



Universidade Federal  
de Campina Grande

# **Processamento Adaptativo de Sinais**

## **Variantes do Algoritmo LMS**

*Edson P. da Silva*

*Programa de Pós-Graduação em Engenharia Elétrica (PPgEE).*

*Unidade Acadêmica de Engenharia Elétrica (UAEE)*

*Universidade Federal de Campina Grande (UFCG)*



Universidade Federal  
de Campina Grande

# Sumário

1. Variantes do algoritmo LMS.
2. Algoritmo LMS complexo.
3. Algoritmo LMS-Newton.
4. Algoritmo LMS normalizado.
5. Algoritmo LMS com domínio transformado.



Universidade Federal  
de Campina Grande

# Variantes do algoritmo LMS



Universidade Federal  
de Campina Grande

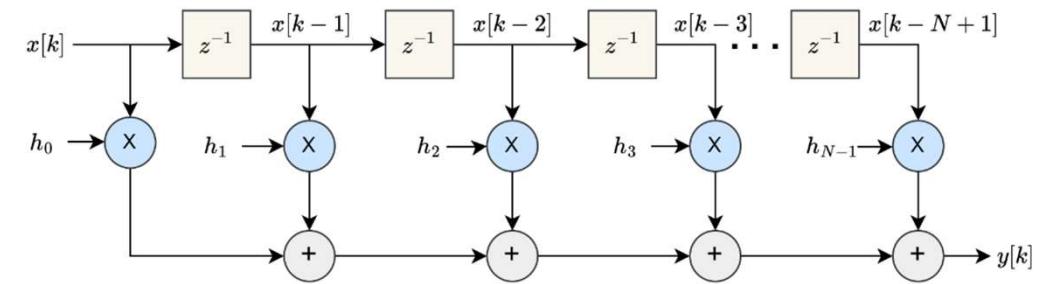
# Variantes do algoritmo LMS

- Existem diversas variantes do algoritmo LMS.

# Variantes do algoritmo LMS

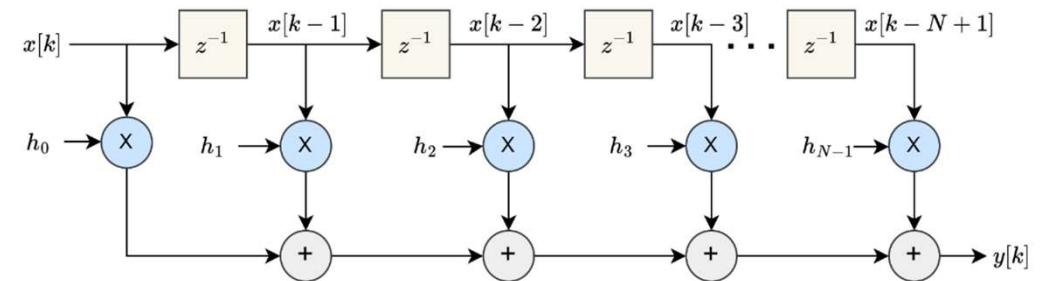
- Existem diversas variantes do algoritmo LMS.
- De maneira geral, as variantes do LMS tem por objetivo reduzir a complexidade computacional ou o tempo de convergência do algoritmo.

# Filtro de Wiener Complexo



# Filtro de Wiener Complexo

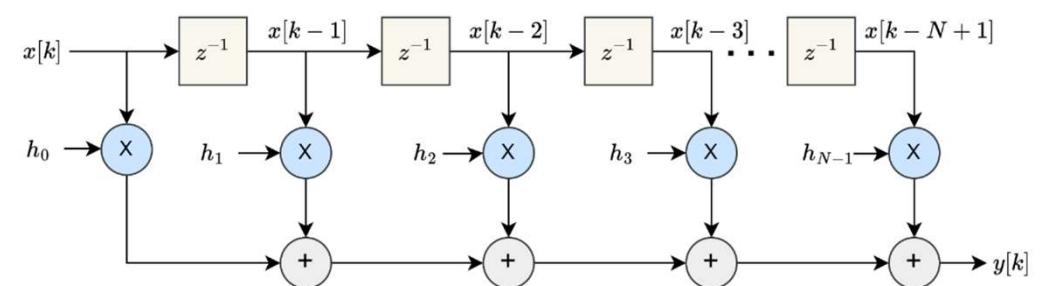
$$\mathbf{x}[k] = [x[k], x[k-1], \dots, x[k-N+1]]^T$$



# Filtro de Wiener Complexo

$$\mathbf{x}[k] = [x[k], x[k-1], \dots, x[k-N+1]]^T$$

$$\mathbf{h}[k] = [h_0[k], h_1[k], \dots, h_{N-1}[k]]^T$$

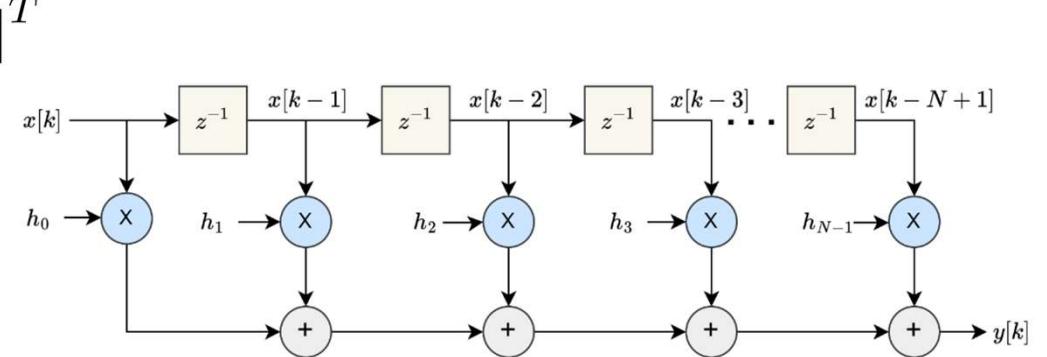


# Filtro de Wiener Complexo

$$\mathbf{x}[k] = [x[k], x[k-1], \dots, x[k-N+1]]^T$$

$$\mathbf{h}[k] = [h_0[k], h_1[k], \dots, h_{N-1}[k]]^T$$

$$y[k] = \sum_{i=0}^{N-1} h_i[k] x[k-i] = \mathbf{h}^H[k] \mathbf{x}[k]$$

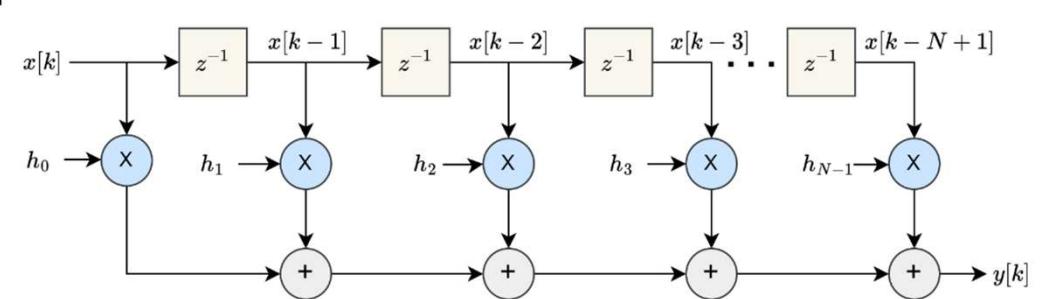


# Filtro de Wiener Complexo

$$\mathbf{x}[k] = [x[k], x[k-1], \dots, x[k-N+1]]^T$$

$$\mathbf{h}[k] = [h_0[k], h_1[k], \dots, h_{N-1}[k]]^T$$

$$y[k] = \sum_{i=0}^{N-1} h_i[k] x[k-i] = \mathbf{h}^H[k] \mathbf{x}[k]$$



Assumindo-se que  $\mathbf{x}[k]$ ,  $\mathbf{h}[k] \in \mathbb{C}^N$  e o sinal de referência  $d[k] \in \mathbb{C}$  e que as condições de estacionariedade para a solução de Wiener são atendidas, o vetor de coeficientes  $\mathbf{h}_{opt}$  que minimiza o MSE entre as saídas do filtro  $y[k]$  e o sinal de referência correspondente  $d[k]$

$$J(\mathbf{h}) = \mathbb{E} [e^*[k]e[k]] = \mathbb{E} [|e[k]|^2]$$

é dado por  $\mathbf{h}_{opt} = \mathbf{R}^{-1}\mathbf{p}$ , em que  $\mathbf{R} = \mathbb{E} [\mathbf{x}[k]\mathbf{x}^H[k]]$  e  $\mathbf{p} = \mathbb{E} [d^*[k]\mathbf{x}[k]]$ .

# Algoritmo LMS complexo



Universidade Federal  
de Campina Grande

# Algoritmo LMS complexo

Nesse caso, teremos que uma aproximação  $\hat{\nabla}_{\mathbf{h}} J(\mathbf{h})[k]$  para o gradiente  $\nabla_{\mathbf{h}} J(\mathbf{h})[k]$  é dada por:

$$\begin{aligned}\hat{\nabla}_{\mathbf{h}} J(\mathbf{h})[k] &= \hat{\mathbf{R}}[k]\mathbf{h}[k] - \hat{\mathbf{p}}[k] \\ &= \mathbf{x}[k]\mathbf{x}^H[k]\mathbf{h}[k] - d^*[k]\mathbf{x}[k]\end{aligned}$$

# Algoritmo LMS complexo

Nesse caso, teremos que uma aproximação  $\hat{\nabla}_{\mathbf{h}} J(\mathbf{h})[k]$  para o gradiente  $\nabla_{\mathbf{h}} J(\mathbf{h})[k]$  é dada por:

$$\begin{aligned}\hat{\nabla}_{\mathbf{h}} J(\mathbf{h})[k] &= \hat{\mathbf{R}}[k]\mathbf{h}[k] - \hat{\mathbf{p}}[k] \\ &= \mathbf{x}[k]\mathbf{x}^H[k]\mathbf{h}[k] - d^*[k]\mathbf{x}[k] \\ &= \mathbf{x}[k]y^*[k] - d^*[k]\mathbf{x}[k]\end{aligned}$$

# Algoritmo LMS complexo

Nesse caso, teremos que uma aproximação  $\hat{\nabla}_{\mathbf{h}} J(\mathbf{h})[k]$  para o gradiente  $\nabla_{\mathbf{h}} J(\mathbf{h})[k]$  é dada por:

$$\begin{aligned}\hat{\nabla}_{\mathbf{h}} J(\mathbf{h})[k] &= \hat{\mathbf{R}}[k]\mathbf{h}[k] - \hat{\mathbf{p}}[k] \\ &= \mathbf{x}[k]\mathbf{x}^H[k]\mathbf{h}[k] - d^*[k]\mathbf{x}[k] \\ &= \mathbf{x}[k]y^*[k] - d^*[k]\mathbf{x}[k] \\ &= -(d[k] - y[k])^*\mathbf{x}[k]\end{aligned}$$

# Algoritmo LMS complexo

Nesse caso, teremos que uma aproximação  $\hat{\nabla}_{\mathbf{h}} J(\mathbf{h})[k]$  para o gradiente  $\nabla_{\mathbf{h}} J(\mathbf{h})[k]$  é dada por:

$$\begin{aligned}\hat{\nabla}_{\mathbf{h}} J(\mathbf{h})[k] &= \hat{\mathbf{R}}[k]\mathbf{h}[k] - \hat{\mathbf{p}}[k] \\&= \mathbf{x}[k]\mathbf{x}^H[k]\mathbf{h}[k] - d^*[k]\mathbf{x}[k] \\&= \mathbf{x}[k]y^*[k] - d^*[k]\mathbf{x}[k] \\&= -(d[k] - y[k])^*\mathbf{x}[k] \\\\hat{\nabla}_{\mathbf{h}} J(\mathbf{h})[k] &= -e^*[k]\mathbf{x}[k]\end{aligned}$$

# Algoritmo LMS complexo

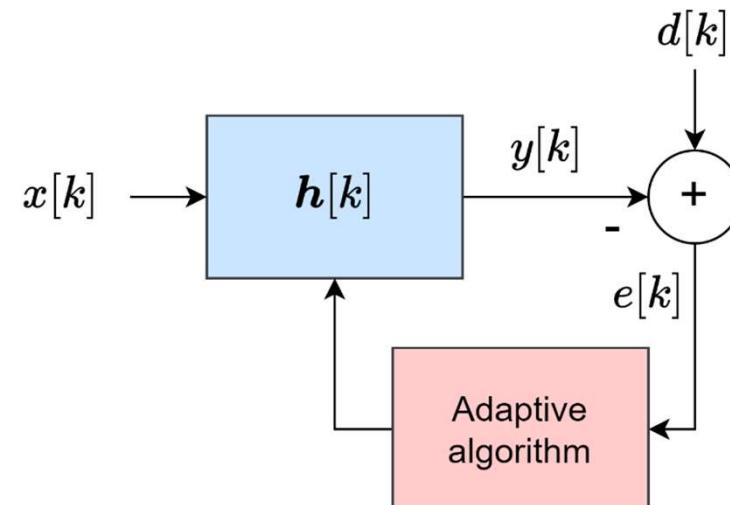
Nesse caso, teremos que uma aproximação  $\hat{\nabla}_{\mathbf{h}} J(\mathbf{h})[k]$  para o gradiente  $\nabla_{\mathbf{h}} J(\mathbf{h})[k]$  é dada por:

$$\begin{aligned}
 \hat{\nabla}_{\mathbf{h}} J(\mathbf{h})[k] &= \hat{\mathbf{R}}[k]\mathbf{h}[k] - \hat{\mathbf{p}}[k] \\
 &= \mathbf{x}[k]\mathbf{x}^H[k]\mathbf{h}[k] - d^*[k]\mathbf{x}[k] \\
 &= \mathbf{x}[k]y^*[k] - d^*[k]\mathbf{x}[k] \\
 &= -(d[k] - y[k])^*\mathbf{x}[k] \\
 \hat{\nabla}_{\mathbf{h}} J(\mathbf{h})[k] &= -e^*[k]\mathbf{x}[k]
 \end{aligned}$$

O algoritmo de gradiente descendente estocástico resultante é conhecido como LMS complexo:

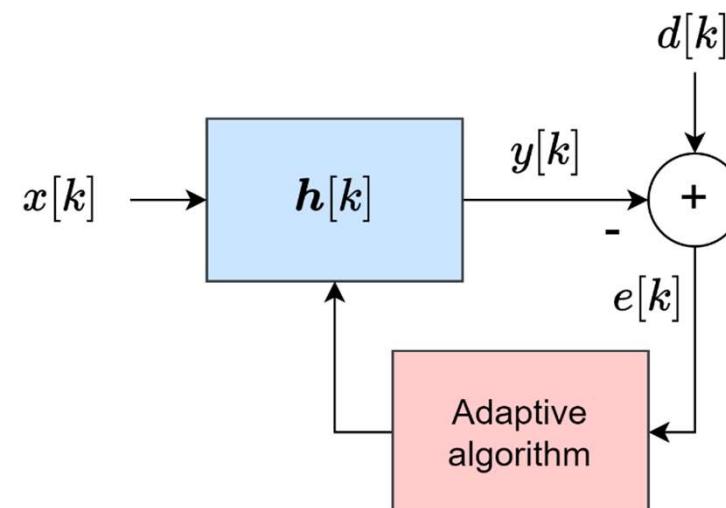
$$\mathbf{h}[k+1] = \mathbf{h}[k] + \mu e^*[k]\mathbf{x}[k]$$

# Algoritmo LMS-Newton



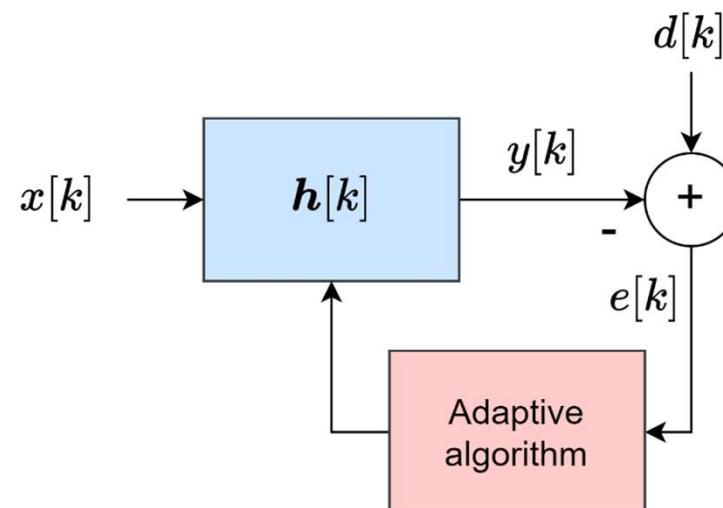
# Algoritmo LMS-Newton

- O algoritmo LMS-Newton utiliza estimativas de estatísticas de segunda ordem para aumentar a velocidade de convergência do algoritmo LMS quando o vetor de entrada é altamente correlacionado.



# Algoritmo LMS-Newton

- O algoritmo LMS-Newton utiliza estimativas de estatísticas de segunda ordem para aumentar a velocidade de convergência do algoritmo LMS quando o vetor de entrada é altamente correlacionado.
- A melhoria na velocidade de convergência é alcançada ao custo de significativo aumento computacional.



# Algoritmo LMS-Newton



Universidade Federal  
de Campina Grande

# Algoritmo LMS-Newton

- Considerando que o vetor de coeficientes  $\mathbf{h}$  a ser determinado é fixo, da análise do filtro de Wiener, temos que:

# Algoritmo LMS-Newton

- Considerando que o vetor de coeficientes  $\mathbf{h}$  a ser determinado é fixo, da análise do filtro de Wiener, temos que:

Assumindo que são conhecidos:

Vetor gradiente:

$$\nabla J_{\mathbf{h}}(\mathbf{h}) = \frac{\partial J(\mathbf{h})}{\partial \mathbf{h}} = 2(\mathbf{R}\mathbf{h} - \mathbf{p})$$

Matriz hessiana:

$$\mathbf{H}_{\mathbf{h}}(\mathbf{h}[k]) = \frac{\partial^2 J(\mathbf{h})}{\partial \mathbf{h}^2} = 2\mathbf{R}$$

# Algoritmo LMS-Newton

- Considerando que o vetor de coeficientes  $\mathbf{h}$  a ser determinado é fixo, da análise do filtro de Wiener, temos que:

Assumindo que são conhecidos:

Vetor gradiente:

$$\nabla J_{\mathbf{h}}(\mathbf{h}) = \frac{\partial J(\mathbf{h})}{\partial \mathbf{h}} = 2(\mathbf{R}\mathbf{h} - \mathbf{p})$$

Matriz hessiana:

$$\mathbf{H}_{\mathbf{h}}(\mathbf{h}[k]) = \frac{\partial^2 J(\mathbf{h})}{\partial \mathbf{h}^2} = 2\mathbf{R}$$

o algoritmo de Newton para minimização do MSE é dado por:

$$\mathbf{h}[k + 1] = \mathbf{h}[k] - \mu \mathbf{H}_{\mathbf{h}}^{-1}(\mathbf{h}[k]) \nabla_{\mathbf{h}} J(\mathbf{h}[k])$$

# Algoritmo LMS-Newton

- Considerando que o vetor de coeficientes  $\mathbf{h}$  a ser determinado é fixo, da análise do filtro de Wiener, temos que:

Assumindo que são conhecidos:

Vetor gradiente:

$$\nabla J_{\mathbf{h}}(\mathbf{h}) = \frac{\partial J(\mathbf{h})}{\partial \mathbf{h}} = 2(\mathbf{R}\mathbf{h} - \mathbf{p})$$

Matriz hessiana:

$$\mathbf{H}_{\mathbf{h}}(\mathbf{h}[k]) = \frac{\partial^2 J(\mathbf{h})}{\partial \mathbf{h}^2} = 2\mathbf{R}$$

o algoritmo de Newton para minimização do MSE é dado por:

$$\mathbf{h}[k + 1] = \mathbf{h}[k] - \mu \mathbf{H}_{\mathbf{h}}^{-1}(\mathbf{h}[k]) \nabla_{\mathbf{h}} J(\mathbf{h}[k])$$

Se  $\mu = 1$ , a solução ótima é encontrada em um passo.

# Algoritmo LMS-Newton



Universidade Federal  
de Campina Grande

# Algoritmo LMS-Newton

- Dessa forma, a regra de atualização o algoritmo LMS pode ser alterada de tal forma a obter-se um algoritmo do tipo Newton:

# Algoritmo LMS-Newton

- Dessa forma, a regra de atualização o algoritmo LMS pode ser alterada de tal forma a obter-se um algoritmo do tipo Newton:

Algoritmo LMS:

$$\begin{aligned}\mathbf{h}[k+1] &= \mathbf{h}[k] - \mu \hat{\nabla}_{\mathbf{h}} J(\mathbf{h}[k]) \\ \mathbf{h}[k+1] &= \mathbf{h}[k] + 2\mu e[k]\mathbf{x}[k]\end{aligned}$$

# Algoritmo LMS-Newton

- Dessa forma, a regra de atualização o algoritmo LMS pode ser alterada de tal forma a obter-se um algoritmo do tipo Newton:

Algoritmo LMS:

$$\begin{aligned}\mathbf{h}[k+1] &= \mathbf{h}[k] - \mu \hat{\nabla}_{\mathbf{h}} J(\mathbf{h}[k]) \\ \mathbf{h}[k+1] &= \mathbf{h}[k] + 2\mu e[k] \mathbf{x}[k]\end{aligned}$$

Algoritmo LMS-Newton:

$$\begin{aligned}\mathbf{h}[k+1] &= \mathbf{h}[k] - \mu \hat{\mathbf{H}}_{\mathbf{h}}^{-1}(\mathbf{h}[k]) \hat{\nabla}_{\mathbf{h}} J(\mathbf{h}[k]) \\ \mathbf{h}[k+1] &= \mathbf{h}[k] + 2\mu e[k] \hat{\mathbf{R}}^{-1}[k] \mathbf{x}[k]\end{aligned}$$

# Algoritmo LMS-Newton

- Dessa forma, a regra de atualização o algoritmo LMS pode ser alterada de tal forma a obter-se um algoritmo do tipo Newton:

Algoritmo LMS:

$$\begin{aligned}\mathbf{h}[k+1] &= \mathbf{h}[k] - \mu \hat{\nabla}_{\mathbf{h}} J(\mathbf{h}[k]) \\ \mathbf{h}[k+1] &= \mathbf{h}[k] + 2\mu e[k] \mathbf{x}[k]\end{aligned}$$

Algoritmo LMS-Newton:

$$\begin{aligned}\mathbf{h}[k+1] &= \mathbf{h}[k] - \mu \hat{\mathbf{H}}_{\mathbf{h}}^{-1}(\mathbf{h}[k]) \hat{\nabla}_{\mathbf{h}} J(\mathbf{h}[k]) \\ \mathbf{h}[k+1] &= \mathbf{h}[k] + 2\mu e[k] \hat{\mathbf{R}}^{-1}[k] \mathbf{x}[k]\end{aligned}$$

Lembrando que:  $\hat{\mathbf{R}}[k] = \mathbf{x}[k] \mathbf{x}^T[k]$

# Algoritmo LMS-Newton



Universidade Federal  
de Campina Grande

# Algoritmo LMS-Newton

- Um estimador não-enviesado de  $\mathbf{R}$  é dado por:

$$\begin{aligned}\hat{\mathbf{R}}[k] &= \frac{1}{k+1} \sum_{i=0}^k \mathbf{x}[i] \mathbf{x}^T[i] \\ &= \frac{k}{k+1} \hat{\mathbf{R}}[k-1] + \frac{1}{k+1} \mathbf{x}[k] \mathbf{x}^T[k]\end{aligned}$$

# Algoritmo LMS-Newton

- Um estimador não-enviesado de  $\mathbf{R}$  é dado por:

$$\begin{aligned}\hat{\mathbf{R}}[k] &= \frac{1}{k+1} \sum_{i=0}^k \mathbf{x}[i] \mathbf{x}^T[i] \\ &= \frac{k}{k+1} \hat{\mathbf{R}}[k-1] + \frac{1}{k+1} \mathbf{x}[k] \mathbf{x}^T[k]\end{aligned}$$

- Um segundo estimador não-enviesado de  $\mathbf{R}$  é dado por:

$$\begin{aligned}\hat{\mathbf{R}}[k] &= \alpha \mathbf{x}[k] \mathbf{x}^T[k] + (1 - \alpha) \hat{\mathbf{R}}[k-1] \\ &= \alpha \mathbf{x}[k] \mathbf{x}^T[k] + \alpha \sum_{i=0}^{k-1} (1 - \alpha)^{k-i} \mathbf{x}[i] \mathbf{x}^T[i]\end{aligned}$$

# Algoritmo LMS-Newton

- Um estimador não-enviesado de  $\mathbf{R}$  é dado por:

$$\begin{aligned}\hat{\mathbf{R}}[k] &= \frac{1}{k+1} \sum_{i=0}^k \mathbf{x}[i] \mathbf{x}^T[i] \\ &= \frac{k}{k+1} \hat{\mathbf{R}}[k-1] + \frac{1}{k+1} \mathbf{x}[k] \mathbf{x}^T[k]\end{aligned}$$

- Um segundo estimador não-enviesado de  $\mathbf{R}$  é dado por:

$$\begin{aligned}\hat{\mathbf{R}}[k] &= \alpha \mathbf{x}[k] \mathbf{x}^T[k] + (1 - \alpha) \hat{\mathbf{R}}[k-1] \\ &= \alpha \mathbf{x}[k] \mathbf{x}^T[k] + \alpha \sum_{i=0}^{k-1} (1 - \alpha)^{k-i} \mathbf{x}[i] \mathbf{x}^T[i]\end{aligned}$$

em que a constante  $\alpha$  é escolhida no intervalo  $0 < \alpha \leq 0.1$ , de modo a manter um balanço entre estimativas passadas e presente do sinal.

# Algoritmo LMS-Newton



Universidade Federal  
de Campina Grande

# Algoritmo LMS-Newton

- Algoritmo LMS-Newton é significativamente mais complexo do que o LMS, uma vez que requer, a cada iteração, o cálculo da inversa da matrix  $\hat{\mathbf{R}}[k]$ , cujo custo computacional é  $O(N^3)$ .

# Algoritmo LMS-Newton

- Algoritmo LMS-Newton é significativamente mais complexo do que o LMS, uma vez que requer, a cada iteração, o cálculo da inversa da matrix  $\hat{\mathbf{R}}[k]$ , cujo custo computacional é  $O(N^3)$ .
- A complexidade no cálculo de  $\hat{\mathbf{R}}^{-1}[k]$  pode ser reduzida a  $O(N^2)$ , utilizando-se um resultado conhecido como o lema da inversão de matrizes, que estabelece que

$$[\mathbf{A} + \mathbf{B}\mathbf{C}\mathbf{D}]^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B} [\mathbf{D}\mathbf{A}^{-1}\mathbf{B} + \mathbf{C}^{-1}]^{-1} \mathbf{D}\mathbf{A}^{-1}$$

em que  $\mathbf{A}$ ,  $\mathbf{B}$ ,  $\mathbf{C}$ ,  $\mathbf{D}$  são matrizes de dimensões apropriadas, sendo  $\mathbf{A}$  e  $\mathbf{C}$  não-singulares.



Universidade Federal  
de Campina Grande

# Algoritmo LMS-Newton

$$[A + BCD]^{-1} = A^{-1} - A^{-1}B [DA^{-1}B + C^{-1}]^{-1} DA^{-1}$$



Universidade Federal  
de Campina Grande

# Algoritmo LMS-Newton

$$[A + BCD]^{-1} = A^{-1} - A^{-1}B [DA^{-1}B + C^{-1}]^{-1} DA^{-1}$$

$$A = (1 - \alpha)R[k - 1]$$

# Algoritmo LMS-Newton

$$[A + BCD]^{-1} = A^{-1} - A^{-1}B [DA^{-1}B + C^{-1}]^{-1} DA^{-1}$$

$$\boldsymbol{A} = (1 - \alpha) \boldsymbol{R}[k-1] \quad \boldsymbol{B} = \boldsymbol{D}^T = \boldsymbol{x}[k]$$

# Algoritmo LMS-Newton

$$[A + BCD]^{-1} = A^{-1} - A^{-1}B [DA^{-1}B + C^{-1}]^{-1} DA^{-1}$$

$$A = (1 - \alpha)R[k - 1] \quad B = D^T = x[k] \quad C = \alpha$$

# Algoritmo LMS-Newton

$$[A + BCD]^{-1} = A^{-1} - A^{-1}B [DA^{-1}B + C^{-1}]^{-1} DA^{-1}$$

$$A = (1 - \alpha)R[k - 1] \quad B = D^T = x[k] \quad C = \alpha$$

$$[A + BCD]^{-1} = \left[ (1 - \alpha)\hat{R}[k - 1] + \alpha x[k]x^T[k] \right]^{-1} = \hat{R}^{-1}[k]$$

# Algoritmo LMS-Newton

$$[A + BCD]^{-1} = A^{-1} - A^{-1}B [DA^{-1}B + C^{-1}]^{-1} DA^{-1}$$

$$A = (1 - \alpha)R[k - 1] \quad B = D^T = x[k] \quad C = \alpha$$

$$[A + BCD]^{-1} = \left[ (1 - \alpha)\hat{R}[k - 1] + \alpha x[k]x^T[k] \right]^{-1} = \hat{R}^{-1}[k]$$

$$A^{-1} = \frac{1}{(1 - \alpha)} R^{-1}[k - 1]$$

# Algoritmo LMS-Newton

$$[A + BCD]^{-1} = A^{-1} - A^{-1}B [DA^{-1}B + C^{-1}]^{-1} DA^{-1}$$

$$A = (1 - \alpha)R[k - 1] \quad B = D^T = x[k] \quad C = \alpha$$

$$[A + BCD]^{-1} = \left[ (1 - \alpha)\hat{R}[k - 1] + \alpha x[k]x^T[k] \right]^{-1} = \hat{R}^{-1}[k]$$

$$A^{-1} = \frac{1}{(1 - \alpha)} R^{-1}[k - 1]$$

$$[DA^{-1}B + C^{-1}]^{-1} = \left[ \frac{1}{(1 - \alpha)} x^T[k] R^{-1}[k - 1] x[k] + \frac{1}{\alpha} \right]^{-1}$$

# Algoritmo LMS-Newton



Universidade Federal  
de Campina Grande



Universidade Federal  
de Campina Grande

# Algoritmo LMS-Newton

$$[A + BCD]^{-1} = A^{-1} - A^{-1}B [DA^{-1}B + C^{-1}]^{-1} DA^{-1}$$

# Algoritmo LMS-Newton

$$[A + BCD]^{-1} = A^{-1} - A^{-1}B [DA^{-1}B + C^{-1}]^{-1} DA^{-1}$$

Logo, temos a seguinte expressão para o cálculo recursivo de  $\hat{\mathbf{R}}^{-1}[k]$ :

$$\hat{\mathbf{R}}^{-1}[k] = \frac{1}{1-\alpha} \left[ \hat{\mathbf{R}}^{-1}[k-1] - \frac{\hat{\mathbf{R}}^{-1}[k-1]\mathbf{x}[k]\mathbf{x}^T[k]\hat{\mathbf{R}}^{-1}[k-1]}{\frac{1-\alpha}{\alpha} + \mathbf{x}^T[k]\hat{\mathbf{R}}^{-1}[k-1]\mathbf{x}[k]} \right]$$

# Algoritmo LMS-Newton

---

## Algorithm 1 Algoritmo LMS-Newton

---

- 1: **Entrada:** Sinal desejado  $d[k]$ , sinal de entrada  $x[k]$ , passo de adaptação  $\mu$ , constante de atualização da matrix inversa  $\alpha$ , ordem do filtro  $N$ .
- 2: **Iniciar:** Coeficientes do filtro  $\mathbf{h} = [0, 0, \dots, 0]$  (comprimento  $N$ )  
3: ▷ Podem ser inicializados de maneira arbitrária.
- 4: **Iniciar:** Inversa da matrix de autocorrelação  $\mathbf{R}^{-1} = \delta \mathbf{I}$   
5: ▷ Em que  $\delta$  é um número muito pequeno.
- 6: **for**  $k = N$  até  $M - 1$  **do** ▷ Onde  $M$  é o comprimento do sinal
- 7:     Construir o vetor de entrada  $\mathbf{x}[k] = [x[k], x[k - 1], \dots, x[k - N + 1]]^T$
- 8:     Calcular o sinal de erro:

$$e[k] = d[k] - \mathbf{h}^T \mathbf{x}[k]$$

- 9:     Atualizar  $\mathbf{R}^{-1}$ :

$$\hat{\mathbf{R}}^{-1}[k] = \frac{1}{1 - \alpha} \left[ \hat{\mathbf{R}}^{-1}[k - 1] - \frac{\hat{\mathbf{R}}^{-1}[k - 1] \mathbf{x}[k] \mathbf{x}^T[k] \hat{\mathbf{R}}^{-1}[k - 1]}{\frac{1 - \alpha}{\alpha} + \mathbf{x}^T[k] \hat{\mathbf{R}}^{-1}[k - 1] \mathbf{x}[k]} \right]$$

- 10:    Atualizar os coeficientes do filtro:

$$\mathbf{h} = \mathbf{h} + \mu e[k] \hat{\mathbf{R}}^{-1}[k] \mathbf{x}[k]$$

- 11: **end for**
  - 12: **Saída:** Coeficientes finais do filtro  $\mathbf{h}$
-

# Algoritmo LMS normalizado (NLMS)



Universidade Federal  
de Campina Grande

# Algoritmo LMS normalizado (NLMS)

- Existe alguma forma de acelerar a convergência do algoritmo LMS sem ter que recorrer a estimativas da matriz de correlação?

# Algoritmo LMS normalizado (NLMS)

- Existe alguma forma de acelerar a convergência do algoritmo LMS sem ter que recorrer a estimativas da matriz de correlação?
- Uma possível solução para o problema seria utilizar uma passo adaptativo na atualização dos coeficientes do filtro:

$$\boldsymbol{h}[k+1] = \boldsymbol{h}[k] + 2\mu[k]e[k]\boldsymbol{x}[k]$$

# Algoritmo LMS normalizado (NLMS)

- Existe alguma forma de acelerar a convergência do algoritmo LMS sem ter que recorrer a estimativas da matriz de correlação?
- Uma possível solução para o problema seria utilizar uma passo adaptativo na atualização dos coeficientes do filtro:

$$\mathbf{h}[k + 1] = \mathbf{h}[k] + 2\mu[k]e[k]\mathbf{x}[k]$$

- Uma solução natural para a escolha do passo de adaptação seria o valor que minimiza o erro quadrático instantâneo  $e^2[k]$  na saída do filtro.

$$\tilde{\mathbf{h}}[k] = \mathbf{h}[k] + \Delta\mathbf{h}[k]$$

# Algoritmo LMS normalizado (NLMS)

- Existe alguma forma de acelerar a convergência do algoritmo LMS sem ter que recorrer a estimativas da matriz de correlação?
- Uma possível solução para o problema seria utilizar uma passo adaptativo na atualização dos coeficientes do filtro:

$$\mathbf{h}[k + 1] = \mathbf{h}[k] + 2\mu[k]e[k]\mathbf{x}[k]$$

- Uma solução natural para a escolha do passo de adaptação seria o valor que minimiza o erro quadrático instantâneo  $e^2[k]$  na saída do filtro.

$$\tilde{\mathbf{h}}[k] = \mathbf{h}[k] + \Delta\mathbf{h}[k]$$

$$e^2[k] = d^2[k] - 2d[k]\mathbf{h}^T[k]\mathbf{x}[k] + \mathbf{h}^T[k]\mathbf{x}[k]\mathbf{h}^T[k]\mathbf{x}[k]$$

# Algoritmo LMS normalizado (NLMS)



Universidade Federal  
de Campina Grande

# Algoritmo LMS normalizado (NLMS)

- Seja  $\Delta\mathbf{h}[k]$  a variação nos coeficientes do filtro na iteração  $k$ , temos:

$$\tilde{\mathbf{h}}[k] = \mathbf{h}[k] + \Delta\mathbf{h}[k]$$

# Algoritmo LMS normalizado (NLMS)

- Seja  $\Delta\mathbf{h}[k]$  a variação nos coeficientes do filtro na iteração  $k$ , temos:

$$\tilde{\mathbf{h}}[k] = \mathbf{h}[k] + \Delta\mathbf{h}[k]$$

$$e^2[k] = d^2[k] - 2d[k]\mathbf{h}^T[k]\mathbf{x}[k] + \mathbf{h}^T[k]\mathbf{x}[k]\mathbf{h}^T[k]\mathbf{x}[k]$$

# Algoritmo LMS normalizado (NLMS)

- Seja  $\Delta\mathbf{h}[k]$  a variação nos coeficientes do filtro na iteração  $k$ , temos:

$$\tilde{\mathbf{h}}[k] = \mathbf{h}[k] + \Delta\mathbf{h}[k]$$

$$e^2[k] = d^2[k] - 2d[k]\mathbf{h}^T[k]\mathbf{x}[k] + \mathbf{h}^T[k]\mathbf{x}[k]\mathbf{h}^T[k]\mathbf{x}[k]$$

$$\tilde{e}^2[k] = d^2[k] - 2d[k]\tilde{\mathbf{h}}^T[k]\mathbf{x}[k] + \tilde{\mathbf{h}}^T[k]\mathbf{x}[k]\tilde{\mathbf{h}}^T[k]\mathbf{x}[k]$$

# Algoritmo LMS normalizado (NLMS)

- Seja  $\Delta\mathbf{h}[k]$  a variação nos coeficientes do filtro na iteração  $k$ , temos:

$$\tilde{\mathbf{h}}[k] = \mathbf{h}[k] + \Delta\mathbf{h}[k]$$

$$e^2[k] = d^2[k] - 2d[k]\mathbf{h}^T[k]\mathbf{x}[k] + \mathbf{h}^T[k]\mathbf{x}[k]\mathbf{h}^T[k]\mathbf{x}[k]$$

$$\tilde{e}^2[k] = d^2[k] - 2d[k]\tilde{\mathbf{h}}^T[k]\mathbf{x}[k] + \tilde{\mathbf{h}}^T[k]\mathbf{x}[k]\tilde{\mathbf{h}}^T[k]\mathbf{x}[k]$$

$$\tilde{e}^2[k] = d^2[k] - 2d[k](\mathbf{h}[k] + \Delta\mathbf{h}[k])^T\mathbf{x}[k] + (\mathbf{h}[k] + \Delta\mathbf{h}[k])^T\mathbf{x}[k](\mathbf{h}[k] + \Delta\mathbf{h}[k])^T\mathbf{x}[k]$$

# Algoritmo LMS normalizado (NLMS)

- Seja  $\Delta\mathbf{h}[k]$  a variação nos coeficientes do filtro na iteração  $k$ , temos:

$$\tilde{\mathbf{h}}[k] = \mathbf{h}[k] + \Delta\mathbf{h}[k]$$

$$e^2[k] = d^2[k] - 2d[k]\mathbf{h}^T[k]\mathbf{x}[k] + \mathbf{h}^T[k]\mathbf{x}[k]\mathbf{h}^T[k]\mathbf{x}[k]$$

$$\tilde{e}^2[k] = d^2[k] - 2d[k]\tilde{\mathbf{h}}^T[k]\mathbf{x}[k] + \tilde{\mathbf{h}}^T[k]\mathbf{x}[k]\tilde{\mathbf{h}}^T[k]\mathbf{x}[k]$$

$$\tilde{e}^2[k] = d^2[k] - 2d[k](\mathbf{h}[k] + \Delta\mathbf{h}[k])^T\mathbf{x}[k] + (\mathbf{h}[k] + \Delta\mathbf{h}[k])^T\mathbf{x}[k](\mathbf{h}[k] + \Delta\mathbf{h}[k])^T\mathbf{x}[k]$$

$$\Delta e^2[k] = \tilde{e}^2[k] - e^2[k] = -2e[k]\Delta\mathbf{h}^T[k]\mathbf{x}[k] + \Delta\mathbf{h}^T[k]\mathbf{x}[k]\mathbf{x}^T[k]\Delta\mathbf{h}[k]$$

# Algoritmo LMS normalizado (NLMS)



Universidade Federal  
de Campina Grande

# Algoritmo LMS normalizado (NLMS)

$$\Delta e^2[k] = -2e[k]\Delta \mathbf{h}^T[k]\mathbf{x}[k] + \Delta \mathbf{h}^T[k]\mathbf{x}[k]\mathbf{x}^T[k]\Delta \mathbf{h}[k]$$

$$\Delta \mathbf{h}[k] = 2\mu[k]e[k]\mathbf{x}[k]$$

# Algoritmo LMS normalizado (NLMS)

$$\Delta e^2[k] = -2e[k]\Delta\mathbf{h}^T[k]\mathbf{x}[k] + \Delta\mathbf{h}^T[k]\mathbf{x}[k]\mathbf{x}^T[k]\Delta\mathbf{h}[k]$$

$$\Delta\mathbf{h}[k] = 2\mu[k]e[k]\mathbf{x}[k]$$

$$\Delta e^2[k] = -2e[k](2\mu[k]e[k]\mathbf{x}[k])^T\mathbf{x}[k] + (2\mu[k]e[k]\mathbf{x}[k])^T\mathbf{x}[k]\mathbf{x}^T[k](2\mu[k]e[k]\mathbf{x}[k])$$

# Algoritmo LMS normalizado (NLMS)

$$\Delta e^2[k] = -2e[k]\Delta \mathbf{h}^T[k]\mathbf{x}[k] + \Delta \mathbf{h}^T[k]\mathbf{x}[k]\mathbf{x}^T[k]\Delta \mathbf{h}[k]$$

$$\Delta \mathbf{h}[k] = 2\mu[k]e[k]\mathbf{x}[k]$$

$$\Delta e^2[k] = -2e[k](2\mu[k]e[k]\mathbf{x}[k])^T\mathbf{x}[k] + (2\mu[k]e[k]\mathbf{x}[k])^T\mathbf{x}[k]\mathbf{x}^T[k](2\mu[k]e[k]\mathbf{x}[k])$$

$$\Delta e^2[k] = -4\mu[k]e^2[k]\mathbf{x}^T[k]\mathbf{x}[k] + 4\mu^2[k]e^2[k][\mathbf{x}^T[k]\mathbf{x}[k]]^2$$

# Algoritmo LMS normalizado (NLMS)

$$\Delta e^2[k] = -2e[k]\Delta\mathbf{h}^T[k]\mathbf{x}[k] + \Delta\mathbf{h}^T[k]\mathbf{x}[k]\mathbf{x}^T[k]\Delta\mathbf{h}[k]$$

$$\Delta\mathbf{h}[k] = 2\mu[k]e[k]\mathbf{x}[k]$$

$$\Delta e^2[k] = -2e[k](2\mu[k]e[k]\mathbf{x}[k])^T\mathbf{x}[k] + (2\mu[k]e[k]\mathbf{x}[k])^T\mathbf{x}[k]\mathbf{x}^T[k](2\mu[k]e[k]\mathbf{x}[k])$$

$$\Delta e^2[k] = -4\mu[k]e^2[k]\mathbf{x}^T[k]\mathbf{x}[k] + 4\mu^2[k]e^2[k][\mathbf{x}^T[k]\mathbf{x}[k]]^2$$

- Aumentar a taxa de convergência implica em fazer  $\Delta e^2[k]$  negativo e mínimo escolhendo um valor apropriado de  $\mu[k]$ .

# Algoritmo LMS normalizado (NLMS)

$$\Delta e^2[k] = -2e[k]\Delta\mathbf{h}^T[k]\mathbf{x}[k] + \Delta\mathbf{h}^T[k]\mathbf{x}[k]\mathbf{x}^T[k]\Delta\mathbf{h}[k]$$

$$\Delta\mathbf{h}[k] = 2\mu[k]e[k]\mathbf{x}[k]$$

$$\Delta e^2[k] = -2e[k](2\mu[k]e[k]\mathbf{x}[k])^T\mathbf{x}[k] + (2\mu[k]e[k]\mathbf{x}[k])^T\mathbf{x}[k]\mathbf{x}^T[k](2\mu[k]e[k]\mathbf{x}[k])$$

$$\Delta e^2[k] = -4\mu[k]e^2[k]\mathbf{x}^T[k]\mathbf{x}[k] + 4\mu^2[k]e^2[k][\mathbf{x}^T[k]\mathbf{x}[k]]^2$$

- Aumentar a taxa de convergência implica em fazer  $\Delta e^2[k]$  negativo e mínimo escolhendo um valor apropriado de  $\mu[k]$ .

$$\frac{\partial \Delta e^2[k]}{\partial \mu[k]} = 0$$

# Algoritmo LMS normalizado (NLMS)

$$\Delta e^2[k] = -2e[k]\Delta\mathbf{h}^T[k]\mathbf{x}[k] + \Delta\mathbf{h}^T[k]\mathbf{x}[k]\mathbf{x}^T[k]\Delta\mathbf{h}[k]$$

$$\Delta\mathbf{h}[k] = 2\mu[k]e[k]\mathbf{x}[k]$$

$$\Delta e^2[k] = -2e[k](2\mu[k]e[k]\mathbf{x}[k])^T\mathbf{x}[k] + (2\mu[k]e[k]\mathbf{x}[k])^T\mathbf{x}[k]\mathbf{x}^T[k](2\mu[k]e[k]\mathbf{x}[k])$$

$$\Delta e^2[k] = -4\mu[k]e^2[k]\mathbf{x}^T[k]\mathbf{x}[k] + 4\mu^2[k]e^2[k][\mathbf{x}^T[k]\mathbf{x}[k]]^2$$

- Aumentar a taxa de convergência implica em fazer  $\Delta e^2[k]$  negativo e mínimo escolhendo um valor apropriado de  $\mu[k]$ .

$$\frac{\partial \Delta e^2[k]}{\partial \mu[k]} = 0 \quad \Rightarrow \quad \mu[k] = \frac{1}{2\mathbf{x}^T[k]\mathbf{x}[k]}$$

# Algoritmo LMS normalizado (NLMS)



Universidade Federal  
de Campina Grande

# Algoritmo LMS normalizado (NLMS)

- Utilizando o passo adaptativo  $\mu[k]$  encontrado, temos que:

$$\boldsymbol{h}[k + 1] = \boldsymbol{h}[k] + 2\mu[k]e[k]\boldsymbol{x}[k]$$

# Algoritmo LMS normalizado (NLMS)

- Utilizando o passo adaptativo  $\mu[k]$  encontrado, temos que:

$$\mathbf{h}[k+1] = \mathbf{h}[k] + 2\mu[k]e[k]\mathbf{x}[k] \implies \mathbf{h}[k+1] = \mathbf{h}[k] + \mu_{\text{NLMS}} \frac{e[k]\mathbf{x}[k]}{\mathbf{x}^T[k]\mathbf{x}[k]}$$

# Algoritmo LMS normalizado (NLMS)

- Utilizando o passo adaptativo  $\mu[k]$  encontrado, temos que:

$$\mathbf{h}[k+1] = \mathbf{h}[k] + 2\mu[k]e[k]\mathbf{x}[k] \implies \mathbf{h}[k+1] = \mathbf{h}[k] + \mu_{\text{NLMS}} \frac{e[k]\mathbf{x}[k]}{\mathbf{x}^T[k]\mathbf{x}[k]}$$

ou seja,

$$\mathbf{h}[k+1] = \mathbf{h}[k] + \mu_{\text{NLMS}} e[k] \frac{\mathbf{x}[k]}{\|\mathbf{x}[k]\|^2}$$

# Algoritmo LMS normalizado (NLMS)

- Utilizando o passo adaptativo  $\mu[k]$  encontrado, temos que:

$$\mathbf{h}[k+1] = \mathbf{h}[k] + 2\mu[k]e[k]\mathbf{x}[k] \implies \mathbf{h}[k+1] = \mathbf{h}[k] + \mu_{\text{NLMS}} \frac{e[k]\mathbf{x}[k]}{\mathbf{x}^T[k]\mathbf{x}[k]}$$

ou seja,

$$\mathbf{h}[k+1] = \mathbf{h}[k] + \mu_{\text{NLMS}} e[k] \frac{\mathbf{x}[k]}{\|\mathbf{x}[k]\|^2}$$

- O algoritmo resultante é conhecido como algoritmo LMS normalizado.

# Algoritmo LMS normalizado (NLMS)

- Utilizando o passo adaptativo  $\mu[k]$  encontrado, temos que:

$$\mathbf{h}[k+1] = \mathbf{h}[k] + 2\mu[k]e[k]\mathbf{x}[k] \implies \mathbf{h}[k+1] = \mathbf{h}[k] + \mu_{\text{NLMS}} \frac{e[k]\mathbf{x}[k]}{\mathbf{x}^T[k]\mathbf{x}[k]}$$

ou seja,

$$\mathbf{h}[k+1] = \mathbf{h}[k] + \mu_{\text{NLMS}} e[k] \frac{\mathbf{x}[k]}{\|\mathbf{x}[k]\|^2}$$

- O algoritmo resultante é conhecido como algoritmo LMS normalizado.
- O intervalo de valores que  $\mu[k]$  pode assumir, garantindo a convergência do algoritmo, pode ser obtido considerando primeiramente que  $E [\mathbf{x}^T[k]\mathbf{x}[k]] = \text{tr}[\mathbf{R}]$ , e logo:

# Algoritmo LMS normalizado (NLMS)

- Utilizando o passo adaptativo  $\mu[k]$  encontrado, temos que:

$$\mathbf{h}[k+1] = \mathbf{h}[k] + 2\mu[k]e[k]\mathbf{x}[k] \implies \mathbf{h}[k+1] = \mathbf{h}[k] + \mu_{\text{NLMS}} \frac{e[k]\mathbf{x}[k]}{\mathbf{x}^T[k]\mathbf{x}[k]}$$

ou seja,

$$\mathbf{h}[k+1] = \mathbf{h}[k] + \mu_{\text{NLMS}} e[k] \frac{\mathbf{x}[k]}{\|\mathbf{x}[k]\|^2}$$

- O algoritmo resultante é conhecido como algoritmo LMS normalizado.
- O intervalo de valores que  $\mu[k]$  pode assumir, garantindo a convergência do algoritmo, pode ser obtido considerando primeiramente que  $E[\mathbf{x}^T[k]\mathbf{x}[k]] = \text{tr}[\mathbf{R}]$ , e logo:

$$E[\Delta\mathbf{h}[k]] = E\left[\mu_{\text{NLMS}} \frac{e[k]\mathbf{x}[k]}{\mathbf{x}^T[k]\mathbf{x}[k]}\right] \approx \mu_{\text{NLMS}} \frac{E[e[k]\mathbf{x}[k]]}{E[\mathbf{x}^T[k]\mathbf{x}[k]]} = \frac{\mu_{\text{NLMS}}}{\text{tr}[\mathbf{R}]} E[e[k]\mathbf{x}[k]]$$

# Algoritmo LMS normalizado (NLMS)

$$E [\Delta \boldsymbol{h}[k]] \approx \frac{\mu_{\text{NLMS}}}{\text{tr}[\boldsymbol{R}]} E [e[k] \boldsymbol{x}[k]] = \frac{2\mu_{\text{NLMS}}}{2 \text{tr}[\boldsymbol{R}]} E [e[k] \boldsymbol{x}[k]]$$

# Algoritmo LMS normalizado (NLMS)



Universidade Federal  
de Campina Grande

$$E [\Delta \boldsymbol{h}[k]] \approx \frac{\mu_{\text{NLMS}}}{\text{tr}[\boldsymbol{R}]} E [e[k] \boldsymbol{x}[k]] = \frac{2\mu_{\text{NLMS}}}{2 \text{tr}[\boldsymbol{R}]} E [e[k] \boldsymbol{x}[k]]$$

- Logo:

# Algoritmo LMS normalizado (NLMS)

$$E [\Delta \mathbf{h}[k]] \approx \frac{\mu_{\text{NLMS}}}{\text{tr}[\mathbf{R}]} E [e[k] \mathbf{x}[k]] = \frac{2\mu_{\text{NLMS}}}{2 \text{tr}[\mathbf{R}]} E [e[k] \mathbf{x}[k]]$$

- Logo:

$$\mu_{\text{LMS}} = \frac{\mu_{\text{NLMS}}}{\text{tr}[2\mathbf{R}]}$$

# Algoritmo LMS normalizado (NLMS)

$$E[\Delta \mathbf{h}[k]] \approx \frac{\mu_{\text{NLMS}}}{\text{tr}[\mathbf{R}]} E[e[k] \mathbf{x}[k]] = \frac{2\mu_{\text{NLMS}}}{2 \text{tr}[\mathbf{R}]} E[e[k] \mathbf{x}[k]]$$

- Logo:

$$\mu_{\text{LMS}} = \frac{\mu_{\text{NLMS}}}{\text{tr}[2\mathbf{R}]} \implies 0 < \frac{\mu_{\text{NLMS}}}{2 \text{tr}[\mathbf{R}]} < \frac{1}{\text{tr}[\mathbf{R}]}$$

# Algoritmo LMS normalizado (NLMS)

$$E[\Delta \mathbf{h}[k]] \approx \frac{\mu_{\text{NLMS}}}{\text{tr}[\mathbf{R}]} E[e[k] \mathbf{x}[k]] = \frac{2\mu_{\text{NLMS}}}{2 \text{tr}[\mathbf{R}]} E[e[k] \mathbf{x}[k]]$$

- Logo:

$$\mu_{\text{LMS}} = \frac{\mu_{\text{NLMS}}}{\text{tr}[2\mathbf{R}]} \implies 0 < \frac{\mu_{\text{NLMS}}}{2 \text{tr}[\mathbf{R}]} < \frac{1}{\text{tr}[\mathbf{R}]} \implies 0 < \mu_{\text{NLMS}} < 2$$

# Algoritmo LMS normalizado (NLMS)

$$E[\Delta \mathbf{h}[k]] \approx \frac{\mu_{\text{NLMS}}}{\text{tr}[\mathbf{R}]} E[e[k] \mathbf{x}[k]] = \frac{2\mu_{\text{NLMS}}}{2 \text{tr}[\mathbf{R}]} E[e[k] \mathbf{x}[k]]$$

- Logo:

$$\mu_{\text{LMS}} = \frac{\mu_{\text{NLMS}}}{\text{tr}[2\mathbf{R}]} \implies 0 < \frac{\mu_{\text{NLMS}}}{2 \text{tr}[\mathbf{R}]} < \frac{1}{\text{tr}[\mathbf{R}]} \implies 0 < \mu_{\text{NLMS}} < 2$$

Na prática:  $0 < \mu_{\text{NLMS}} \leq 1$

# Algoritmo LMS no domínio transformado



Universidade Federal  
de Campina Grande

# Algoritmo LMS no domínio transformado

- A transformação do domínio é outra técnica para aumentar a velocidade de convergência do algoritmo LMS quando o sinal de entrada é altamente correlacionado.

# Algoritmo LMS no domínio transformado

- A transformação do domínio é outra técnica para aumentar a velocidade de convergência do algoritmo LMS quando o sinal de entrada é altamente correlacionado.
- A ideia básica é aplicar uma transformação linear unitária  $\mathbf{M}$  ao vetor de entrada  $\mathbf{x}$  do filtro adaptativo de modo que o número de condicionamento  $C = \frac{\lambda_{\max}}{\lambda_{\min}}$  da matriz de correlação  $\mathbf{R}_M$  correspondente seja melhorado.

# Algoritmo LMS no domínio transformado

- A transformação do domínio é outra técnica para aumentar a velocidade de convergência do algoritmo LMS quando o sinal de entrada é altamente correlacionado.
- A ideia básica é aplicar uma transformação linear unitária  $\mathbf{M}$  ao vetor de entrada  $\mathbf{x}$  do filtro adaptativo de modo que o número de condicionamento  $C = \frac{\lambda_{\max}}{\lambda_{\min}}$  da matriz de correlação  $\mathbf{R}_M$  correspondente seja melhorado.

Transformação unitária :  $\mathbf{M}\mathbf{M}^T = \mathbf{M}^T\mathbf{M} = \mathbf{I}$

# Algoritmo LMS no domínio transformado

- A transformação do domínio é outra técnica para aumentar a velocidade de convergência do algoritmo LMS quando o sinal de entrada é altamente correlacionado.
- A ideia básica é aplicar uma transformação linear unitária  $\mathbf{M}$  ao vetor de entrada  $\mathbf{x}$  do filtro adaptativo de modo que o número de condicionamento  $C = \frac{\lambda_{\max}}{\lambda_{\min}}$  da matriz de correlação  $\mathbf{R}_M$  correspondente seja melhorado.

Transformação unitária :  $\mathbf{M}\mathbf{M}^T = \mathbf{M}^T\mathbf{M} = \mathbf{I}$        $\mathbf{x}_M[k] = \mathbf{M}\mathbf{x}$

# Algoritmo LMS no domínio transformado

- A transformação do domínio é outra técnica para aumentar a velocidade de convergência do algoritmo LMS quando o sinal de entrada é altamente correlacionado.
- A ideia básica é aplicar uma transformação linear unitária  $\mathbf{M}$  ao vetor de entrada  $\mathbf{x}$  do filtro adaptativo de modo que o número de condicionamento  $C = \frac{\lambda_{\max}}{\lambda_{\min}}$  da matriz de correlação  $\mathbf{R}_M$  correspondente seja melhorado.

Transformação unitária :  $\mathbf{M}\mathbf{M}^T = \mathbf{M}^T\mathbf{M} = \mathbf{I}$        $\mathbf{x}_M[k] = \mathbf{M}\mathbf{x}$

- Ao aplicarmos a transformada, a nova matriz de correlação  $\mathbf{R}_M$  será dada por:

$$\mathbf{R}_M = \mathbb{E}[\mathbf{x}_M \mathbf{x}_M^T]$$

# Algoritmo LMS no domínio transformado

- A transformação do domínio é outra técnica para aumentar a velocidade de convergência do algoritmo LMS quando o sinal de entrada é altamente correlacionado.
- A ideia básica é aplicar uma transformação linear unitária  $\mathbf{M}$  ao vetor de entrada  $\mathbf{x}$  do filtro adaptativo de modo que o número de condicionamento  $C = \frac{\lambda_{\max}}{\lambda_{\min}}$  da matriz de correlação  $\mathbf{R}_M$  correspondente seja melhorado.

Transformação unitária :  $\mathbf{M}\mathbf{M}^T = \mathbf{M}^T\mathbf{M} = \mathbf{I}$        $\mathbf{x}_M[k] = \mathbf{M}\mathbf{x}$

- Ao aplicarmos a transformada, a nova matriz de correlação  $\mathbf{R}_M$  será dada por:

$$\mathbf{R}_M = \mathbb{E}[\mathbf{x}_M \mathbf{x}_M^T]$$

$$\mathbf{R}_M = \mathbb{E}[\mathbf{M}\mathbf{x} \mathbf{x}^T \mathbf{M}^T]$$

# Algoritmo LMS no domínio transformado

- A transformação do domínio é outra técnica para aumentar a velocidade de convergência do algoritmo LMS quando o sinal de entrada é altamente correlacionado.
- A ideia básica é aplicar uma transformação linear unitária  $\mathbf{M}$  ao vetor de entrada  $\mathbf{x}$  do filtro adaptativo de modo que o número de condicionamento  $C = \frac{\lambda_{\max}}{\lambda_{\min}}$  da matriz de correlação  $\mathbf{R}_M$  correspondente seja melhorado.

Transformação unitária :  $\mathbf{M}\mathbf{M}^T = \mathbf{M}^T\mathbf{M} = \mathbf{I}$        $\mathbf{x}_M[k] = \mathbf{M}\mathbf{x}$

- Ao aplicarmos a transformada, a nova matriz de correlação  $\mathbf{R}_M$  será dada por:

$$\mathbf{R}_M = \mathbb{E}[\mathbf{x}_M \mathbf{x}_M^T]$$

$$\mathbf{R}_M = \mathbf{M} \mathbb{E}[\mathbf{x} \mathbf{x}^T] \mathbf{M}^T$$

$$\mathbf{R}_M = \mathbb{E}[\mathbf{M} \mathbf{x} \mathbf{x}^T \mathbf{M}^T]$$

# Algoritmo LMS no domínio transformado

- A transformação do domínio é outra técnica para aumentar a velocidade de convergência do algoritmo LMS quando o sinal de entrada é altamente correlacionado.
- A ideia básica é aplicar uma transformação linear unitária  $\mathbf{M}$  ao vetor de entrada  $\mathbf{x}$  do filtro adaptativo de modo que o número de condicionamento  $C = \frac{\lambda_{\max}}{\lambda_{\min}}$  da matriz de correlação  $\mathbf{R}_M$  correspondente seja melhorado.

Transformação unitária :  $\mathbf{M}\mathbf{M}^T = \mathbf{M}^T\mathbf{M} = \mathbf{I}$        $\mathbf{x}_M[k] = \mathbf{M}\mathbf{x}$

- Ao aplicarmos a transformada, a nova matriz de correlação  $\mathbf{R}_M$  será dada por:

$$\mathbf{R}_M = \mathbb{E}[\mathbf{x}_M \mathbf{x}_M^T]$$

$$\mathbf{R}_M = \mathbb{E}[\mathbf{M}\mathbf{x}\mathbf{x}^T\mathbf{M}^T]$$

$$\mathbf{R}_M = \mathbf{M}\mathbb{E}[\mathbf{x}\mathbf{x}^T]\mathbf{M}^T$$

$$\mathbf{R}_M = \mathbf{M}\mathbf{R}\mathbf{M}^T$$

# Algoritmo LMS no domínio transformado



Universidade Federal  
de Campina Grande

# Algoritmo LMS no domínio transformado

- Se os elementos de  $\mathbf{x}_M[k]$  forem descorrelacionados,  $\mathbf{R}_M$  será diagonal. Logo,  $\mathbf{M}$  é a matriz que diagonaliza  $\mathbf{R}$ , ou seja, é a matriz cujas colunas são os autovetores de  $\mathbf{R}$ .

# Algoritmo LMS no domínio transformado

- Se os elementos de  $\mathbf{x}_M[k]$  forem descorrelacionados,  $\mathbf{R}_M$  será diagonal. Logo,  $\mathbf{M}$  é a matriz que diagonaliza  $\mathbf{R}$ , ou seja, é a matriz cujas colunas são os autovetores de  $\mathbf{R}$ .
- A matriz de transformação  $\mathbf{M}$  resultante corresponde à Transformada de Karhunen-Loëve (KLT).

# Algoritmo LMS no domínio transformado

- Se os elementos de  $\mathbf{x}_M[k]$  forem descorrelacionados,  $\mathbf{R}_M$  será diagonal. Logo,  $\mathbf{M}$  é a matriz que diagonaliza  $\mathbf{R}$ , ou seja, é a matriz cujas colunas são os autovetores de  $\mathbf{R}$ .
- A matriz de transformação  $\mathbf{M}$  resultante corresponde à Transformada de Karhunen-Loëve (KLT).
- Podemos, então, normalizar individualmente cada componente do vetor  $\mathbf{x}_M[k]$ , dividindo cada componente  $x_{M,i}$  por  $\sigma_i^2 + \gamma$ , em que  $\sigma_i^2 = \mathbb{E}[x_{M,i}^2]$ .

# Algoritmo LMS no domínio transformado

- Se os elementos de  $\mathbf{x}_M[k]$  forem descorrelacionados,  $\mathbf{R}_M$  será diagonal. Logo,  $\mathbf{M}$  é a matriz que diagonaliza  $\mathbf{R}$ , ou seja, é a matriz cujas colunas são os autovetores de  $\mathbf{R}$ .
- A matriz de transformação  $\mathbf{M}$  resultante corresponde à Transformada de Karhunen-Loëve (KLT).
- Podemos, então, normalizar individualmente cada componente do vetor  $\mathbf{x}_M[k]$ , dividindo cada componente  $x_{M,i}$  por  $\sigma_i^2 + \gamma$ , em que  $\sigma_i^2 = \mathbb{E}[x_{M,i}^2]$ .

$$\sigma_i^2[k] = \mathbb{E}[x_{M,i}^2] \approx \alpha x_{M,i}^2[k] + (1 - \alpha) \sigma_i^2[k - 1]$$

# Algoritmo LMS no domínio transformado



Universidade Federal  
de Campina Grande

# Algoritmo LMS no domínio transformado

- A cada iteração  $k$ , podemos representar os fatores de normalização por meio de uma matriz diagonal  $\Sigma^{-2}[k]$ , dada por:

$$\Sigma^{-2}[k] = \begin{bmatrix} \frac{1}{\gamma + \sigma_0^2[k]} & 0 & \cdots & 0 \\ 0 & \frac{1}{\gamma + \sigma_1^2[k]} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{\gamma + \sigma_{N-1}^2[k]} \end{bmatrix}$$

# Algoritmo LMS no domínio transformado

- A cada iteração  $k$ , podemos representar os fatores de normalização por meio de uma matriz diagonal  $\Sigma^{-2}[k]$ , dada por:

$$\Sigma^{-2}[k] = \begin{bmatrix} \frac{1}{\gamma + \sigma_0^2[k]} & 0 & \cdots & 0 \\ 0 & \frac{1}{\gamma + \sigma_1^2[k]} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{\gamma + \sigma_{N-1}^2[k]} \end{bmatrix}$$

- Note que isto equivale a aplicar uma normalização similar a do NLMS, porém *por dimensão do vetor de entrada*.

# Algoritmo LMS no domínio transformado

- A cada iteração  $k$ , podemos representar os fatores de normalização por meio de uma matriz diagonal  $\Sigma^{-2}[k]$ , dada por:

$$\Sigma^{-2}[k] = \begin{bmatrix} \frac{1}{\gamma + \sigma_0^2[k]} & 0 & \cdots & 0 \\ 0 & \frac{1}{\gamma + \sigma_1^2[k]} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{\gamma + \sigma_{N-1}^2[k]} \end{bmatrix}$$

- Note que isto equivale a aplicar uma normalização similar a do NLMS, porém *por dimensão do vetor de entrada*.
- O algoritmo LMS toma a seguinte forma:

# Algoritmo LMS no domínio transformado

- A cada iteração  $k$ , podemos representar os fatores de normalização por meio de uma matriz diagonal  $\Sigma^{-2}[k]$ , dada por:

$$\Sigma^{-2}[k] = \begin{bmatrix} \frac{1}{\gamma + \sigma_0^2[k]} & 0 & \cdots & 0 \\ 0 & \frac{1}{\gamma + \sigma_1^2[k]} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{\gamma + \sigma_{N-1}^2[k]} \end{bmatrix}$$

- Note que isto equivale a aplicar uma normalização similar a do NLMS, porém *por dimensão do vetor de entrada*.
- O algoritmo LMS toma a seguinte forma:

$$y[k] = \mathbf{h}_M[k] \mathbf{x}_M[k] \quad \sigma_i^2[k] = \alpha x_{M,i}^2[k] + (1 - \alpha) \sigma_i^2[k - 1]$$

$$e[k] = d[k] - y[k] \quad \mathbf{h}_M[k + 1] = \mathbf{h}_M[k] + 2\mu e[k] \Sigma^{-2}[k] \mathbf{x}_M[k]$$

# Algoritmo LMS no domínio transformado

- Pode ser demonstrado que, se  $\mu$  for escolhido de maneira apropriada,  $\mathbf{h}_M[k]$  convergirá para a solução de Wiener, que nesse caso será:

$$\mathbf{h}_{M,opt} = \mathbf{R}_M^{-1} \mathbf{p}_M$$

em que  $\mathbf{R}_M = \mathbf{M} \mathbf{R} \mathbf{M}^T$  e  $\mathbf{p}_M = \mathbf{M} \mathbf{p}$ .

- Logo:

$$\begin{aligned} \mathbf{h}_{M,opt} &= (\mathbf{M} \mathbf{R} \mathbf{M}^T)^{-1} \mathbf{M} \mathbf{p} \implies \mathbf{h}_{M,opt} = (\mathbf{R} \mathbf{M}^T)^{-1} \mathbf{M}^T \mathbf{M} \mathbf{p} \\ &\implies \mathbf{h}_{M,opt} = \mathbf{R}^{-1} \mathbf{p} \implies \mathbf{h}_{M,opt} = \mathbf{M} \mathbf{h}_{opt} \end{aligned}$$

$$(\mathbf{AB})^{-1} = \mathbf{B}^{-1} \mathbf{A}^{-1}$$