Universidade Federal do Ceará
Centro de Tecnologia
Departamento de Engenharia de Teleinformática
Filtragem Adaptativa - TIP7188

Lista de Exercícios

Aluno: Lucas de Souza Abdalah (539567)

Professor: Charles Casimiro Cavalcante e Guilherme de Alencar Barreto
Data de Entrega: 20/07/2022

Fortaleza
2022

Sumário

1 Lista 1: Estatísticas de Segunda Ordem

1.1 Média e Autocorrelação

Para obter a média, basta a expressão de acordo com o operador esperança $\mathbb{E}\{\cdot\}$, dado que as variáveis aleatórias tem mesma média, resume-se a expressão:

$$\begin{aligned}\mathbb{E}\{x(n)\} &= \mathbb{E}\{v(n) + 3v(n-1)\}\\ &= \mu + 3\mu\\ &= 4\mu\end{aligned}$$

Já a variância, é obtida aplicando o mesmo operador, "abrindo"o termo ao quadrado, reorganizando em função do termo σ^2 e sabendo que $v(n)$ e $v(n-1)$ são descorrelacionadas:

$$\begin{aligned}\mathbb{E}\{[(x(n) - \mu_X)]^2\} &= \mathbb{E}\{[v(n) - 3v(n-1) - \mu_X]^2\}\\ &= \mathbb{E}\{[v(n) - 3v(n-1) - 4\mu]^2\}\\ &= \mathbb{E}\{[v(n) - \mu + 3v(n-1) - 3\mu]^2\}\\ &= \sigma^2 + 9\sigma^2 + \mathbb{E}\{6[v(n) - \mu][v(n-1) - \mu]\}\\ &= 10\sigma^2\end{aligned}$$

Para afirmar que o processo apresentado é estacionário em sentido amplo, abrevidado em inglês para **WSS**, as estatísticas de primeira e de segunda ordem devem ser independentes ao deslocamento no tempo. Isto pode ser observado, assumindo novamente que $x(n)$ e $x(n+\tau)$, via função de correlação, dada por:

$$\begin{aligned}\mathbb{E}\{x(n)x(n+\tau)\} &= \mathbb{E}\{[v(n) + 3v(n-1)][v(n+\tau) + 3v(n-1+\tau)]\},\\ &= \mathbb{E}\{v(n)v(n+\tau) + 3v(n)v(n-1+\tau) + 3v(n-1)v(n+\tau) + 9v(n-1)v(n-1+\tau)\}\\ &= \mathbb{E}\{\mu^2 + 3\mu^2 + 3\mu^2 + 9\mu^2\}\\ &= 16\mu^2\end{aligned}$$

Visto que os pré-requisitos são cumpridos, pode-se concluir que o processo é de fato WSS. Entretanto, para afirmar algo além disso é necessário conhecer os movimentos de ordem superior do caso estudado.

1.2 Processos Estacionários

Funções de autocorrelação de x e de y

Primeiramente, é conveniente definir o processo de ruído branco, visto que este possui propriedades bastante conveniente para a solução do problema. O processo desta natureza tem média nula e tem todas as suas amostras independentes entre si. Isto permite que seja obtida a média de novos processos resultantes da mistura linear desses ruídos.

Para $x(n)$, obtém-se a média dado:

$$\begin{aligned}\mathbb{E}\{x(n)\} &= \mathbb{E}\{v_1(n) + 3v_2(n-1)\}\\ &= \mu_1 + 3\mu_2\\ &= 0\end{aligned}$$

Enquanto para a variânciam, tem-se que (semelhante ao exercício 1.1):

$$
\begin{aligned}
\mathbb{E}\{[x(n)-\mu]^2\} &= \mathbb{E}\{[x(n)-0]^2\}\\
&= \mathbb{E}\{[v_1(n)+3v_2(n-1)]^2\}\\
&= \mathbb{E}\{[v_1^2(n)]+6[v_1(n)v_2(n-1)]+9[v_2^2(n-1)]\}\\
&= 1\sigma_1^2+9\sigma_2^2\\
&= 5
\end{aligned}
$$

O mesmo procedimento é aplicado para $y(n)$:

$$
\begin{aligned}
\mathbb{E}\{y(n)\} &= \mathbb{E}\{v_2(n+1)+3v_1(n-1)\}\\
&= \mu_2+3\mu_1\\
&= 0
\end{aligned}
$$

$$
\begin{aligned}
\mathbb{E}\{[y(n)-\mu]^2\} &= \mathbb{E}\{[y(n)-0]^2\}\\
&= \mathbb{E}\{[v_2(n+1)+3v_1(n-1)]^2\}\\
&= \mathbb{E}\{[v_2^2(n)]+6[v_2(n+1)v_1(n-1)]+9[v_2^1(n-1)]\}\\
&= 9\sigma_1^2+1\sigma_2^2\\
&= 5
\end{aligned}
$$

Para função de autocorrelação de $x(n)$, sabe-se que as amostras são descorrelacionadas e o processo é de média nula, então o produto de diversos termos igual a zero também é zero. Temos que:

$$
\begin{aligned}
r_x(\tau) &= \mathbb{E}\{x(n)x(n+\tau)\} = \mathbb{E}\{[v_1(n)+3v_2(n-1)][v_1(n+\tau)+3v_2(n-1+\tau)]\},\\
&= \mathbb{E}\{v_1(n)v_1(n+\tau)+3v_1(n)v_2(n-1+\tau)+3v_2(n-1)v_1(n+\tau)+9v_2(n-1)v_2(n-1+\tau)\}\\
&= \vdots \quad \text{(Mesmo passo a passo do problema 1.1)}\\
&= 0
\end{aligned}
$$

Para $y(n)$, o processo é o mesmo, consquentemente $r_y(\tau)=0$.

Finalmente, observa-se que estatísticas de primeira e de segunda ordem são independentes do tempo para ambos, i.e, os dois processos são **WSS**.

Função de correlação cruzada

Para obter a função de correlação cruzada, basta aplicar as premissas utilizadas anteriormente: I) Processo de ruído branco é descorrelacionado; II) Média nula.

$$
\begin{aligned}
r_{x,y}(n_1,n_0) &= \mathbb{E}\{[x(n_1)y^*(n_0)]\}\\
&= \mathbb{E}\{[v_1(n_1)+3v_2(n_1-1)][v_2(n_0+1)+3v_1(n_0-1)]^*\},\\
&= \mathbb{E}\{v_1(n_1)v_2^*(n_0+1)+3v_1(n_1)v_1^*(n_0-1)+3v_2(n_1-1)v_2^*(n_0+1)+9v_2(n_1-1)v_1^*(n_0-1)\}\\
&= 0
\end{aligned}
$$

Esta função também é igual a zero, $r_{x,y}(n_1,n_0)=0$. Isto implica que os processos são conjuntamente estacionários, pois há independência do tempo da função, e por partir de processos de ruído branco, processos WSS individualmente, isto sustenta os desenvolvimento acima.

1.3 Matriz de Autocorrelação

Um vetor aleatório bidimensional

Para garantir a existência da matriz de correlação, deve-se corresponder as seguintes premissas: I) $\mathbf{R_x} = \mathbf{R_x^H}$; II) $\mathbf{a^H R_{xa}} \geq 0$; III) $\mathbf{Ax} = \lambda\mathbf{x}, \forall\lambda \geq 0|\mathbf{x} \in \mathbb{R}$.

Assumindo um vetor aleatório bidimensional: $\mathbf{X} = (x_1, x_2)$, a existência de $\mathbf{R}$ e sua hermitiana, $\mathbf{R}^H$.

Os elementos da contra-diagonal da matriz hermitiana deve obdece para são equivalência simétrica: I) $\mathbb{E}\{[x_1 x_2^*]\} = \mathbb{E}\{[x_2 x_1^*]\}$ e II) $\mathbb{E}\{[x_2 x_1^*]\} = \mathbb{E}\{[x_1 x_2^*]\}$

Além disso, a vantagem de $\mathbb{E}\{\cdot\}$ ser um operador linear, garante que os resultados são de fato iguais, independente da ordem dos vetores.

Já a limitação dos autovalores está diretamente ao determinante da matriz, sendo esse maior que zero, o critério imposto é atingido, i.e, para o caso 2×2, o produto dos elementos da diagonal principal é maior que o produto dos elementos da contra-diagonal.

Processo estocástico estacionário escalar

Em processo semelhante ao exemplo anterior, assume-se um processo estocástico estacionário escalar do tipo $\mathbf{X}_{(t)} = x(t)$ e sua versão atrasada $\mathbf{X}_{(t+\tau)} = x(t+\tau)$. Dado a matriz $\mathbf{R}$ e sua hermitiana, $\mathbf{R}^H$.

Os elementos da contra-diagonal da matriz hermitiana deve obdece para são equivalência simétrica: I) $\mathbb{E}\{[x(t)x^*(t+\tau)]\} = \mathbb{E}\{[x(t+\tau)x^*(t)]\}$ e II) $\mathbb{E}\{[x(t+\tau)x^*(t)]\} = \mathbb{E}\{[x(t)x^*(t+\tau)]\}$.

Novamente, a vantagem do operador linear é conveniente para que independente da ordem, e igualdade na contra-diagonal, i.e, simetria.

Para os autovalores, o produto dos elementos da diagonal principal é maior que o produto dos elementos da contra-diagonal, i.e:

$$\mathbb{E}\{[x^2(t)]\}\mathbb{E}\{[x^2(t+\tau)]\} > \mathbb{E}\{[x(t)x^*(t+\tau)]]\}\mathbb{E}\{[x(t+\tau)]x^*(t)]\}$$

1.4 Matriz Definida Positiva

Assumindo a expressão que define matriz de autocorrelação e que existe sua inversa bem,

$$\begin{gathered}\mathbb{E}\{\mathbf{x}\mathbf{x}^{\mathrm{H}}\} = \mathbf{R}_x \\ \mathbb{E}\{\mathbf{x}\mathbf{x}^{\mathrm{H}}\}\mathbf{R}_x^{-1} = \mathbf{R}_x\mathbf{R}_x^{-1}\end{gathered}$$

A inversa pode adentrar o operador, enquanto do lado direito obtém-se uma matrix identidade

$$\mathbb{E}\{\mathbf{x}\mathbf{x}^{\mathrm{H}}\mathbf{R}_x^{-1}\} = \mathbf{I}_{N\times N}$$

Isto permite aplicar o traço da matriz e por meio da propriedade de permutação cíclica do operador, tem-se que:

$$\mathrm{Trace}\{\mathbb{E}\{\mathbf{x}\mathbf{x}^{\mathrm{H}}\mathbf{R}_x^{-1}\}\} = \mathrm{Tr}\{\mathbf{I}_{N\times N}\}$$

Observa-se que o traço da matriz identidade $\mathbf{I}_{N\times N}$ é justamente a soma dos elementos da diagonal, N.

$$\begin{aligned}\mathrm{Trace}\{\mathbb{E}\{\mathbf{x}^{\mathrm{H}}\mathbf{R}_x^{-1}\mathbf{x}\}\} &= \sum^{N} 1 \\ &= N\end{aligned}$$

1.5 Covariância e correlação

Expressão 1 Dado que a matriz de Covariância pode ser obtida por:

$$\begin{aligned} C_X &= \mathbb{E}\{[(x-\mu)(x-\mu)^H]\} \\ &= \mathbb{E}\{xx^H\} - \mathbb{E}\{x\mu^H\} - \mathbb{E}\{\mu x^H\} + \mathbb{E}\{\mu\mu^H\} \end{aligned}$$

Considerando que a matriz de correlação pode ser escrita como demonstrado no exercício 1.4:

$$\begin{aligned} C_X &= R_X - \mu^H\mathbb{E}\{[x]\} - \mu\mathbb{E}\{[x^H]\} + \mu\mu^H, \\ &= R_X - \mu\mu^H - \mu\mu^H + \mu\mu^H, \\ &= R_X - \mu\mu^H, \end{aligned}$$

Por fim, obtém-se que:

$$R_X = C_X + \mu\mu^H.$$

Expressão 2 As expressões de correlação cruzada são obtidos de forma análoga, tal que:

$$\begin{aligned} \mathbf{C_{xy}} &= \mathbb{E}\{[x-\mu_x][y-\mu_y]\}, && (1.1) \\ &= \mathbb{E}\{[xy]\} - \mathbb{E}\{[x\mu_y]\} - \mathbb{E}\{[\mu_x y]\} + \mathbb{E}\{[\mu_x\mu_y]\} && (1.2) \\ &= \mathbb{E}\{[xy]\} - \mu_y\mu_x - \mu_x\mu_y + \mu_x\mu_y && (1.3) \\ &= \mathbb{E}\{[xy]\} + \mu_x\mu_y, && (1.4) \\ &= \mu_x\mu_y && (1.5) \\ & && (1.6) \end{aligned}$$

De forma análoga, obtém que $\mathbf{C_{yx}} = -\mu_x\mu_y$, consquentemente

$$\begin{aligned} \mathbf{C_{xy}} + \mathbf{C_{yx}} &= \mu_x\mu_y - \mu_x\mu_y \\ &= 0 \end{aligned}$$

1.6 Função de autocorrelação

Função do Processo Como nos problemas anteriores, utiliza-se como premissa que os processos são descorrelacionados e tem média nula. A função é dada por $r_x = \mathbb{E}\{x(n)x^*(n)\}$, tal que

$$\begin{aligned} r_x &= \mathbb{E}\{[v_1(n) + 2v_1(n+1) + 3v_2(n-1)][v_1(n) + 2v_1(n+1) + 3v_2(n-1)]^*(n)\} \\ &= r_v(n,n) + 2r_v(n,n+1) + 2r_v(n+1,n) + 4r_v(n+1,n+1) + 9r_v(n-1,n-1) \end{aligned}$$

Observa-se que apenas termos onde a função degrau está presente permanecem, enquanto o restantes podem ser cancelados, de modo que:

$$\begin{aligned} r_x &= 2r_v(n,n+1) + 2r_v(n+1,n) \\ &= \delta(n-n-1) + \delta(n+1-n) \end{aligned}$$

Isto pode ser reorganizado, sendo $\tau = n_1 - n_2$, de modo que:

$$r_x(n_1, n_2) = \delta(\tau) + \delta(-\tau)$$

Há apenas um deslocamento temporal (τ) atrelado à correlação, logo o processo é WSS.

Matriz de Correlação

Utilizando as relações obtidas anteriormente, é possível observar que os únicos elementos não nulos pertencem à diagonal, onde $n_1 = n_2$, acarretando $\delta(0) + \delta(0) = 2$.

Considerando 8 amostras consecutivas, a matriz de correlação é dada por:

$$\mathbf{R_x} = 2 \times I_{8\times 8}$$

Isto é, uma matriz 8×8, onde apenas a diagonal é não nula, preenchida por 2.

Filtragem Adaptativa - TIP 7188

Prof. Dr. Charles Casimiro Cavalcante
Prof. Dr. Guilherme de Alencar Barreto

Período: 2022.2

Lista de Exercícios No. 1: Estatísticas de Segunda Ordem

1. **(Média e autocorrelação)** Determine a média e a função de autocorrelação para o processo aleatório

$$x(n) = v(n) + 3v(n-1)$$

em que $v(n)$ é uma sequência de variáveis aleatórias independentes com média μ e variância σ^2. $x(n)$ é estacionário? Justifique.

2. **(Processos estacionários)** Sejam os processos aleatórios $x(n)$ e $y(n)$ definidos por

$$x(n) = v_1(n) + 3v_2(n-1)$$

e

$$y(n) = v_2(n+1) + 3v_1(n-1)$$

em que $v_1(n)$ e $v_2(n)$ são processos de ruído branco independentes cada um com variância igual a $0,5$.

(a) Quais são as funções de autocorrelação de x e de y? Os processos são WSS?

(b) Qual é a função de correlação cruzada $r_{xy}(n_1, n_0)$? Estes processos são conjuntamente estacionários (no sentido amplo)? Justifique.

3. **(Matriz de autocorrelação)** Quais as condições que os elementos de uma matriz

$$\mathbf{R} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

devem satisfazer tal que $\mathbf{R}$ seja uma matriz de autocorrelação válida de

(a) Um vetor aleatório bidimensional?

(b) Um processo estocástico estacionário escalar?

4. **(Matriz definida positiva)** Assuma que a inversa $\mathbf{R}_{\mathbf{x}}^{-1}$ da matriz de autocorrelação de um vetor coluna N-dimensional exista. Mostre que

$$E\left\{\mathbf{x}^H \mathbf{R}_{\mathbf{x}}^{-1} \mathbf{x}\right\} = N$$

5. **(Covariância e correlação)** Mostre que as matrizes de correlação e covariância satisfazem as relações abaixo:

- $\mathbf{R}_{\mathbf{x}} = \mathbf{C}_{\mathbf{x}} + \boldsymbol{\mu}_{\mathbf{x}} \boldsymbol{\mu}_{\mathbf{x}}^H$
- $\mathbf{C}_{\mathbf{x}+\mathbf{y}} = \mathbf{C}_{\mathbf{x}} + \mathbf{C}_{\mathbf{xy}} + \mathbf{C}_{\mathbf{yx}} + \mathbf{C}_{\mathbf{y}}$, para $\mathbf{x}$ e $\mathbf{x}$ descorrelacionados

6. **(Função de autocorrelação)** Processos aleatórios $v_1(n)$ e $v_2(n)$ são independentes e têm a mesma função de correlação

$$r_v(n_1, n_0) = 0.5\delta(n_1 - n_0)$$

(a) Qual é a função de correlação do processo aleatório

$$x(n) = v_1(n) + 2v_1(n+1) + 3v_2(n-1)?$$

Este é um processo WSS? Justifique.

(b) Encontre a a matrix de correlação de um vetor aleatório consistindo de oito amostras consecutivas de $x(n)$.

2 Lista 2: Filtragem Linear Ótima

2.1 Filtragem Ótima

Coeficientes de Wiener

Considerando o problema de filtragem de Wiener, e assumindo conhecimento da matriz de correlação $\mathbf{R_X}$ e do vetor de correlação cruzada $\mathbf{p_X}_d$, pode-se obter os coeficientes de $\mathbf{w}$.

$$\mathbf{w} = \mathbf{R_X}^{-1}\mathbf{p_X}_d \tag{2.1}$$

Aplicando a equação 2.1, obtém-se o vetor de pesos do filtro.

$$\mathbf{R_X}^{-1} = \begin{bmatrix} 1.3333 & -0.6667 \\ -0.6667 & 1.3333 \end{bmatrix}$$

$$\begin{aligned} \mathbf{w} &= \begin{bmatrix} 1.3333 & -0.6667 \\ -0.6667 & 1.3333 \end{bmatrix} \begin{bmatrix} 0.5 \\ 0.25 \end{bmatrix} \\ &= \begin{bmatrix} 0.5 \\ 0 \end{bmatrix}. \end{aligned}$$

Erro Médio Quadrático

A partir do vetor de pesos, resultado de 2.1, basta aplicá-lo na equação do erro mínimo.

$$\mathbb{E}\{e^2(n)\} = \sigma_d^2 - 2\mathbf{w}^\top \mathbf{p_X}_d + \mathbf{w}^\top \mathbf{R_X}\mathbf{w} \tag{2.2}$$

$$\begin{aligned} e &= \sigma_d^2 - 2\begin{bmatrix} 0.5 & 0.0 \end{bmatrix} \begin{bmatrix} 0.5 \\ 0.25 \end{bmatrix} + \begin{bmatrix} 0.5 & 0.0 \end{bmatrix} \begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix} \begin{bmatrix} 0.5 \\ 0.0 \end{bmatrix} \\ &= \sigma_d^2 - 2 \times 0.25 + 0.25 \\ &= \sigma_d^2 - 0.25 \end{aligned}$$

Representação em Autovalores

A decomposição em valores singulares (EVD) pode ser aplicada na matriz de correlação

$$\mathbf{R}_X = \mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^{-1} \tag{2.3}$$

Aplicando diretamente o resultado da EVD 2.3 na equação do filtro ótimo 2.1, obtém-se:

$$\mathbf{w} = (\mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^{-1})^{-1}\mathbf{p_X}_d \tag{2.4}$$

Finalmente, o resultado é uma expressão que compreende a inversão de matrizes menos custosas computacionalmente. Isto se dá principalmente por $\boldsymbol{\Lambda}$ ser uma matriz diagonal contendo os autovalores da matriz de autocorrelação, bastando calcular $1/\lambda_i$ para obter a sua inversa.

$$\mathbf{w} = \mathbf{Q}^{-1}\boldsymbol{\Lambda}^{-1}\mathbf{Q}\mathbf{p_X}_d. \tag{2.5}$$

2.2 Erro Médio Quadrático Mínimo

Para verificar a expressão proposta, é necessário obter a matriz de correlação do vetor aumentado.

$$\begin{aligned}\mathbf{A} &= \mathbb{E}\left\{\begin{bmatrix} d(n) \\ x(n)\end{bmatrix}\begin{bmatrix} d(n)^\top & x(n)^\top\end{bmatrix}\right\} \\ &= \begin{bmatrix}\mathbb{E}\{d(n)d(n)^\top\} & \mathbb{E}\{d(n)x(n)^\top\} \\ \mathbb{E}\{x(n)d(n)^\top\} & \mathbb{E}\{x(n)x(n)^\top\}\end{bmatrix} \\ &= \begin{bmatrix}\sigma_d^2 & \mathbf{p}_{\mathbf{X}d}^\top \\ \mathbf{p}_{\mathbf{X}d} & \mathbf{R}_X\end{bmatrix}\end{aligned}$$

É conveniente observar que ao desenvolver a equação, os elementos resultantes da expressão são todo conhecidos. Multiplicando o resultado obtido pelo vetor $\begin{bmatrix} 1 \\ -w\end{bmatrix}$ à direita e assumindo o modelo nas condições de filtragem ótima, dado filtro de wiener, onde, $\mathbf{w}_{\text{opt}} = \mathbf{R}_X^{-1}\mathbf{p}_{\mathbf{X}d}$, temos que:

$$\begin{aligned}\mathbf{A}\begin{bmatrix} 1 \\ -\mathbf{w}\end{bmatrix} &= \begin{bmatrix}\sigma_d^2 & \mathbf{p}_{\mathbf{X}d}^\top \\ \mathbf{p}_{\mathbf{X}d} & \mathbf{R}_X\end{bmatrix}\begin{bmatrix} 1 \\ -\mathbf{w}\end{bmatrix} \\ &= \begin{bmatrix}\sigma_d^2 - \mathbf{p}_{\mathbf{X}d}^\top\mathbf{w} \\ \mathbf{p}_{\mathbf{X}d} - \mathbf{R}_X\mathbf{w}\end{bmatrix} \\ &= \begin{bmatrix}\sigma_d^2 - \mathbf{p}_{\mathbf{X}d}^\top\mathbf{R}_X^{-1}\mathbf{p}_{\mathbf{X}d} \\ \mathbf{p}_{\mathbf{X}d} - \mathbf{R}_X\mathbf{R}_X^{-1}\mathbf{p}_{\mathbf{X}d}\end{bmatrix} \\ &= \begin{bmatrix}\sigma_d^2 - \mathbf{p}_{\mathbf{X}d}^\top\mathbf{R}_X^{-1}\mathbf{p}_{\mathbf{X}d} \\ \mathbf{p}_{\mathbf{X}d} - \mathbf{I}_X\mathbf{p}_{\mathbf{X}d}\end{bmatrix} \\ &= \begin{bmatrix}\sigma_d^2 - \mathbf{p}_{\mathbf{X}d}^T\mathbf{R}_X^{-1}\mathbf{p}_{\mathbf{X}d} \\ 0\end{bmatrix}\end{aligned}$$

Finalmente, dado a equação obtida, com expressão equivalente à J_{min}, pode-se escrever a relação proposta.

$$\mathbf{A}\begin{bmatrix} 1 \\ -\mathbf{w}\end{bmatrix} = \begin{bmatrix} J_{min} \\ 0\end{bmatrix}$$

2.3 Cancelamento de Ruído

Formulando a expressão do erro a partir do sistema sugerido:

$$\begin{aligned}e(n) &= x(n) - \hat{v}_1 \\ &= x(n) - \mathbf{w}^T v_2(n)\end{aligned}$$

Dado a equação obtida acima, deve-se aplicar: I) função erro quadrático médio (MSE), II) o operador valor esperado, com o filtro apresentando coeficientes constantes.

$$\begin{aligned}\mathbb{E}\{e^2(n)\} &= \mathbb{E}\{x^2(n)\} - 2\mathbf{w}^T\mathbb{E}\{x(n)v_2(n)\} + \mathbf{w}^T\mathbb{E}\{v_2(n)v_2(n)^T\}\mathbf{w}, \\ &= \sigma_x^2 - 2\mathbf{w}^T\mathbf{p}_{xv_2} + \mathbf{w}^T\mathbf{R}_{v_2}\mathbf{w}.\end{aligned}$$

Isto permite encontrar a equação para calculcar o $\mathbf{w}$ que minimiza o MSE via gradiante.

$$\begin{aligned}\nabla_{\mathbf{w}}\mathbb{E}\{e^2(n)\} &= 0\\ -2\mathbf{p}_{xv_2} + 2\mathbf{R}_{v_2}\mathbf{w} &= 0\\ -\mathbf{p}_{xv_2} + \mathbf{R}_{v_2}\mathbf{w} &= 0\end{aligned}$$

Por fim, temos que $\mathbf{w} = \mathbf{R}_{v_2}^{-1}\mathbf{p}_{xv_2}$ é o vetor de pesos do filtro.

2.4 Predição Ótima

O primeiro passo é definir a matriz de autocorrelação para $x(n)$. Visando a simplicidade, mas sem perda de generalidade, pode-se considerar que o processo S é WSS e tem variância σ_s^2:

$$\mathbf{R}_x = \begin{bmatrix} \mathbb{E}\{x(n)x^*(n)\} & \mathbb{E}\{x(n-1)x^*(n)\} \\ \mathbb{E}\{x(n)x^*(n-1)\} & \mathbb{E}\{x(n-1)x^*(n-1)\} \end{bmatrix}$$

Dado as premissas assumidas, a matriz pode ser simplificada para uma matriz diagonal preenchido por $2\sigma_s^2$.

Finalmente, considerando média nula para o processo D, a consequencia é que o vetor de correlação cruzada também é nulo.

$$\begin{aligned}\mathbf{w}_{\text{opt}} &= \mathbf{R}_x^{-1}\mathbf{p}_{xd}\\ &= \begin{bmatrix} 2\sigma_s^2 & 0 \\ 0 & 2\sigma_s^2 \end{bmatrix}\begin{bmatrix} 0 \\ 0 \end{bmatrix}\\ &= \begin{bmatrix} 0 \\ 0 \end{bmatrix}\end{aligned}$$

Isto implica que o filtro linear ótimo para esse processo seria o próprio vetor nulo.

2.5 Superfície de Erro

Dado os coeficientes, temos que a matriz de correlação $\mathbf{R}_x$ do filtro ótimo é uma matriz identidade de ordem 2×2.

Aplicando a solução do filtro ótimo de Wiener, obtém-se finalmente o vetor de pesos:

$$\begin{aligned}\mathbf{w}_{\text{opt}} &= \mathbf{R}_x^{-1}\mathbf{p}_{xd}\\ &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 2 \\ 4.5 \end{bmatrix}\\ &= \begin{bmatrix} 2 \\ 4.5 \end{bmatrix}\end{aligned}$$

A superfície definida por $J(\mathbf{w})$

Abrindo a equação do erro médio, para obter a expressãoque define a superfície.

$$\begin{aligned}\mathbf{J}(w) &= \mathbb{E}\{e^2(n)\} && (2.6)\\ &= \sigma_d^2 - 2\mathbf{w}^T\mathbf{p}_{xd} + w^T\mathbf{R}_X\mathbf{w}. && (2.7)\end{aligned}$$

Aplicando os valores obtidos na expressão da superfície:

$$\begin{aligned}\mathbf{J}(w_0, w_1) &= 24.40 - 2\begin{bmatrix} w_0 w_1 \end{bmatrix}\begin{bmatrix} 2 \\ 4.5 \end{bmatrix} + \begin{bmatrix} w_0 w_1 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} w_0 \\ w_1 \end{bmatrix} \\ &= 24.40 - 4w_0 - 9w_1 + w_0^2 + w_1^2\end{aligned}$$

Utilizando um MATLAB é possível obter a Figura 2.1 onde é traçada a superfície de erro MSE expressada na Equação (2.7).

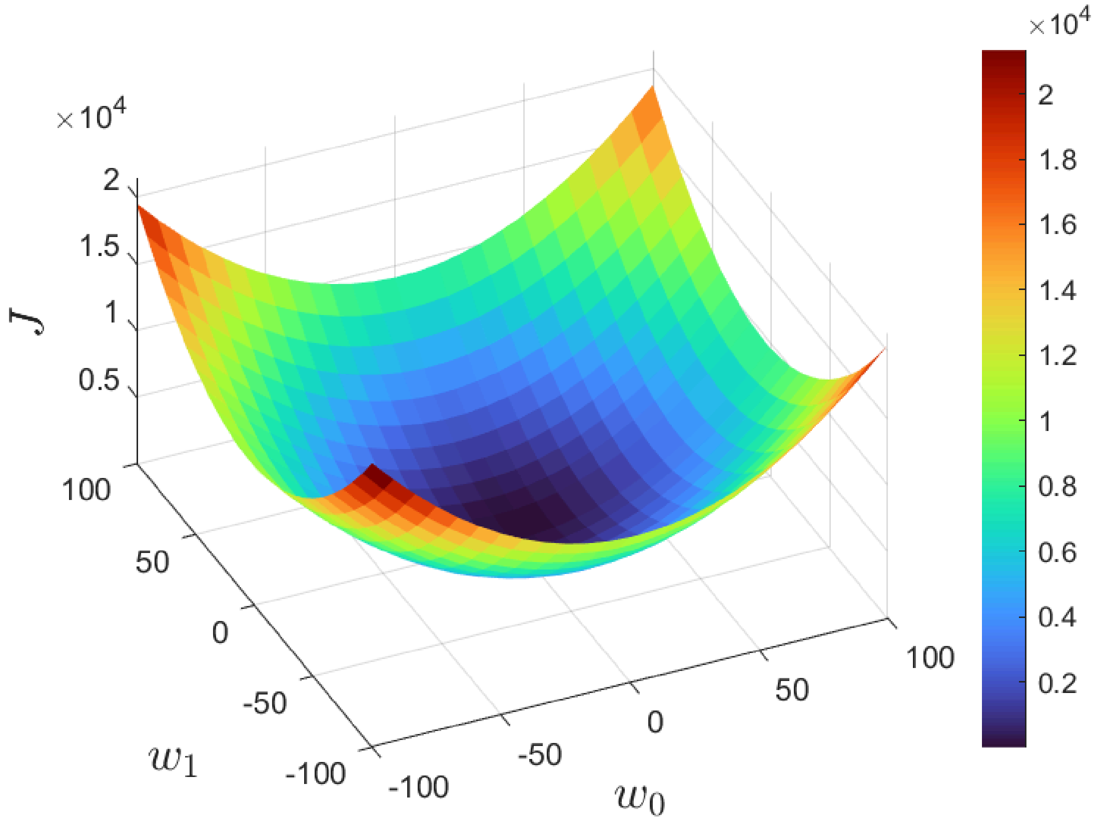

Figura 2.1: Superfície de erro $J(w_0, w_1)$.

Filtragem Adaptativa - TIP 7188

Prof. Dr. Charles Casimiro Cavalcante
Prof. Dr. Guilherme de Alencar Barreto

Período: 2022.2

Lista de Exercícios No. 2: Filtragem Linear Ótima

1. (Filtragem ótima) Considere um problema de filtragem de Wiener conforme caracterizado a seguir. A matriz de correlação $\mathbf{R_x}$ de um vetor de entrada $\mathbf{x}(n)$ é dada por

$$\mathbf{R_x} = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}.$$

O vetor de correlação cruzada $\mathbf{p}_{\mathbf{x}d}$ entre o vetor de entrada $\mathbf{x}$ e a resposta desejada $d(n)$ é

$$\mathbf{p}_{\mathbf{x}d} = \begin{bmatrix} 0.5 \\ 0.25 \end{bmatrix}$$

(a) Encontre o vetor de coeficientes do filtro de Wiener.

(b) Qual é o mínimo erro médio quadrático fornecido por este filtro?

(c) Formule uma representação do filtro de Wiener em termos dos autovalores da matriz $\mathbf{R_x}$ e de seus autovetores associados.

2. (Erro médio quadrático mínimo) Mostre que a equação do erro mínimo pode se escrita da seguinte maneira:

$$\mathbf{A} \begin{bmatrix} 1 \\ -\mathbf{w} \end{bmatrix} = \begin{bmatrix} J_{\min} \\ \mathbf{0} \end{bmatrix}$$

em que $J_{\min}$ é o mínimo erro médio quadrático, $\mathbf{w}$ é o filtro de Wiener, e $\mathbf{A}$ é a matriz de correlação do vetor aumentado

$$\begin{bmatrix} d(n) \\ \mathbf{x}(n) \end{bmatrix}$$

em que $d(n)$ é o sinal desejado e $\mathbf{x}(\mathbf{n})$ é o sinal de entrada do filtro de Wiener.

3. (Cancelamento de ruído) Em várias aplicações práticas há uma necessidade de cancelar ruído que foi adicionado a um sinal. Por exemplo, se estamos usando o telefone celular dentro de um ruído e o ruído do carro ou rádio é adicionado à mensagem que estamos tentando transmitir. A Figura 1 ilustra as situações de contaminação de ruído. Calcule o filtro de Wiener (filtro ótimo) de tal configuração em relação às estatísticas dos sinais envolvidos que você dispõe (conhece).

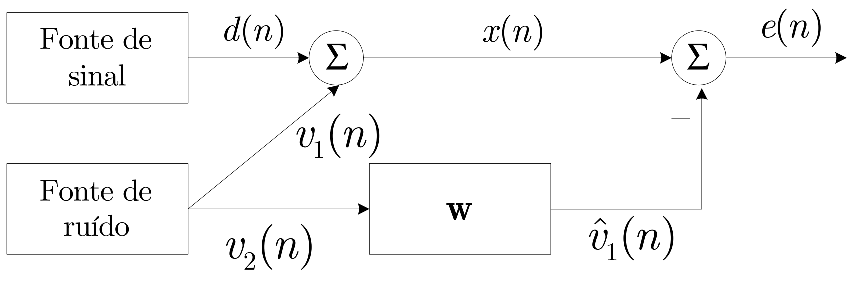

Figure 1: Esquema de cancelamento de ruído.

4. **(Predição ótima)** Seja um processo estocástico dado por

$$x(n) = s(n + a) + s(n - 4a),$$

em que $S(n)$ é um processo estocástico WSS dado e a é uma constante.

Deseja-se filtrar o processo de tal forma obter-se um processo $D(s) = s(n - a)$, o qual também sabe-se que é um processo WSS. Suponha que o sinal $d(n)$ possua média nula e variância unitária.

(a) Calcule o filtro, com dois coeficientes, que fornece a solução ótima em relação ao erro médio quadrático.

(b) Calcule o preditor direto ótimo de passo unitário, com dois coeficientes, que fornece a solução ótima em relação ao erro médio quadrático.

(c) Compare as soluções dos dois.

5. **(Superfície de erro)** Suponha que foram encontrados os seguintes coeficientes de autocorrelação: $r_x(0) = 1$ e $r_x(1) = 0$. Tais coeficientes foram obtidos de amostras corrompidas com ruído. Além disso, a variância do sinal desejado é $\sigma_d^2 = 24.40$ e o vetor de correlação cruzada é $\mathbf{p}_{\mathbf{x}d} = [2 \;\; 4.5]^T$. Encontre:

(a) O valor dos coeficientes do filtro de Wiener.

(b) A superfície definida por $J(\mathbf{w})$. Faça um gráfico da mesma.

3 Implementações em MATLAB

A implementação é dividida em dois arquivos. O primeiro é chamado *filter_hw.m*[1], onde há a definição dos métodos utilizados nos problemas. O segundo é o script *main.m*[2], que chama os métodos para serem executados. São apresentados nas sessões 3.1 e 3.2, respectivamente.

[1] *filter_hw.m*: `https://github.com/lucasabdalah/Courses-HWs/blob/master/Master/TIP7188-FILTRAGEM_ADAPTATIVA/homework/code/filter_hw.m`

[2] *main.m*: `https://github.com/lucasabdalah/Courses-HWs/blob/master/Master/TIP7188-FILTRAGEM_ADAPTATIVA/homework/code/main.m`.

Table of Contents

MÉTODOS

[TIP7188 - Filtragem Adaptativa] Author: Lucas Abdalah

filter_hw.m

filter_hw is a package developped for the Adaptative Filtering Course It is a way to make a compilation for all function

CONTENT

```
HOMEWORK 2 - PROBLEM 5

SAVE DATA TO TXT FILE
    filter_hw.MAT2TXT       - Write a matrix X into a txt file
    filter_hw.TENSOR2TXT    - Write a 3D tensor X into a txt file

PLACE HOLDER

classdef filter_hw

methods(Static)
```

HOMEWORK 2 - PROBLEM 5

```
function hw2p5(varargin)
% FILTER_HW.HW2P5  Perfom the error surface propose on the Hw 2,
 problem 5
%
%   See also.

    if isempty(varargin)
        save_results = false;
    else
        save_results = varargin{1};
    end

    N = 25;
```

```
    w_lim = 100;
    w = [linspace(-w_lim,w_lim,N); linspace(-w_lim,w_lim,N)];
    [w_0, w_1] = meshgrid(w(1,:), w(2,:));
    J_surface = @(w_0, w_1) 24.40 - 4.*w_0 - 9.*w_1 + w_0.^2 + w_1.^2;
    J = J_surface(w_0, w_1);
    h = figure();
    surf(w_0, w_1, J, 'EdgeColor', 'none');
    colormap turbo;
    xlabel('$w_0$', 'FontSize', 16, 'interpreter', 'latex');
    ylabel('$w_1$', 'FontSize', 16, 'interpreter', 'latex');
    zlabel('$J$', 'FontSize', 16, 'interpreter', 'latex');
    view([-24.5036297640653 47.6514617014408]);
    colorbar('box', 'off');
    grid on;
    axis tight;
    pathName = 'figures/';
    filter_hw.export_fig(save_results, h, [pathName, 'hw2p5']);
end
```

HOMEWORK 3 - PROBLEM 4

```
function filter_path(signal_d_var, weights, wiener, Rx, p, c_)
% FILTER_HW.FILTER_PATH  Perfom the weights path surface
%
%   See also.
    step = 0.25;
    X = meshgrid (-1:step:1,-1:step:1);
    w = [X(:), reshape(transpose(X),[],1)];
    [wLen, ~] = size(w);
    J = zeros(wLen, 1);
    for n = 1:wLen
        J(n) = signal_d_var - 2*w(n,:)*p + w(n,:)*Rx*w(n,:).';
    end
    contour(X, X', reshape(J,size(X)), '-.', 'color', 'k');
    hold on;
    scatter(weights(1,:), weights(2, :), '.', 'MarkerEdgeColor', c_);
    hold off;
    ha = annotation('textarrow', [0 0], [0 0],'String', 'Wiener');
    ha.Parent = gca;
    ha.X = [wiener(1)+0.15 wiener(1)];
    ha.Y = [wiener(2)-0.4 wiener(2)];
    grid on;

end

function [error, weights] = dga(signal_x, signal_d, order, mi, Rx, p)
% FILTER_HW.DGA  Perfom the Deterministic Gradient Algorithm
%
%   See also.
    N = length(signal_x);
    error = zeros(N,1);
    weights = zeros(order, N);
    signal_d = signal_d(order:end,1);
```

```
    for n = 1:(N - order - 1)
        error(n,1) = signal_d(n) - weights(:,n)'*signal_x(n:n
+order-1);
        weights(:,n+1) = weights(:,n) - 2*mi*(Rx*weights(:,n) - p);
    end
end


function [error, weights] = lms(signal_x, signal_d, order, mi)
% FILTER_HW.LMS  Perfom the LMS Algorithm
%
%   See also.
    N = length(signal_x);
    error = zeros(N,1);
    weights = zeros(order, N);
    signal_d = signal_d(order:end,1);

    for n = 1:(N - order - 1)
        error(n) = signal_d(n) - weights(:,n)' * signal_x(n:n
+order-1);
        weights(:,n+1) = weights(:,n) + 2 * mi * error(n) *
 signal_x(n:n+order-1);
    end
    weights = flip(weights);
end


function [error, weights] = newton(signal_x, signal_d, order, mi,
 wiener)
% FILTER_HW.NEWTON  Perfom the Newton Algorithm
%
%   See also.
    N = length(signal_x);
    error = zeros(N,1);
    weights = zeros(order, N);
    signal_d = signal_d(order:end,1);

    for n = 1:(N - order - 1)
        error(n,1) = signal_d(n) - weights(:,n)'*signal_x(n:n
+order-1);
        weights(:,n+1) = weights(:,n) - mi*(weights(:,n) - wiener);
    end
end


function [error, weights] = nlms(signal_x, signal_d, order, mi, gamma)
% FILTER_HW.NLMS  Perfom the NLMS Algorithm
%
%   See also.
    N = length(signal_x);
    error = zeros(N,1);
    weights = zeros(order, N);
    signal_d = signal_d(order:end,1);
```

```
    for n = 1:(N - order - 1)
        mi_normalized = mi/(gamma + norm(signal_x));
        error(n) = signal_d(n) - weights(:,n)' * signal_x(n:n +
 order-1);
        weights(:,n+1) = weights(:,n) +  2 * mi_normalized * error(n)
 * signal_x(n:n+order-1);
    end
    weights = flip(weights);
end


function hw3p4(varargin)

    % Save or not the results
    if isempty(varargin)
        save_results = false;
    else
        save_results = varargin{1};
    end

    pathName = 'figures/';

    h0 = figure();
    viscircles([0, 0], 1,'Color','k', 'LineStyle','-', 'LineWidth',
 1.5);
    line([0 0],[-1 1], 'Color', 'k', 'HandleVisibility','off');
    line([-1 1], [0 0], 'Color', 'k', 'HandleVisibility','off');
    hold on
    scatter(-1.6, 0, 'o', 'filled');
    scatter(0.45, 0, 'o', 'filled');
    hold off
    xlabel('$\Re (Z)$', 'interpreter', 'latex');
    ylabel('$\Im (Z)$', 'interpreter', 'latex');
    axis([-1.7 1.7 -1.7 1.7]);
    legend('Channel Zeros', 'Filter Zeros', 'Location', 'Northeast');
    grid minor
    axis square

    filter_hw.export_fig(save_results, h0, [pathName, 'hw3p4-zeros']);


    % Color scheme to plot -----------------------------------
    c_ = struct('dg', [57 106 177]./255, 'lms', [204 37
 41]./255, 'newton', [62 150 81]./255, 'nlms', [107 76
 154]./255, 'mean', 'k');
    % General Setup ------------------------------------
    N = 1000; % Number of samples
    order = 2; % Filter order
    % Signal Model ------------------------------------
    signal_d = randn(N,1);
    signal_d_var = var(signal_d);
    % Noisy Version ------------------------------------
    Hz = [1 1.6];
```

```
   signal_x = filter(Hz,1,signal_d);
   noise = sqrt(1/(10^(inf/10))).*randn(N,1);
   signal_x = signal_x + noise;
   % Wiener Filter ------------------------------------
   Rxcorr = sort(xcorr(Hz));
   Rx = reshape([Rxcorr(end) Rxcorr], [2, 2]); % Autocorrelation
matrix
   p = eye(2,1); % Cross-correlation
   wiener = Rx\p; % Wiener solution
   fprintf('Wiener solution: %2.2f \n %2.2f \n', wiener);
   % Deterministic Gradient Algorithm
------------------------------------
   dg.mi = 1e-2;
   [dg.error, dg.weights] = filter_hw.dga(signal_x, signal_d, order,
dg.mi, Rx, p);
   % Newton Implementation ------------------------------------
   newton.mi = 5e-2;
   [newton.error, newton.weights] = filter_hw.newton(signal_x,
signal_d, order, newton.mi, wiener);
   % LMS Algorithm ------------------------------------
   lms.mi = 1e-3;
   [lms.error, lms.weights] = filter_hw.lms(signal_x, signal_d,
order, lms.mi);
   % NLMS Algorithm ------------------------------------
   nlms.mi = 5e-1;
   gamma = 0.5;
   [nlms.error, nlms.weights] = filter_hw.nlms(signal_x, signal_d,
order, nlms.mi, gamma);
   % Plot - Deterministic Gradient Algorithm
------------------------------------
   h1 = figure(1);
   subplot(2,1,1);
   semilogy(1:N, dg.error.^2,'-','color', c_.dg , "linewidth", 1); %
MSE
   hold on
   semilogy(1:N, repelem(mean(dg.error.^2), N), '--', 'color',
c_.mean, "linewidth", 1);
   hold off
   xlabel('Iterations');
   ylabel('MSE');
   legend('Deterministic Gradient', 'Mean', 'Location', 'Best')
   grid on;
   axis tight
   subplot(2,1,2);
   filter_hw.filter_path(signal_d_var, dg.weights, wiener, Rx, p,
c_.dg); % Solution Path
   xlabel('$w_1$', 'interpreter', 'latex');
   ylabel('$w_0$', 'interpreter', 'latex');
   legend('Solution Contour', 'Deterministic
Gradient', 'Location', 'Northeast')
   axis tight

   filter_hw.export_fig(save_results, h1, [pathName, 'hw3p4-dga']);
```

```
    % Plot - Newton Implementation
 ------------------------------------
    h2 = figure(2);
    subplot(2,1,1);
    semilogy(1:N, newton.error.^2,'-','color', c_.newton, "linewidth",
 1); % MSE Curve
    hold on
    semilogy(1:N, repelem(mean(newton.error.^2), N), '--', 'color',
 c_.mean, "linewidth", 1);
    hold off
    xlabel('Iterations');
    ylabel('MSE');
    legend('Newton', 'Mean', 'Location', 'Best');
    grid on;
    axis tight
    subplot(2,1,2);
    filter_hw.filter_path(signal_d_var, newton.weights, wiener, Rx, p,
 c_.newton);
    xlabel('$w_1$', 'interpreter', 'latex');
    ylabel('$w_0$', 'interpreter', 'latex');
    legend('Solution Contour', 'Newton',  'Location', 'Northeast')
    axis tight

    filter_hw.export_fig(save_results, h2, [pathName, 'hw3p4-
newton']);

    % Plot - LMS Algorithm ------------------------------------
    h3 = figure(3);
    subplot(2,1,1);
    semilogy(1:N, lms.error.^2,'-','color', c_.lms , "linewidth",
 1); % MSE
    hold on
    semilogy(1:N, repelem(mean(lms.error.^2), N), '--', 'color',
 c_.mean, "linewidth", 1);
    hold off
    xlabel('Samples, N');
    ylabel('MSE');
    legend('LMS', 'Mean', 'Location', 'Best')
    grid on;
    axis tight
    subplot(2,1,2);
    filter_hw.filter_path(signal_d_var, lms.weights, wiener, Rx, p,
 c_.lms); % Solution Path
    xlabel('$w_1$', 'interpreter', 'latex');
    ylabel('$w_0$', 'interpreter', 'latex');
    legend('Solution Contour', 'LMS',  'Location', 'Northeast')
    axis tight

    filter_hw.export_fig(save_results, h3, [pathName, 'hw3p4-lms']);

    % Plot - NLMS Implementation ------------------------------------
    h4 = figure(4);
    subplot(2,1,1);
    semilogy(1:N, nlms.error.^2,'-','color', c_.nlms, "linewidth", 1);
```

```
    hold on
    semilogy(1:N, repelem(mean(nlms.error.^2), N), '--', 'color',
 c_.mean, "linewidth", 1);
    hold off
    xlabel('Samples, N');
    ylabel('MSE');
    legend('NLMS','Mean', 'Location', 'Best');
    grid on;
    axis tight
    subplot(2,1,2);
    filter_hw.filter_path(signal_d_var, nlms.weights, wiener, Rx, p,
 c_.nlms);
    xlabel('$w_1$', 'interpreter', 'latex');
    ylabel('$w_0$', 'interpreter', 'latex');
    legend('Solution Contour', 'NLMS',  'Location', 'Northeast')
    axis tight

    filter_hw.export_fig(save_results, h4, [pathName, 'hw3p4-nlms']);

end
```

HOMEWORK 3 - PROBLEM 5

```
function [error, weights, signal_d_hat] = hw3p5_lms(signal_x,
 signal_d, M, mi)
    N = length(signal_x);
    error = zeros(N,1);
    weights = zeros(M, N);
    signal_d_hat = zeros(size(signal_x));

    for ss = 1:(N - M)
        signal_d_hat(ss) = weights(:,ss)'*signal_x(ss:ss+M-1);
        error(ss) = signal_d(ss) - weights(:,ss)' * signal_x(ss:ss
+M-1);
        weights(:,ss+1) = weights(:,ss) + 2 * mi * error(ss) *
 signal_x(ss:ss+M-1);
    end
    signal_d_hat = zscore(signal_d_hat);
end


function hw3p5(varargin)
    % General Setup
    c_ = struct('original', [57 106 177]./255, 'estimated', [204 37
 41]./255, 'lms', [107 76 154]./255, 'mean', 'k');
    order = 15; M = order + 1;
    N = 5000 + M; % Number of samples
    mi_ceil = 1/97;

    % Signal Model
    SNR_dB = 30;
    SNR_li = 10^(SNR_dB/10);
    variance_noise = 1/SNR_li;
```

```
    noise = sqrt(variance_noise).*randn(N,1);
    signal_d = zscore(randn(N,1)); % Z-score Normalization
    Hz = ones(1,12);
    signal_x = filtfilt(Hz,1,signal_d);
    signal_x = zscore(signal_x + noise);

    [mu02.error, mu02.weights, mu02.signal_d_hat] =
filter_hw.hw3p5_lms(signal_x, signal_d, M, mi_ceil/2);
    [mu10.error, mu10.weights, mu10.signal_d_hat] =
filter_hw.hw3p5_lms(signal_x, signal_d, M, mi_ceil/10);
    [mu50.error, mu50.weights, mu50.signal_d_hat] =
filter_hw.hw3p5_lms(signal_x, signal_d, M, mi_ceil/50);

    % Plot - mu/2
    h1 = figure();
    subplot(3,1,1)
    semilogy(1:N, abs(mu02.error).^2,'-','color',
c_.lms , "linewidth", 1);
    hold on
    semilogy(1:N, repelem(mean(abs(mu02.error).^2), N),'--','color',
c_.mean , "linewidth", 1);
    hold off
    xlabel('Samples, N');
    xlim([0 N]);
    ylabel('MSE');
    legend('LMS', 'Mean', 'Location', 'Best');
    title('$\mu_{\max}/2$', 'interpreter', 'latex')
    grid on;

    subplot(3,1,2)
    [Hf,wf] = freqz(mu02.weights(:, N - M + 1).',1, 'whole', 512);
    [Hc,wc] = freqz(ones(1,12), 1, 'whole', 512);
    plot(wc/pi,20*log10(abs(Hc)), '--', 'color',
c_.original, "linewidth", 1.5);
    hold on;
    plot(wf/pi,20*log10(abs(Hf)), '-', 'color',
c_.estimated, "linewidth", 1.5);
    xlabel('Normalized Frequency (\times\pi rad/sample)')
    ylabel('Magnitude (dB)')
    legend('System', 'Filter', 'Location', 'Best');
    grid on;

    subplot(3,1,3)
    plot(1:N, signal_d, '--','color', c_.original, "linewidth", 1.5);
    hold on;
    plot(1:N, mu02.signal_d_hat, '-', 'color',
c_.estimated, "linewidth", 1.5);
    xlabel('Samples, N');
    xlim([1000 1050]);
    ylabel('Magnitude');
    legend('Original', 'Estimated', 'Location', 'Best');
    grid on;
    % savefig_tight(h1, 'figures/hw3p5b-mu02', 'both');
```

```
% Plot - mu/10
h2 = figure();
subplot(3,1,1)
semilogy(1:N, abs(mu10.error).^2,'-','color',
c_.lms , "linewidth", 1);
hold on
semilogy(1:N, repelem(mean(abs(mu10.error).^2), N),'--','color',
c_.mean , "linewidth", 1);
hold off
xlabel('Samples, N');
xlim([0 N]);
ylabel('MSE');
legend('LMS', 'Mean', 'Location', 'Best');
title('$\mu_{\max}/10$', 'interpreter', 'latex')
grid on;

subplot(3,1,2)
[Hf,wf] = freqz(mu10.weights(:, N - M + 1).',1, 'whole', 512);
[Hc,wc] = freqz(ones(1,12), 1, 'whole', 512);
plot(wc/pi,20*log10(abs(Hc)), '--', 'color',
c_.original, "linewidth", 1.5);
hold on;
plot(wf/pi,20*log10(abs(Hf)), '-', 'color',
c_.estimated, "linewidth", 1.5);
xlabel('Normalized Frequency (\times\pi rad/sample)')
ylabel('Magnitude (dB)')
legend('System', 'Filter', 'Location', 'Best');
grid on;

subplot(3,1,3)
plot(1:N, signal_d, '--','color', c_.original, "linewidth", 1.5);
hold on;
plot(1:N, mu10.signal_d_hat, '-', 'color',
c_.estimated, "linewidth", 1.5);
xlabel('Samples, N');
xlim([4000 4050]);
ylabel('Magnitude');
legend('Original', 'Estimated', 'Location', 'Best');
grid on;
% savefig_tight(h2, 'figures/hw3p5b-mu10', 'both');

% Plot - mu/50
h3 = figure();
subplot(3,1,1)
semilogy(1:N, abs(mu50.error).^2,'-','color',
c_.lms , "linewidth", 1);
hold on
semilogy(1:N, repelem(mean(abs(mu50.error).^2), N),'--','color',
c_.mean , "linewidth", 1);
hold off
xlabel('Samples, N');
xlim([0 N]);
ylabel('MSE');
legend('LMS', 'Mean', 'Location', 'Best');
```

```
    title('$\mu_{\max}/50$', 'interpreter', 'latex')
    grid on;

    subplot(3,1,2)
    [Hf,wf] = freqz(mu50.weights(:, N - M + 1).',1, 'whole', 512);
    [Hc,wc] = freqz(ones(1,12), 1, 'whole', 512);
    plot(wc/pi,20*log10(abs(Hc)), '--', 'color',
c_.original, "linewidth", 1.5);
    hold on;
    plot(wf/pi,20*log10(abs(Hf)), '-', 'color',
c_.estimated, "linewidth", 1.5);
    xlabel('Normalized Frequency (\times\pi rad/sample)')
    ylabel('Magnitude (dB)')
    legend('System', 'Filter', 'Location', 'Best');
    grid on;

    subplot(3,1,3)
    plot(1:N, signal_d, '--','color', c_.original, "linewidth", 1.5);
    hold on;
    plot(1:N, mu50.signal_d_hat, '-', 'color',
c_.estimated, "linewidth", 1.5);
    xlabel('Samples, N');
    xlim([4000 4050]);
    ylabel('Magnitude');
    legend('Original', 'Estimated', 'Location', 'Best');
    grid on;
    % savefig_tight(h3, 'figures/hw3p5b-mu50', 'both');


    % Misadjustment for all scenarios
    Mcoef = 12;
    Rxx = zeros(Mcoef,Mcoef);
    RMC = 10000;

    for k = 1:RMC
        x = zscore(randn(RMC,1) + randn(RMC,1));
        y = zeros(length(x) + Mcoef - 1, 1);
        for i = Mcoef:length(x)
            for ii = 0:11
                y(i + Mcoef - 1) = y(i + Mcoef - 1) + x(i - ii);
            end
        end
        [~,R] = corrmtx(y, Mcoef - 1, 'autocorrelation');

        Rxx = Rxx + R;
    end

    Rxx = Rxx./RMC;

    rTrace = trace(Rxx);
    rTraceceil = trace(ceil(Rxx));

    mis.the02 = ((0.05/2)*(rTraceceil))/(1 - (0.05/2)*(rTraceceil));
    mis.emp02 = ((0.05/2)*(rTrace))/(1 - (0.05/2)*(rTrace));
```

```
    mis.the10 = ((0.05/10)*(rTraceceil))/(1 - (0.05/10)*(rTraceceil));
    mis.emp10 = ((0.05/10)*(rTrace))/(1 - (0.05/10)*(rTrace));
    mis.the50 = ((0.05/50)*(rTraceceil))/(1 - (0.05/50)*(rTraceceil));
    mis.emp50 = ((0.05/50)*(rTrace))/(1 - (0.05/50)*(rTrace));

    Export = [mis.the02 mis.emp02; mis.the10 mis.emp10; mis.the50
 mis.emp50];
    filter_hw.mat2txt('hw3p5mis.txt', Export, 'w', 'Misadjustment');

    fprintf('Misadjustment ------------ \n\t Emp | The \n');
    fprintf('\t %2.4f | %2.4f \n', Export');
end
```

HOMEWORK 3 - PROBLEM 6

```
function SER = hw3p6(varargin)
    % (a) --------------------------------------
    c_ = struct('original', [57 106 177]./255, 'estimated', [204 37
 41]./255, 'nlms', [107 76 154]./255, 'mean', 'k');

    disp('a')
    % Training Phase
 -----------------------------------------------------------------------------
    % General setup
    mi = 0.4e-0;
    gamma = 1e-3;
    order = 15; M = order + 1;
    N = 500; % Samples

    % Empty vectors to fill with obtained coefficients.
    error = zeros(N,1);
    weights = zeros(M, N);

    % Signal Model
    SNR = inf;
    QAM_train = 4;
    signal_d_train = randi([0,QAM_train - 1],[N 1]);
    signal_d_train = qammod(signal_d_train,QAM_train);
    Hz = [0.5 1.2 1.5 -1];
    signal_x_train = filtfilt(Hz,1,signal_d_train);
    snr = 10^(SNR/10);
    energy = mean(abs(signal_x_train(:)).^2);
    noise = sqrt(energy.*1/snr/2) * (complex(randn(N,1), randn(N,1)));

    % Generating the noisy received signal.
    signal_x_train = signal_x_train + noise;
    % NLMS algorithm
    for s = M:N
        window_x = signal_x_train(s:-1:s-M+1);
        mi_normalized = mi/(gamma + norm(window_x)^2);
        error(s) = signal_d_train(s-M+1) - weights(:,s)'*window_x;
        weights(:,s+1) = weights(:,s) + mi_normalized * conj(error(s))
 * window_x;
```

```
    end

    % Transmission
--------------------------------------------------------------------------
    N = 5000 + M;

    % Signal Model
    SNR = 30;
    QAM = 16;
    signal_d = randi([0,QAM - 1],[N 1]); % The same pilot for every
pilot frame and block.
    signal_d = qammod(signal_d,QAM); % 4-QAM Pilot Signal.
    signal_x = filtfilt(Hz,1,signal_d);
    snr = 10^(SNR/10);
    energy = mean(abs(signal_x(:)).^2); % Energy symbol pilot.
    noise = sqrt(energy.*1/snr/2) * (complex(randn(N,1), randn(N,1)));
    signal_x = signal_x + noise;

    % Empty vectors to fill with obtained coefficients.
    weightsShape = weights(:,s+1);
    error = zeros(N,1);
    weights = zeros(M, N);
    weights(:,M) = weightsShape;
    signal_d_hat = zeros(size(signal_d));

    % NLMS algorithm with QAM signal
    for s = M:N
        window_x = signal_x(s:-1:s-M+1);
        mi_normalized = mi/(gamma + norm(window_x)^2);
        signal_d_hat(s-M+1) = weights(:,s)'*window_x; % Filtering the
signal
        error(s) = qammod(qamdemod(signal_x(s-M+1),QAM),QAM) -
weights(:,s)'*window_x;
        weights(:,s+1) = weights(:,s) + mi_normalized * conj(error(s))
* window_x;
    end

    % MSE Curve
    h1 = figure();
    semilogy(1:N, abs(error).^2,'-','color', c_.nlms , "linewidth",
1);
    hold on
    semilogy(1:N, repelem(mean(abs(error).^2), N),'--','color',
c_.mean , "linewidth", 1);
    hold off
    xlabel('Samples, N');
    ylabel('MSE');
    xlim([0 N]);
    legend('NLMS', 'Mean', 'Location', 'Best');
    grid on;
    % savefig_tight(h1, 'figures/hw3p6a-MSE', 'both');

    % Temporal Evolution
    ShowEvolution = qamdemod(signal_d_hat,QAM);
```

```
    Lsamples = 50;
    h2 = figure();
    subplot(2,2,1)
    stem(1:Lsamples, qamdemod(signal_d(1:Lsamples),QAM),'-','color',
 c_.original, "linewidth", 1, "markersize", 2);
    hold on;
    stem(1:Lsamples, ShowEvolution(1:Lsamples), '--', 'color',
 c_.estimated, "linewidth", 1, "markersize", 2);
    hold off;
    xlabel('Sample, N');
    ylabel('Magnitude');
    axis([0 50 0 20])
    grid on;
    subplot(2,2,2)
    stem(300:350, qamdemod(signal_d(300:350),QAM),'-','color',
 c_.original, "linewidth", 1, "markersize", 2);
    hold on;
    stem(300:350, ShowEvolution(300:350), '--', 'color',
 c_.estimated, "linewidth", 1, "markersize", 2);
    hold off;
    xlabel('Sample, N');
    ylabel('Magnitude');
    axis([300 350 0 20])

 legend('Original', 'Estimated', 'Location', 'northeastoutside','Orientation', 'Ho
 [0.5 0.47 0.0 1], 'Units','normalized');
    legend boxoff
    grid on;
    subplot(2,2,3)
    stem(3000:3050, qamdemod(signal_d(3000:3050),QAM),'-','color',
 c_.original, "linewidth", 1, "markersize", 2);
    hold on;
    stem(3000:3050, ShowEvolution(3000:3050), '--', 'color',
 c_.estimated, "linewidth", 1, "markersize", 2);
    hold off;
    xlabel('Sample, N');
    ylabel('Magnitude');
    axis([3000 3050 0 20])
    grid on;
    subplot(2,2,4)
    stem((5000-Lsamples):5000, qamdemod(signal_d((5000-
Lsamples):5000),QAM),'-','color', c_.original, "linewidth",
 1, "markersize", 2);
    hold on;
    stem((5000-Lsamples):5000, ShowEvolution((5000-
Lsamples):5000), '--','color', c_.estimated, "linewidth",
 1, "markersize", 2);
    hold off;
    xlabel('Sample, N');
    ylabel('Magnitude');
    axis([4950 5000 0 20])
    grid on;
    % savefig_tight(h2, 'figures/hw3p6a-evolution', 'both');
```

```
% Plot Results
h3 = figure();
subplot(2,2,1)
plot(signal_d_train,'.','color', 'y',"markersize", 8)
title('Training');
xlabel('In Phase');
ylabel('Quadrature');
axis([-2 2 -2 2]);
set(gca,'Color','k');
subplot(2,2,2)
plot(signal_d,'.','color', 'y',"markersize", 8)
title('Original');
xlabel('In Phase');
ylabel('Quadrature');
set(gca,'Color','k');
subplot(2,2,3)
plot(signal_x,'.','color', 'y',"markersize", 8)
title('Transmitted');
xlabel('In Phase');
ylabel('Quadrature');
set(gca,'Color','k');
subplot(2,2,4)

plot(qammod(qamdemod(signal_d_hat,QAM),QAM),'.','color', 'y',"markersize",
8)
title('Filter and Decisor');
xlabel('In Phase');
ylabel('Quadrature');
set(gca,'Color','k');
set(gcf, 'InvertHardcopy', 'off')

% savefig_tight(h3, 'figures/hw3p6a-QAM', 'both');

% General setup


% (b) --------------------------------------
disp('b')

% General setup
mi = 1e-3;
order = 15; M = order + 1;
N = 5000 + 50;

% Signal Model
SNR = 30;
QAM = 16;
signal_d = qammod(randi([0,QAM - 1],[N 1]),QAM);
Hz = [0.5 1.2 1.5 -1];
signal_x = filter(Hz,1,signal_d);
snr = 10^(SNR/10);
energy = mean(abs(signal_x(:)).^2); % Energy symbol pilot.
noise = sqrt(energy.*(1/snr)/2)*complex(randn(N,1), randn(N,1));
signal_x = signal_x + noise;
```

```
    % Training (50 Samples)
    N = 50;
    error = zeros(N,1);
    weights = zeros(M, N);

    % Signal Model
    QAM_train = 4;
    signal_d_train = (1/sqrt(2)) * qammod(randi([0,QAM_train - 1],[N
1]),QAM_train);
    Hz = [0.5 1.2 1.5 -1];
    signal_x_train = filter(Hz,1,signal_d_train);
    snr = 10^(inf/10);
    energy = mean(abs(signal_x_train(:)).^2);
    noise = sqrt(energy.*1/snr/2) * (complex(randn(N,1), randn(N,1)));
    signal_x_train = signal_x_train + noise;

    % LMS algorithm
    for s = M:N
        window_x = signal_x_train(s:-1:s-M+1);
        error(s) = signal_d_train(s-M+1) - weights(:,s)'*window_x;
        weights(:,s+1) = weights(:,s) + 2 * mi * conj(error(s)) *
window_x;
    end

    % Transmission
    N = 5000 + 50; % Samples

    % Empty vectors
    weights = zeros(M, N);
    error = zeros(N,1);
    weightsShape = weights(:,s+1);
    weights(:,M) = weightsShape;
    signal_d_hat_50 = zeros(size(signal_d));

    for s = M:N
        windowX= signal_x(s:-1:s-M+1);
        signal_d_hat_50(s-M+1) = weights(:,s)'*windowX;
        error(s) = qammod(qamdemod(signal_x(s-M+1),QAM),QAM) -
weights(:,s)'*windowX;
        weights(:,s+1) = weights(:,s) + 2 * mi * conj(error(s)) *
windowX;
    end

    % Training (150 Samples)
    N = 150;
    % Empty vectors
    error = zeros(N,1);
    weights = zeros(M, N);

    % Signal Model
    QAM_train = 4;
    signal_d_train = randi([0,QAM_train - 1],[N 1]);
    signal_d_train = (1/sqrt(2)) * qammod(signal_d_train,QAM_train);
```

```
    Hz = [0.5 1.2 1.5 -1];
    signal_x_train = filter(Hz,1,signal_d_train);
    snr = 10^(inf/10);
    energy = mean(abs(signal_x_train(:)).^2);
    noise = sqrt(energy.*1/snr/2) * (complex(randn(N,1), randn(N,1)));
    signal_x_train = signal_x_train + noise;

    % LMS
    for s = M:N
        aux = signal_x_train(s:-1:s-M+1);
        error(s) = signal_d_train(s-M+1) - weights(:,s)'*aux;
        weights(:,s+1) = weights(:,s) + 2 * mi * conj(error(s)) *
windowX;
    end

    % Transmission

    N = 5000 + 50; % Samples
    % Empty vectors
    error = zeros(N,1);
    weightsShape = weights(:,s+1);
    weights = zeros(M, N);
    weights(:,M) = weightsShape;
    signal_d_hat_150 = zeros(size(signal_d));

    % LMS algorithm
    for s = M:N
        windowX= signal_x(s:-1:s-M+1);
        signal_d_hat_150(s-M+1) = weights(:,s)'*windowX;
        error(s) = qammod(qamdemod(signal_x(s-M+1),QAM),QAM) -
weights(:,s)'*windowX;
        weights(:,s+1) = weights(:,s) + 2 * mi * conj(error(s)) *
windowX;
    end

    % Training (300 Samples)
    N = 300;
    % Empty vectors
    error = zeros(N,1);
    weights = zeros(M, N);
    % Signal Model
    QAM_train = 4;
    signal_d_train = randi([0,QAM_train - 1],[N 1]);
    signal_d_train = (1/sqrt(2)) * qammod(signal_d_train,QAM_train);
    Hz = [0.5 1.2 1.5 -1];
    signal_x_train = filter(Hz,1,signal_d_train);
    snr = 10^(inf/10);
    energy = mean(abs(signal_x_train(:)).^2);
    noise = sqrt(energy.*1/snr/2) * (complex(randn(N,1), randn(N,1)));
    signal_x_train = signal_x_train + noise;

    % LMS algorithm
    for s = M:N
        aux = signal_x_train(s:-1:s-M+1);
```

```
        error(s) = signal_d_train(s-M+1) - weights(:,s)'*aux;
        weights(:,s+1) = weights(:,s) + 2 * mi * conj(error(s)) *
windowX;
    end

    % Transmission

    % Empty vectors
    N = 5000 + 50;
    error = zeros(N,1);
    weightsShape = weights(:,s+1);
    weights = zeros(M, N);
    weights(:,M) = weightsShape;
    signal_d_hat_300 = zeros(size(signal_d));

    % LMS algorithm
    for s = M:N
        windowX= signal_x(s:-1:s-M+1);
        signal_d_hat_300(s-M+1) = weights(:,s)'*windowX;
        error(s) = qammod(qamdemod(signal_x(s-M+1),QAM),QAM) -
weights(:,s)'*windowX;
        weights(:,s+1) = weights(:,s) + 2 * mi * conj(error(s)) *
windowX;
    end

    % Training (500 Samples)
    N = 500;

    % Empty vectors
    error = zeros(N,1);
    weights = zeros(M, N);

    % Signal Model
    QAM_train = 4;
    signal_d_train = randi([0,QAM_train - 1],[N 1]);
    signal_d_train = qammod(signal_d_train,QAM_train);
    Hz = [0.5 1.2 1.5 -1];
    signal_x_train = filter(Hz,1,signal_d_train);
    snr = 10^(inf/10);
    energy = mean(abs(signal_x_train(:)).^2);
    noise = sqrt(energy.*1/snr/2) * (complex(randn(N,1), randn(N,1)));
    signal_x_train = signal_x_train + noise;

    % LMS
    for s = M:N
        aux = signal_x_train(s:-1:s-M+1);
        error(s) = signal_d_train(s-M+1) - weights(:,s)'*aux;
        weights(:,s+1) = weights(:,s) + 2 * mi * conj(error(s)) *
windowX;
    end

    % Transmission
    N = 5000 + 50;
```

```
    % Empty vectors
    error = zeros(N,1);
    weightsShape = weights(:,s+1);
    weights = zeros(M, N);
    weights(:,M) = weightsShape;
    signal_d_hat_500 = zeros(size(signal_d));

    % LMS algorithm
    for s = M:N
        windowX= signal_x(s:-1:s-M+1);
        signal_d_hat_500(s-M+1) = weights(:,s)'*windowX;
        error(s) = qammod(qamdemod(signal_x(s-M+1),QAM),QAM) -
weights(:,s)'*windowX;
        weights(:,s+1) = weights(:,s) + 2 * mi * conj(error(s)) *
windowX;
    end

    % Temporal Evolution
    selectWindow = 4975:5000;
    [~,~,temporalShift] =
alignsignals(qamdemod(signal_d,QAM),qamdemod(signal_d_hat_500,QAM));


    evolutionWindow =
circshift(qamdemod(signal_d_hat_50,QAM),temporalShift);
    evolutionWindow_50 = evolutionWindow(selectWindow);
    evolutionWindow =
circshift(qamdemod(signal_d_hat_150,QAM),temporalShift);
    evolutionWindow_150 = evolutionWindow(selectWindow);
    evolutionWindow =
circshift(qamdemod(signal_d_hat_300,QAM),temporalShift);
    evolutionWindow_300 = evolutionWindow(selectWindow);
    evolutionWindow =
circshift(qamdemod(signal_d_hat_500,QAM),temporalShift);
    evolutionWindow_500 = evolutionWindow(selectWindow);

    h4 = figure;
    subplot(2,2,1)
    stem(selectWindow,
qamdemod(signal_d(selectWindow),QAM),'-','color',
c_.original, "linewidth", 1, "markersize", 1);
    hold on;
    stem(selectWindow, evolutionWindow_50,'--','color',
c_.estimated, "linewidth", 1, "markersize", 1);
    hold off;
    title('50 Samples');
    xlabel('Sample, N');
    xlim([min(selectWindow) max(selectWindow)]);
    ylabel('Magnitude');
    ylim([0 20])

legend('Original', 'Estimated', 'Location', 'northeastoutside','Orientation', 'Ho
[0.5 0.47 0.0 1], 'Units','normalized');
    grid on;
```

```
    legend boxoff
    subplot(2,2,2)
    stem(selectWindow,
qamdemod(signal_d(selectWindow),QAM),'-','color',
c_.original, "linewidth", 1, "markersize", 1);
    hold on;
    stem(selectWindow, evolutionWindow_150,'--','color',
c_.estimated, "linewidth", 1, "markersize", 1);
    hold off;
    title('150 Samples');
    xlabel('Sample, N');
    xlim([min(selectWindow) max(selectWindow)]);
    ylabel('Magnitude');
    grid on;
    subplot(2,2,3)
    stem(selectWindow,
qamdemod(signal_d(selectWindow),QAM),'-','color',
c_.original, "linewidth", 1, "markersize", 1);
    hold on;
    stem(selectWindow, evolutionWindow_300,'--','color',
c_.estimated, "linewidth", 1, "markersize", 1);
    hold off;
    title('300 Samples');
    xlabel('Sample, N');
    xlim([min(selectWindow) max(selectWindow)]);
    ylabel('Magnitude');
    grid on;
    subplot(2,2,4)
    stem(selectWindow,
qamdemod(signal_d(selectWindow),QAM),'-','color',
c_.original, "linewidth", 1, "markersize", 1);
    hold on;
    stem(selectWindow, evolutionWindow_500,'--','color',
c_.estimated, "linewidth", 1, "markersize", 1);
    hold off;
    title('500 Samples');
    xlabel('Sample, N');
    xlim([min(selectWindow) max(selectWindow)]);
    ylabel('Magnitude');
    grid on;

    savefig_tight(h4, 'figures/hw3p6b-evolutionSamples', 'both');


    % (c) --------------------------------------
    disp('c');

    % General Setup
    N = 500;
    mi = 0.4;
    gamma = 1e-3;
    order = 15; M = order+1;

    % Empty vectors
```

```
    error = zeros(N,1);
    weights = zeros(M, N);

    % Signal Model
    SNR = 30;
    QAM_train = 4;
    signal_d_train = randi([0,QAM_train - 1],[N 1]);
    signal_d_train = qammod(signal_d_train,QAM_train);
    Hz = [0.5 1.2 1.5 -1];
    signal_x_train = filtfilt(Hz,1,signal_d_train);
    snr = 10^(inf/10);
    energy = mean(abs(signal_x_train(:)).^2);
    noise = sqrt(energy.*1/snr/2) * complex(randn(N,1), randn(N,1));;
    signal_x_train = signal_x_train + noise;

    % LMS
    for s = M:N
        aux = signal_x_train(s:-1:s-M+1);
        mi_normalized = mi/(gamma + norm(aux)^2);
        error(s) = signal_d_train(s-M+1) - weights(:,s)'*aux;
        weights(:,s+1) = weights(:,s) + mi_normalized * conj(error(s))
* aux;
    end

    % Transmission
    N = 5000 + 50; % Number of samples

    % Empty vectors
    error = zeros(N,1);
    weights = zeros(M, N);

    % Signal Model
    SNR = 30;
    QAM = 256;
    signal_d = randi([0,QAM - 1],[N 1]);
    signal_d = qammod(signal_d,QAM); % 4-QAM Pilot Signal.
    Hz = [0.5 1.2 1.5 -1];
    signal_x = filtfilt(Hz,1,signal_d);
    snr = 10^(SNR/10);
    energy = mean(abs(signal_x(:)).^2);
    noise = sqrt(energy.*1/snr/2)*complex(randn(N,1), randn(N,1));;
    signal_x = signal_x + noise;
    signal_d_hat = zeros(size(signal_d));

    % NLMS
    for s = M:N
        aux = signal_x(s:-1:s-M+1);
        mi_normalized = mi/(gamma + norm(aux)^2);
        signal_d_hat(s-M+1) = weights(:,s)'*aux;
        error(s) = qammod(qamdemod(signal_x(s-M+1),QAM),QAM) -
weights(:,s)'*aux;
        weights(:,s+1) = weights(:,s) + mi_normalized * conj(error(s))
* aux;
    end
```

```
    % MSE
    h5 = figure();
    semilogy(1:N, abs(error).^2,'-','color', c_.nlms , "linewidth",
1);
    hold on
    semilogy(1:N, repelem(mean(abs(error).^2), N),'--','color',
c_.mean , "linewidth", 1);
    hold off
    xlabel('Samples, N');
    xlim([0 N]);
    ylabel('MSE');
    legend('NLMS', 'Mean', 'Location', 'Best');
    grid on;
    savefig_tight(h5, 'figures/hw3p6c-MSE', 'both');

    % Temporal Evolution
    L = 50;
    aux = qamdemod(signal_d_hat,QAM);
    aux1 = aux(1:L);
    aux2 = aux(5000-L:5000);

    figure
    subplot(211)
    stem(1:L, qamdemod(signal_d(1:L),QAM),'-','color',
c_.original, "linewidth", 1, "markersize", 3);
    hold on;
    stem(1:L, aux1,'-','color', c_.estimated, "linewidth",
1, "markersize", 3);
    hold off;
    title('First Samples');
    xlabel('Sample, N');
    xlim([0 L])
    ylabel('Magnitude');

legend('Original', 'Estimated', 'Location', 'northeastoutside','Orientation', 'Ho
[0.5 0.47 0.0 1.03], 'Units','normalized');
    legend boxoff
    grid on;
    subplot(212)
    stem((5000-L):5000, qamdemod(signal_d((5000-
L):5000),QAM),'-','color', c_.original, "linewidth", 1, "markersize",
3);
    hold on;
    stem((5000-L):5000, aux2,'-','color', c_.estimated, "linewidth",
1, "markersize", 3);
    hold off;
    title('Last Samples');
    xlabel('Sample, N');
    ylabel('Magnitude');
    xlim([(5000-L) 5000])
    grid on;
    % savefig_tight(h5, 'figures/hw3p6c-evolution', 'both');
```

```
    % (d) --------------------------------------
    disp('d')
    close all;

    % General Setup
    RMC = 1000;
    QAM_train = 4;
    QAM_symbols = 4.^(1:4);
    SNRdB = 0:10:30;
    order = 15; M = order + 1;
    mi = 0.4;
    gamma = 1e3;
    train.N = 500;
    trans.N = 5000;
    Hz = [0.5 1.2 1.5 -1];

    train.error = zeros(train.N,1);
    train.weights = zeros(M, train.N);

    trans.error = zeros(trans.N,1);
    trans.weights = zeros(M, trans.N);

    SER = cell(RMC, length(QAM_symbols), length(SNRdB));

    tic;
    for rmc = 1:RMC
        for iiQAM = 1:length(QAM_symbols)
            for iiSNR = 1:length(SNRdB)
                fprintf('RMC, SNR (%2.0f, %2.0f dB) -- %2.0f-QAM \n',
 rmc, SNRdB(iiSNR), QAM_symbols(iiQAM))

                % Training
                signal_d_train = qammod(randi([0,QAM_train - 1],
[train.N 1]),QAM_train);
                signal_x_train = filtfilt(Hz,1,signal_d_train);
                energy = mean(abs(signal_x_train(:)).^2); % Energy
 symbol
                signal_x_train = signal_x_train + sqrt(energy.*1/
(10^(inf/10))/2) * complex(randn(train.N,1), randn(train.N,1));

                for s = M:train.N
                    aux = signal_x_train(s:-1:s-M+1);
                    mi_normalized = mi/(gamma + norm(aux)^2);
                    train.error(s) = signal_d_train(s-M+1) -
 train.weights(:,s)'*aux;
                    train.weights(:,s+1) = train.weights(:,s) +
 mi_normalized * conj(train.error(s)) * aux;
                end

                % Transmission
                QAM = QAM_symbols(iiQAM);
                signal_d = qammod(randi([0,QAM - 1],[trans.N 1]),QAM);
                signal_x = filtfilt(Hz,1,signal_d);
```

```
                energy = mean(abs(signal_x(:)).^2); % Energy symbol
 pilot.
                signal_x = signal_x + sqrt(energy.*1/
(10^(SNRdB(iiSNR)/10))/2) * (randn(trans.N,1) + 1i*randn(trans.N,1));
                signal_d_hat = zeros(size(signal_d));

                % NLMS
                for s = M:trans.N
                    aux = signal_x(s:-1:s-M+1);
                    mi_normalized = mi/(gamma + norm(aux)^2);
                    signal_d_hat(s-M+1) = trans.weights(:,s)'*aux;
                    trans.error(s) = qammod(qamdemod(signal_x(s-M
+1),QAM),QAM) - trans.weights(:,s)'*aux;
                    trans.weights(:,s+1) = trans.weights(:,s) +
 mi_normalized * conj(trans.error(s)) * aux;
                end

                SER{rmc, iiSNR, iiQAM} = sum(qamdemod(signal_d,QAM) ~=
 qamdemod(signal_d_hat,QAM)) / length(qamdemod(signal_d,QAM));

            end
        end
        fprintf('-----------------------------------------------\n\n')
    end

    t = toc;
    disp(t)

    c_ =
 struct('QAM4', 'y', 'QAM16', 'k', 'QAM64', 'r', 'QAM256', 'b', 'mean', 'k');

    h6 = figure();
    semilogy(SNRdB, mean(cell2mat(SER(:, :, 1)), 1),'-', 'color',
 c_.QAM4, 'linewidth', 1.5);
    hold on;
    semilogy(SNRdB, mean(cell2mat(SER(:, :, 2)), 1),'-', 'color',
 c_.QAM16, 'Marker', 's', 'MarkerFaceColor', c_.QAM16, 'linewidth',
 1.5);
    semilogy(SNRdB, mean(cell2mat(SER(:, :, 3)), 1),'-.', 'color',
 c_.QAM64, 'Marker', 'o', 'MarkerFaceColor', c_.QAM64, 'linewidth',
 1.5);
    semilogy(SNRdB, mean(cell2mat(SER(:, :, 4)), 1),'--', 'color',
 c_.QAM256, 'Marker', '^', 'MarkerFaceColor', c_.QAM256, 'linewidth',
 1.5);
    hold off;
    xlabel('SNR (dB)');
    ylabel('SER');
    xticks(SNRdB);
    ylim([2e-3 2]);
    legend('4-QAM', '16-QAM', '64-QAM', '256-
QAM', 'Location', 'Best');
    grid minor

    save('hw3p6d.mat', 'SNRdB', 'SER', 'c_');
```

```
    savefig_tight(h6, 'figures/hw3p6d-SER', 'both');

    disp('pause');
    pause();
    return

end
```

HOMEWORK 4 - PROBLEM 1

```
function [y, weights]  = hw4p1rls(signal_x, signal_d, M, lambda,
 delta, fixcoeff)
    N = length(signal_d);
    error = zeros(N,1);
    weights = zeros(M, N);
    Rd = delta*eye(M);
    y = zeros(N,1);
    weights(1,1) = 1;

    for n = 2:(N - M - 1)
        Rd = (1/lambda)*(Rd - (Rd*signal_x(n:n+M-1)*signal_x(n:n
+M-1)'*Rd)/(lambda + signal_x(n:n+M-1)'*Rd*signal_x(n:n+M-1)));
        error(n) = signal_d(n) - weights(:,n-1)' * signal_x(n:n+M-1);
        weights(:,n) = weights(:,n-1) + Rd*error(n)*signal_x(n:n+M-1);
        if fixcoeff
            weights(1,n) = 1; % Impose first coeff fix
        end
        y(n) = weights(:,n-1)' * signal_x(n:n+M-1);
    end

end

function hw4p1(varargin)
    disp('hw4p1')
    c_ = struct('original', [57 106 177]./255, 'fixcoef', [204 37
 41]./255, 'freecoef', [62 150 81]./255);

    % General Setup
    N = 100;
    order = 2; M = order + 1;
    lambda = 0.98;
    delta = 1;

    % Signal Model
    t = linspace(-3*pi,3*pi,N).';
    signal_d = cos(pi*t/3);
    SNR_dB = 10;
    noise = sqrt((1/(10^(SNR_dB/10)))/2).*randn(N,1);
    signal_x = signal_d + noise;

    [fixcoef.y, fixcoef.weights] = filter_hw.hw4p1rls(signal_x,
 signal_d, M, lambda, delta, true);
```

```
    [freecoef.y, freecoef.weights] = filter_hw.hw4p1rls(signal_x,
 signal_d, M, lambda, delta, false);

    filter_hw.mat2txt('hw4p1coef.txt',
 fixcoef.weights(:,1:10).', 'w', 'coef fix');
    filter_hw.mat2txt('hw4p1coef.txt',
 freecoef.weights(:,1:10).', 'a', 'free fix');

    % MSE Curve
    h1 = figure();
    plot(signal_d,'-','color', 'k', "linewidth", 1);
    hold on;
    plot(fixcoef.y,'--','color', c_.fixcoef,
  'Marker', '^', 'MarkerFaceColor',
 c_.fixcoef, 'MarkerIndices',1:20:length(fixcoef.y), "linewidth", 1);
    plot(freecoef.y,'-.','color', c_.freecoef,
  'Marker', 'o', 'MarkerFaceColor',
 c_.freecoef, 'MarkerIndices',1:25:length(freecoef.y), "linewidth",
 1);
    hold off;
    xlabel('Samples, N');
    ylabel('Magnitude');
    legend('Original (SNR = 10 dB)', 'Fix Coef', 'Free
 Coef', 'Location', 'Best');
    grid on;
    savefig_tight(h1, 'figures/hw4p1', 'both');

end
```

HOMEWORK 4 - PROBLEM 3

```
function [error, weights] = hw4p3rls(signal_d, M, SNR_dB, lambda)
    N = length(signal_d);
    error = zeros(N,1);
    weights = zeros(M, N);

    noise = sqrt((1/(10^(SNR_dB/10)))/2).*randn(N,1);
    signal_x = signal_d + noise; % Defining delta by the inverse of
 the signal energy

    delta  = 1/(sum(signal_x.^2)/length(signal_x));
    Rd = delta*eye(M);

    signal_d = signal_d(M:end,1);
    for ss = 2:(N - M - 1)
        Rd = (1/lambda)*(Rd - (Rd*signal_x(ss:ss+M-1)*signal_x(ss:ss
+M-1)'*Rd)/(lambda + signal_x(ss:ss+M-1)'*Rd*signal_x(ss:ss+M-1)));
        error(ss) = signal_d(ss) - weights(:,ss-1)' * signal_x(ss:ss
+M-1);
        weights(:,ss) = weights(:,ss-1) + Rd*error(ss)*signal_x(ss:ss
+M-1);
    end
    weights = flip(weights);
```

```
end

function hw4p3(varargin)
    disp('hw4p3');

    c_ = struct('original', [57 106 177]./255, 'estimated', [204 37
 41]./255, 'nlms', [107 76 154]./255, 'mean', 'k');

    % General Setup
    N = 510;
    A.lambda = 0.9;
    B.lambda = 0.99;
    C.lambda = 0.999;

    % Order = 2
 --------------------------------------------------------------------------------
    order = 2; M = order + 1;
    SNR_dB = 3;

    % Signal Model
    t = linspace(-pi,pi,N).';
    signal_d = sin(2*pi*t); % Generating the noisy received signal.

    % Change: M, SNR, lambda
    [A.error, A.weights] = filter_hw.hw4p3rls(signal_d, M, SNR_dB,
 A.lambda);
    [B.error, B.weights] = filter_hw.hw4p3rls(signal_d, M, SNR_dB,
 B.lambda);
    [C.error, C.weights] = filter_hw.hw4p3rls(signal_d, M, SNR_dB,
 C.lambda);

    % Plot - 3 dB
    h1 = figure();
    subplot(2,2,1)
    semilogy(1:N, A.error.^2,'-','color', c_.nlms, "linewidth",
 1, "markersize", 8);
    hold on
    semilogy(1:N, repelem(mean(A.error.^2), N), '--', 'color',
 c_.mean, "linewidth", 1);
    hold off
    xlabel('Samples, N');
    xlim([0 N-10]);
    ylim([1e-8 1e0])
    ylabel('MSE');
    legend(strcat(sprintf('RLS (M = %2.0f, SNR = $%2.0f$ dB,', order,
 SNR_dB), ' $\lambda$ = ', num2str(A.lambda), ')'), strcat('Mean =',
 num2str(mean(A.error.^2))), 'interpreter', 'latex', 'Orientation', 'Vertical', 'L
    legend boxoff
    grid on;

    subplot(2,2,2)
    semilogy(1:N, B.error.^2,'-','color', c_.nlms, "linewidth",
 1, "markersize", 8);
```

```
    hold on
    semilogy(1:N, repelem(mean(B.error.^2), N), '--', 'color',
c_.mean, "linewidth", 1);
    hold off
    xlabel('Samples, N');
    xlim([0 N-10]);
    ylim([1e-8 1e0])
    ylabel('MSE');
    legend(strcat(sprintf('RLS (M = %2.0f, SNR = $%2.0f$ dB,', order,
SNR_dB), ' $\lambda$ = ', num2str(B.lambda), ')'), strcat('Mean =',
num2str(mean(B.error.^2))), 'interpreter', 'latex', 'Orientation', 'Vertical', 'L
    legend boxoff
    grid on;

    subplot(2,2,3)
    plot(1:N, A.weights(1,:),'-','color', c_.original, "linewidth",
1);
    hold on;
    plot(1:N, A.weights(2,:),'--','color', c_.estimated, "linewidth",
1);
    hold off;
    xlabel('Samples, N');
    ylabel('Magnitude');
    xlim([0 N-10]);

legend('$w_0$', '$w_1$', 'interpreter', 'latex', 'Orientation', 'Horizontal', 'Lo
    legend boxoff
    grid on;

    subplot(2,2,4)
    plot(1:N, B.weights(1,:),'-','color', c_.original, "linewidth",
1);
    hold on;
    plot(1:N, B.weights(2,:),'--','color', c_.estimated, "linewidth",
1);
    hold off;
    xlabel('Samples, N');
    ylabel('Magnitude');
    xlim([0 N-10]);

legend('$w_0$', '$w_1$', 'interpreter', 'latex', 'Orientation', 'Horizontal', 'Lo
    legend boxoff
    grid on;

    % savefig_tight(h1, 'figures/hw4p3-fig1', 'both');

    h2 = figure();
    subplot(2,1,1)
    semilogy(1:N, C.error.^2,'-','color', c_.nlms, "linewidth",
1, "markersize", 8);
    hold on
    semilogy(1:N, repelem(mean(C.error.^2), N), '--', 'color',
c_.mean, "linewidth", 1);
    hold off
```

```
    xlabel('Samples, N');
    xlim([0 N-10]);
    ylim([1e-8 1e0])
    ylabel('MSE');
    legend(strcat(sprintf('RLS (M = %2.0f, SNR = $%2.0f$ dB,', order,
SNR_dB), ' $\lambda$ = ', num2str(C.lambda), ')'), strcat('Mean =',
num2str(mean(B.error.^2))), 'interpreter', 'latex', 'Orientation', 'Vertical', 'L
    legend boxoff
    grid on;

    subplot(2,1,2)
    plot(1:N, C.weights(1,:),'-','color', c_.original, "linewidth",
1);
    hold on;
    plot(1:N, C.weights(2,:),'--','color', c_.estimated, "linewidth",
1);
    hold off;
    xlabel('Samples, N');
    ylabel('Magnitude');
    xlim([0 N-10]);

legend('$w_0$', '$w_1$', 'interpreter', 'latex', 'Orientation', 'Horizontal', 'Lo
    legend boxoff
    grid on;

    % savefig_tight(h2, 'figures/hw4p3-fig2', 'both');

    pause;
    close all;

    SNR_dB = inf;

    % Signal Model
    t = linspace(-pi,pi,N).';
    signal_d = sin(2*pi*t); % Generating the noisy received signal.

    % Change: M, SNR, lambda
    [A.error, A.weights] = filter_hw.hw4p3rls(signal_d, M, SNR_dB,
A.lambda);
    [B.error, B.weights] = filter_hw.hw4p3rls(signal_d, M, SNR_dB,
B.lambda);
    [C.error, C.weights] = filter_hw.hw4p3rls(signal_d, M, SNR_dB,
C.lambda);

    % Plot - inf dB
    h3 = figure();
    subplot(2,2,1)
    semilogy(1:N, A.error.^2,'-','color', c_.nlms, "linewidth",
1, "markersize", 8);
    hold on
    semilogy(1:N, repelem(mean(A.error.^2), N), '--', 'color',
c_.mean, "linewidth", 1);
    hold off
    xlabel('Samples, N');
```

```
    xlim([0 N-10]);
    ylim([1e-8 1e0])
    ylabel('MSE');
    legend(strcat(sprintf('RLS (M = %2.0f, SNR = $%2.0f$ dB,', order,
SNR_dB), ' $\lambda$ = ', num2str(A.lambda), ')'), strcat('Mean =',
num2str(mean(A.error.^2))), 'interpreter', 'latex', 'Orientation', 'Vertical', 'L
    legend boxoff
    grid on;

    subplot(2,2,2)
    semilogy(1:N, B.error.^2,'-','color', c_.nlms, "linewidth",
1, "markersize", 8);
    hold on
    semilogy(1:N, repelem(mean(B.error.^2), N), '--', 'color',
c_.mean, "linewidth", 1);
    hold off
    xlabel('Samples, N');
    xlim([0 N-10]);
    ylim([1e-8 1e0])
    ylabel('MSE');
    legend(strcat(sprintf('RLS (M = %2.0f, SNR = $%2.0f$ dB,', order,
SNR_dB), ' $\lambda$ = ', num2str(B.lambda), ')'), strcat('Mean =',
num2str(mean(B.error.^2))), 'interpreter', 'latex', 'Orientation', 'Vertical', 'L
    legend boxoff
    grid on;

    subplot(2,2,3)
    plot(1:N, A.weights(1,:),'-','color', c_.original, "linewidth",
1);
    hold on;
    plot(1:N, A.weights(2,:),'--','color', c_.estimated, "linewidth",
1);
    hold off;
    xlabel('Samples, N');
    ylabel('Magnitude');
    xlim([0 N-10]);

legend('$w_0$', '$w_1$', 'interpreter', 'latex', 'Orientation', 'Horizontal', 'Lo
    legend boxoff
    grid on;

    subplot(2,2,4)
    plot(1:N, B.weights(1,:),'-','color', c_.original, "linewidth",
1);
    hold on;
    plot(1:N, B.weights(2,:),'--','color', c_.estimated, "linewidth",
1);
    hold off;
    xlabel('Samples, N');
    ylabel('Magnitude');
    xlim([0 N-10]);

legend('$w_0$', '$w_1$', 'interpreter', 'latex', 'Orientation', 'Horizontal', 'Lo
    legend boxoff
```

```
    grid on;

    % savefig_tight(h3, 'figures/hw4p3-fig3', 'both');

    h4 = figure();
    subplot(2,1,1)
    semilogy(1:N, C.error.^2,'-','color', c_.nlms, "linewidth",
1, "markersize", 8);
    hold on
    semilogy(1:N, repelem(mean(C.error.^2), N), '--', 'color',
c_.mean, "linewidth", 1);
    hold off
    xlabel('Samples, N');
    xlim([0 N-10]);
    ylim([1e-8 1e0])
    ylabel('MSE');
    legend(strcat(sprintf('RLS (M = %2.0f, SNR = $%2.0f$ dB,', order,
SNR_dB), ' $\lambda$ = ', num2str(C.lambda), ')'), strcat('Mean =',
num2str(mean(B.error.^2))), 'interpreter', 'latex', 'Orientation', 'Vertical', 'L
    legend boxoff
    grid on;

    subplot(2,1,2)
    plot(1:N, C.weights(1,:),'-','color', c_.original, "linewidth",
1);
    hold on;
    plot(1:N, C.weights(2,:),'--','color', c_.estimated, "linewidth",
1);
    hold off;
    xlabel('Samples, N');
    ylabel('Magnitude');
    xlim([0 N-10]);

legend('$w_0$', '$w_1$', 'interpreter', 'latex', 'Orientation', 'Horizontal', 'Lo
    legend boxoff
    grid on;

    % savefig_tight(h4, 'figures/hw4p3-fig4', 'both');

    pause;
    close all;

    % Order = 3
--------------------------------------------------------------------------------
    order = 3; M = order + 1;
    SNR_dB = 3;

    % Signal Model
    t = linspace(-pi,pi,N).';
    signal_d = sin(2*pi*t); % Generating the noisy received signal.

    % Change: M, SNR, lambda
    [A.error, A.weights] = filter_hw.hw4p3rls(signal_d, M, SNR_dB,
A.lambda);
```

```
    [B.error, B.weights] = filter_hw.hw4p3rls(signal_d, M, SNR_dB,
B.lambda);
    [C.error, C.weights] = filter_hw.hw4p3rls(signal_d, M, SNR_dB,
C.lambda);

    % Plot - 3 dB
    h5 = figure();
    subplot(2,2,1)
    semilogy(1:N, A.error.^2,'-','color', c_.nlms, "linewidth",
1, "markersize", 8);
    hold on
    semilogy(1:N, repelem(mean(A.error.^2), N), '--', 'color',
c_.mean, "linewidth", 1);
    hold off
    xlabel('Samples, N');
    xlim([0 N-10]);
    ylim([1e-8 1e0])
    ylabel('MSE');
    legend(strcat(sprintf('RLS (M = %2.0f, SNR = $%2.0f$ dB,', order,
SNR_dB), ' $\lambda$ = ', num2str(A.lambda), ')'), strcat('Mean =',
num2str(mean(A.error.^2))), 'interpreter', 'latex', 'Orientation', 'Vertical', 'L
    legend boxoff
    grid on;

    subplot(2,2,2)
    semilogy(1:N, B.error.^2,'-','color', c_.nlms, "linewidth",
1, "markersize", 8);
    hold on
    semilogy(1:N, repelem(mean(B.error.^2), N), '--', 'color',
c_.mean, "linewidth", 1);
    hold off
    xlabel('Samples, N');
    xlim([0 N-10]);
    ylim([1e-8 1e0])
    ylabel('MSE');
    legend(strcat(sprintf('RLS (M = %2.0f, SNR = $%2.0f$ dB,', order,
SNR_dB), ' $\lambda$ = ', num2str(B.lambda), ')'), strcat('Mean =',
num2str(mean(B.error.^2))), 'interpreter', 'latex', 'Orientation', 'Vertical', 'L
    legend boxoff
    grid on;

    subplot(2,2,3)
    plot(1:N, A.weights(1,:),'-','color', c_.original, "linewidth",
1);
    hold on;
    plot(1:N, A.weights(2,:),'--','color', c_.estimated, "linewidth",
1);
    hold off;
    xlabel('Samples, N');
    ylabel('Magnitude');
    xlim([0 N-10]);

legend('$w_0$', '$w_1$', 'interpreter', 'latex', 'Orientation', 'Horizontal', 'Lo
    legend boxoff
```

```
    grid on;

    subplot(2,2,4)
    plot(1:N, B.weights(1,:),'-','color', c_.original, "linewidth",
1);
    hold on;
    plot(1:N, B.weights(2,:),'--','color', c_.estimated, "linewidth",
1);
    hold off;
    xlabel('Samples, N');
    ylabel('Magnitude');
    xlim([0 N-10]);

legend('$w_0$', '$w_1$', 'interpreter', 'latex', 'Orientation', 'Horizontal', 'Lo
    legend boxoff
    grid on;

    % savefig_tight(h5, 'figures/hw4p3-fig5', 'both');

    h6 = figure();
    subplot(2,1,1)
    semilogy(1:N, C.error.^2,'-','color', c_.nlms, "linewidth",
1, "markersize", 8);
    hold on
    semilogy(1:N, repelem(mean(C.error.^2), N), '--', 'color',
c_.mean, "linewidth", 1);
    hold off
    xlabel('Samples, N');
    xlim([0 N-10]);
    ylim([1e-8 1e0])
    ylabel('MSE');
    legend(strcat(sprintf('RLS (M = %2.0f, SNR = $%2.0f$ dB,', order,
SNR_dB), ' $\lambda$ = ', num2str(C.lambda), ')'), strcat('Mean =',
num2str(mean(B.error.^2))), 'interpreter', 'latex', 'Orientation', 'Vertical', 'L
    legend boxoff
    grid on;

    subplot(2,1,2)
    plot(1:N, C.weights(1,:),'-','color', c_.original, "linewidth",
1);
    hold on;
    plot(1:N, C.weights(2,:),'--','color', c_.estimated, "linewidth",
1);
    hold off;
    xlabel('Samples, N');
    ylabel('Magnitude');
    xlim([0 N-10]);

legend('$w_0$', '$w_1$', 'interpreter', 'latex', 'Orientation', 'Horizontal', 'Lo
    legend boxoff
    grid on;

    % savefig_tight(h6, 'figures/hw4p3-fig6', 'both');
```

```
   pause;
   close all;

   SNR_dB = inf;

   % Signal Model
   t = linspace(-pi,pi,N).';
   signal_d = sin(2*pi*t); % Generating the noisy received signal.

   % Change: M, SNR, lambda
   [A.error, A.weights] = filter_hw.hw4p3rls(signal_d, M, SNR_dB,
A.lambda);
   [B.error, B.weights] = filter_hw.hw4p3rls(signal_d, M, SNR_dB,
B.lambda);
   [C.error, C.weights] = filter_hw.hw4p3rls(signal_d, M, SNR_dB,
C.lambda);

   % Plot - inf dB
   h7 = figure();
   subplot(2,2,1)
   semilogy(1:N, A.error.^2,'-','color', c_.nlms, "linewidth",
1, "markersize", 8);
   hold on
   semilogy(1:N, repelem(mean(A.error.^2), N), '--', 'color',
c_.mean, "linewidth", 1);
   hold off
   xlabel('Samples, N');
   xlim([0 N-10]);
   ylim([1e-8 1e0])
   ylabel('MSE');
   legend(strcat(sprintf('RLS (M = %2.0f, SNR = $%2.0f$ dB,', order,
SNR_dB), ' $\lambda$ = ', num2str(A.lambda), ')'), strcat('Mean =',
num2str(mean(A.error.^2))), 'interpreter', 'latex', 'Orientation', 'Vertical', 'L
   legend boxoff
   grid on;

   subplot(2,2,2)
   semilogy(1:N, B.error.^2,'-','color', c_.nlms, "linewidth",
1, "markersize", 8);
   hold on
   semilogy(1:N, repelem(mean(B.error.^2), N), '--', 'color',
c_.mean, "linewidth", 1);
   hold off
   xlabel('Samples, N');
   xlim([0 N-10]);
   ylim([1e-8 1e0])
   ylabel('MSE');
   legend(strcat(sprintf('RLS (M = %2.0f, SNR = $%2.0f$ dB,', order,
SNR_dB), ' $\lambda$ = ', num2str(B.lambda), ')'), strcat('Mean =',
num2str(mean(B.error.^2))), 'interpreter', 'latex', 'Orientation', 'Vertical', 'L
   legend boxoff
   grid on;

   subplot(2,2,3)
```

```
    plot(1:N, A.weights(1,:),'-','color', c_.original, "linewidth",
1);
    hold on;
    plot(1:N, A.weights(2,:),'--','color', c_.estimated, "linewidth",
1);
    hold off;
    xlabel('Samples, N');
    ylabel('Magnitude');
    xlim([0 N-10]);

legend('$w_0$', '$w_1$', 'interpreter', 'latex', 'Orientation', 'Horizontal', 'Lo
    legend boxoff
    grid on;

    subplot(2,2,4)
    plot(1:N, B.weights(1,:),'-','color', c_.original, "linewidth",
1);
    hold on;
    plot(1:N, B.weights(2,:),'--','color', c_.estimated, "linewidth",
1);
    hold off;
    xlabel('Samples, N');
    ylabel('Magnitude');
    xlim([0 N-10]);

legend('$w_0$', '$w_1$', 'interpreter', 'latex', 'Orientation', 'Horizontal', 'Lo
    legend boxoff
    grid on;

    savefig_tight(h7, 'figures/hw4p3-fig7', 'both');

    h8 = figure();
    subplot(2,1,1)
    semilogy(1:N, C.error.^2,'-','color', c_.nlms, "linewidth",
1, "markersize", 8);
    hold on
    semilogy(1:N, repelem(mean(C.error.^2), N), '--', 'color',
c_.mean, "linewidth", 1);
    hold off
    xlabel('Samples, N');
    xlim([0 N-10]);
    ylim([1e-8 1e0])
    ylabel('MSE');
    legend(strcat(sprintf('RLS (M = %2.0f, SNR = $%2.0f$ dB,', order,
SNR_dB), ' $\lambda$ = ', num2str(C.lambda), ')'), strcat('Mean =',
num2str(mean(B.error.^2))), 'interpreter', 'latex', 'Orientation', 'Vertical', 'L
    legend boxoff
    grid on;

    subplot(2,1,2)
    plot(1:N, C.weights(1,:),'-','color', c_.original, "linewidth",
1);
    hold on;
```

```
    plot(1:N, C.weights(2,:),'--','color', c_.estimated, "linewidth",
1);
    hold off;
    xlabel('Samples, N');
    ylabel('Magnitude');
    xlim([0 N-10]);

legend('$w_0$', '$w_1$', 'interpreter', 'latex', 'Orientation', 'Horizontal', 'Lo
    legend boxoff
    grid on;

    savefig_tight(h8, 'figures/hw4p3-fig8', 'both');



end
```

VERBOSE DETAILS

```
function export_fig(Activate, h, filename)
    if Activate
        savefig_tight(h, filename, 'both');
        filter_hw.verbose_save(filename);
    else
        pause(2)
        close(h);
    end
end

function verbose_save(filename)
    fprintf('Saving Results for:\n\t %s \n', filename);
end
```

SAVE DATA TO TXT FILE

```
function mat2txt(filename, X, permission, header)
% ND.MAT2TXT  Write a matrix X into a txt file
%   mat2txt(filename, X, 'w', header) - Overwite the file
%   mat2txt(filename, X, 'a', header) - Append to the file end
%
%   See also.
        [I, J] = size(X);
        fileID = fopen(filename, permission);
        fprintf(fileID, [repelem('-', strlength(header)+3), '\n',
 header, ...
                '\n', repelem('-', strlength(header)+3), '\n']);
        fprintf(fileID, 'X(%d, %d)\n', I, J);
            for ii = 1:I
                for jj = 1:J
                    fprintf(fileID, ' %2.4f', X(ii,jj));
                end
                fprintf(fileID, ';\n');
```

```
            end
        fprintf(fileID, '\n');
        fclose(fileID);
end


% end methods list

end
end
```

```
ans =

  filter_hw with no properties.
```

Published with MATLAB® R2021a

Função Main

[TIP7188 - Filtragem Adaptativa] Author: Lucas Abdalah

main.m

```
clearvars;
close all;
clc; pause(0.1)

% publish('main.m', 'pdf');
% publish('filter_hw.m', 'pdf');

% filter_hw.hw2p5();
% filter_hw.hw3p4();
% filter_hw.hw3p5();
% filter_hw.hw3p6();
% filter_hw.hw4p1();
% filter_hw.hw4p3();
```

Published with MATLAB® R2021a