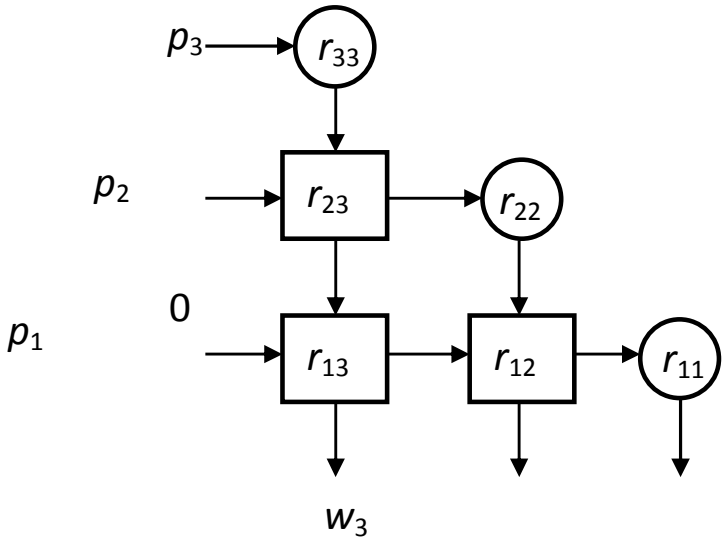
unde **R** este o matrice superior triunghiulară.



Celula marginală **Celula internă**

7.3 ALGORITMI RECURSIVI BAZAŢI PE DESCOMPUNEREA QR. STRUCTURI SISTOLICE

7.3.3 Implementarea algoritmilor de filtrare adaptivă cu ajutorul ariilor sistolice



Tactul 1: se efectuează numai împărţirea $\frac{p_3}{r_{33}}$ În "r_{33}".

Tactul 2: $p_2 - \frac{p_3}{r_{33}} r_{23}$ în "r_{23}".

Tactul 3, $p_1 - \frac{p_3}{r_{33}} r_{13}$ în "r_{13}" şi $\frac{p_2}{r_{22}} - \frac{p_3}{r_{22} r_{33}} r_{23}$ "r_{22}". La sfârşitul acestui tact se obţine deci valoarea w_3. În mod asemănător se deduce că la sfârşitul tactului 4 se obţine w_2, iar la tactul următor, w_1.

7.3 ALGORITMI RECURSIVI BAZAŢI PE DESCOMPUNEREA QR. STRUCTURI SISTOLICE

7.3.3 Implementarea algoritmilor de filtrare adaptivă cu ajutorul ariilor sistolice

Cum în realitate atât coeficienţii p cât şi parametrii r sunt funcţii de n, trebuie avut în vedere modul corect de aplicare a semnalelor pe intrări şi momentele în care se pot reactualiza parametrii $r_{ij.}$

Succesiunea semnalelor pe cele trei intrări în tactele $n, n+1, n+2, \cdots$ va fi:

$$\begin{matrix} p_3(n) & p_3(n+1) & p_3(n+2) & \cdots \\ 0 & p_2(n) & p_2(n+1) & \cdots \\ 0 & 0 & p_1(n) & \cdots \end{matrix}$$

Valorile $r_{ij}(n)$ se schimbă şi ele la fiecare tact; de exemplu, după efectuarea tactului n, parametrul $r_{33}(n)$ al celulei "r_{33}" se înlocuieşte cu $r_{33}(n+1)$, în celula "r_{23}" se înlocuieşte $r_{23}(n-1)$ cu $r_{23}(n)$ şi aşa mai departe. În momentul când se obţine la ieşirea celulei "r_{11}" valoarea $w_1(n)$, la ieşirile lui "r_{12}" şi "r_{13}" se obţin $w_2(n+1)$ şi respectiv $w_3(n+2)$.

O arhitectură sistolică poate fi utilizată în aplicarea unui algoritm care se bucură de proprietăţile de *modularitate* şi *conectare locală*.

7.3 ALGORITMI RECURSIVI BAZAŢI PE DESCOMPUNEREA QR. STRUCTURI SISTOLICE

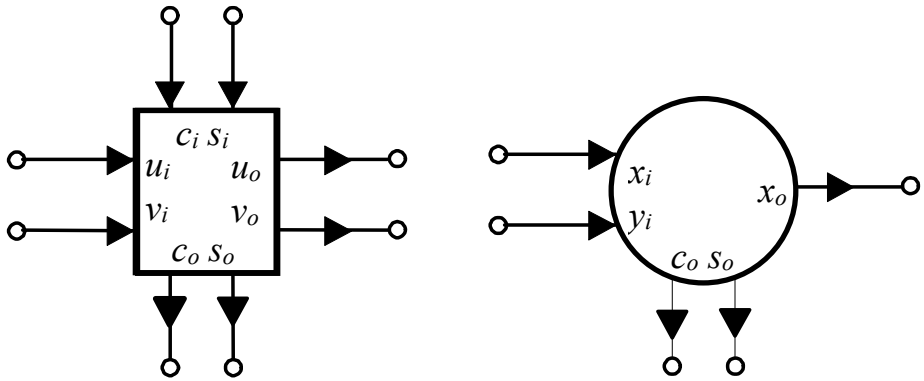
7.3.3 Implementarea algoritmilor de filtrare adaptivă cu ajutorul ariilor sistolice

Descompunerea QR cu ajutorul ariilor sistolice.

Vom porni de la algoritmul bazat pe rotaţii Givens.

- Celula internă, (*celulă rotor*) realizează operaţii de întârziere pentru operanzii verticali şi de rotire pentru cei orizontali:

$$c_o(n) = c_i(n-1)$$
$$s_o(n) = s_i(n-1)$$
$$u_o = c_i u_i - s_i \sqrt{\lambda} v_i$$
$$v_o = s_i^* u_i + c_i \sqrt{\lambda} v_i$$



a **b**

7.3 ALGORITMI RECURSIVI BAZAŢI PE DESCOMPUNEREA QR. STRUCTURI SISTOLICE

7.3.3 Implementarea algoritmilor de filtrare adaptivă cu ajutorul ariilor sistolice

- Celula marginală, *celulă cos/sin* are rolul de a genera coeficienţii rotaţiei:

$$x_o = \left(|x_i|^2 + \lambda |y_i|^2 \right)^{1/2}$$

$$c_o = \frac{\sqrt{\lambda}\, y_i}{x_o}$$

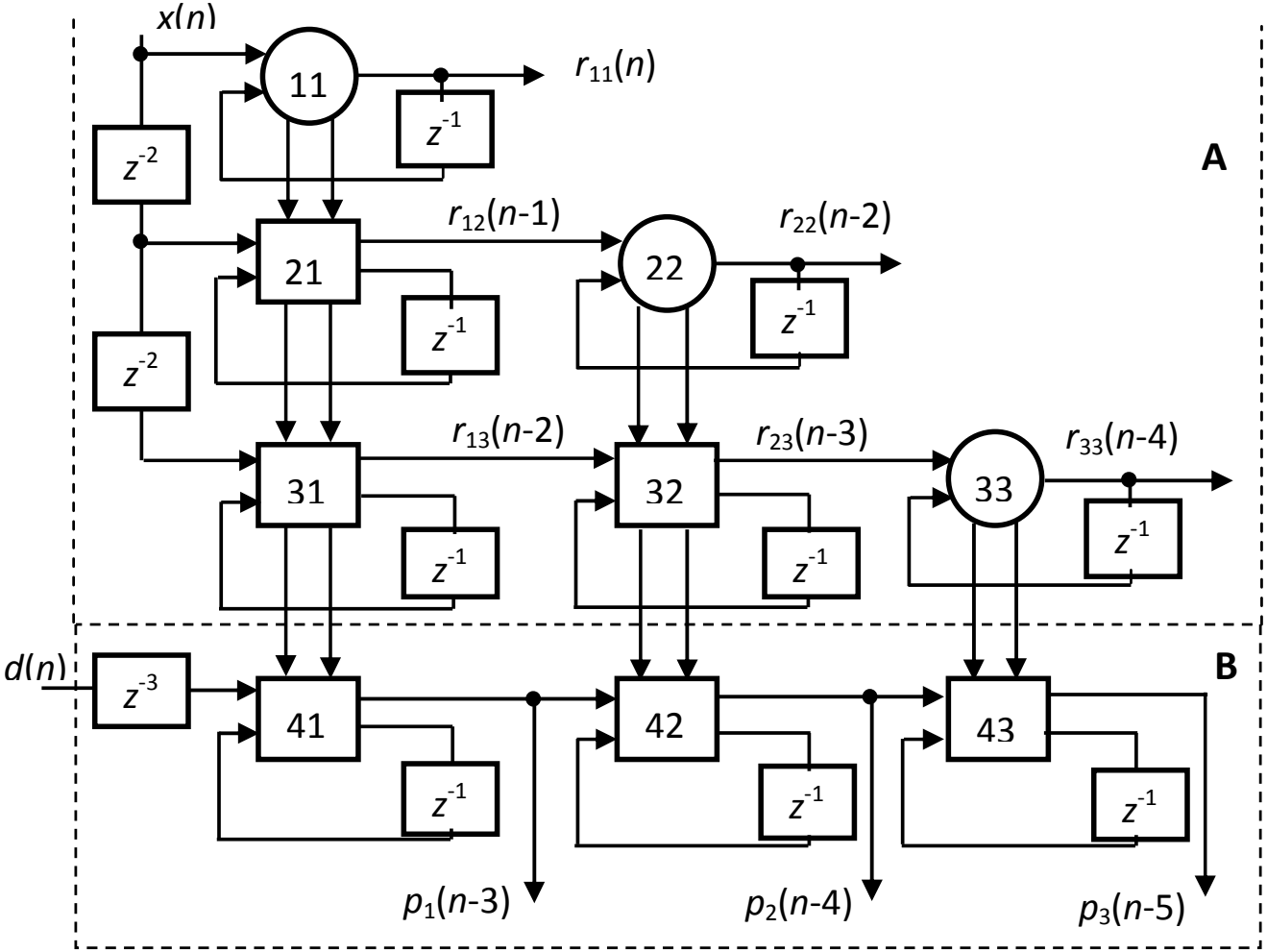
$$s_o = \frac{x_i}{x_o}$$

Prin adăugarea unor elemente de memorie (reprezentate prin circuitele de întârziere z^{-1}), celulele marginale calculează coeficienţii *s* şi *c* şi dau şi elementele $r_{ii}(n)$ iar cele interne, calculează restul de coeficienţi $r_{ij}(n)$.

Zona triunghiulară (A) generează elementele $r_{jk}(n)$ ale matricei $\mathbf{R}^T(n)$. Prima coloană va genera coeficienţii *c* şi *s* aşa încât să fie anulat efectul lui $x(n)$. Celula cos/sin respectivă va da de asemenea la ieşire $r_{11}(n)$. Aceiaşi coeficienţi trebuie aplicaţi şi celorlalte date conţinute în vectorul $\mathbf{x}(n)$, pentru a obţine celelalte elemente ale liniei 1 (r_{12},$r_{13.}$). În elementele primei coloane a procesorului vom avea la un moment dat $r_{11}(n)$, $r_{12}(n\text{-}1)$, r_{13}(n-2).

7.3 ALGORITMI RECURSIVI BAZAŢI PE DESCOMPUNEREA QR. STRUCTURI SISTOLICE

7.3.3 Implementarea algoritmilor de filtrare adaptivă cu ajutorul ariilor sistolice



7.3 ALGORITMI RECURSIVI BAZAŢI PE DESCOMPUNEREA QR. STRUCTURI SISTOLICE

7.3.3 Implementarea algoritmilor de filtrare adaptivă cu ajutorul ariilor sistolice

Coloana a doua a procesorului efectuează operaţiile necesare pentru eliminarea următorului element din linia introdusă prin datele noi. Ea generează elementele $r_{22}(n\text{-}2)$, $r_{23}(n\text{-}3)$. In fine, ultima coloană va genera elementul $r_{33}(n\text{-}4)$.

Linia B operează asupra semnalului $d(n)$ utilizând coeficienţii de rotaţie generaţi de aria A. Ea generează elementele vectorului **p**, în ordinea $p_1(n\text{-}3)$, $p_2(n\text{-}4)$, $p_3(n\text{-}5)$.