UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA DE TELEINFORMÁTICA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE TELEINFORMÁTICA

MESTRADO ACADÊMICO EM ENGENHARIA DE TELEINFORMÁTICA

TIP7188 - FILTRAGEM ADAPTATIVA

RÔMULO BANDEIRA PIMENTEL DRUMOND

TRABALHO AP2: MÉTODOS DOS MÍNIMOS QUADRADOS E APRENDIZADO POR TEORIA DA INFORMAÇÃO

FORTALEZA

2018

SUMÁRIO

1 QUESTÃO 1

1.1 Enunciado

Use o algoritmo RLS complexo para equalizar um canal com a função de transferência dada por

$$H(z) = (0.34 - 0.27j) + (0.87 + 0.43j)z^{-1} + (0.34 - 0.21j)z^{-2} \tag{1.1}$$

O sinal de entrada é um sinal de modulação 4-QAM representando uma sequência de bits com relação sinal-ruído $\frac{\sigma_{\tilde{x}}^2}{\sigma_n^2} = 20$ no receptor. Ou seja, $\widetilde{x}(k)$ é o sinal recebido sem levar em consideração o ruído adicional do canal. O filtro adaptativo tem 10 (dez) coeficientes.

(a) Use um valor apropriado para λ no intervalo 0.95 - 0.99, execute o algoritmo e comente o comportamento da convergência.

(b) Mostre o gráfico da parte real versus imaginária do sinal recebido antes e depois da equalização.

(c) Aumente o número de coeficientes do filtro para 20 (vinte) e repetida o experimento em (b).

1.2 Resolução

Para as simulações de equalização do canal foram escolhidos como hiper-parâmetros, além dos expostos no enunciado da questão:

- Número de símbolos processados = 10.000;
- *Seed* do gerador de número aleatórios = 2018 (visa reprodutibilidade das simulações).

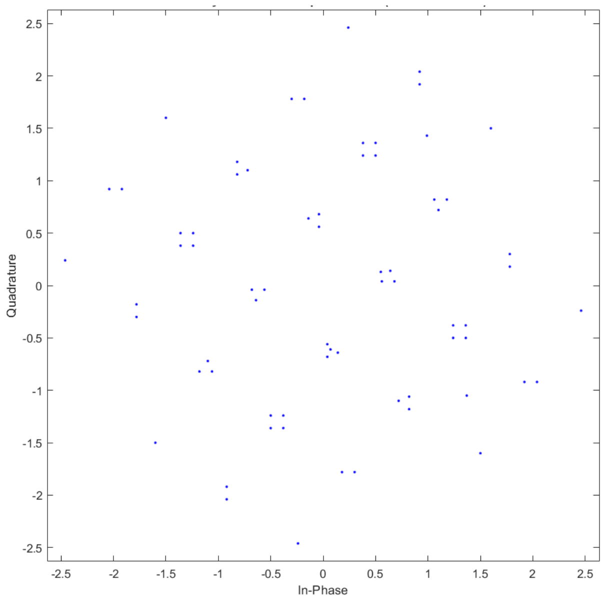
O código abaixo foi utilizado para gerar, aleatoriamente, os dados que foram transmitidos pelo canal e recebidos pelo filtro:

```
close all
clear all
clc

% Parâmetros
ns = 1e4;     % Total de símbolos a serem processados
C = 4;        % Tamanho da constelação QAM
snr = 10*log10(20); % Signal-to-noise ratio em dB
k = log2(C);     % k bits por simbolo.
```

```

% Gera o vetor de ns símbolos (ns*k bits) aleatórios
rng(2018); % fixa a semente do gerador de números aleatórios
data = randi([0 1],ns*k,1);

d = qammod(data,C, 'InputType','bit'); % Modulação QAM

% Transmissão pelo canal H e adição de ruído branco
h = [0.34 - 0.21j; 0.87 + 0.43j; 0.34 - 0.27j]; % parâmetros do canal
X = conv(d,h); % X contem o dados após passarem pelo canal

% Plota a constelação de X
scatterplot(X); title("Constelação dos dados após canal H (antes do AWGN)");

% Acrescenta ruido AWGN a X
rng(2018);
X = awgn(X,snr,'measured');

% Plota a constelação de X com ruído
scatterplot(X);
title(sprintf("Constelação antes do filtro \n(após as distorções do canal H e AWGN)"));
```

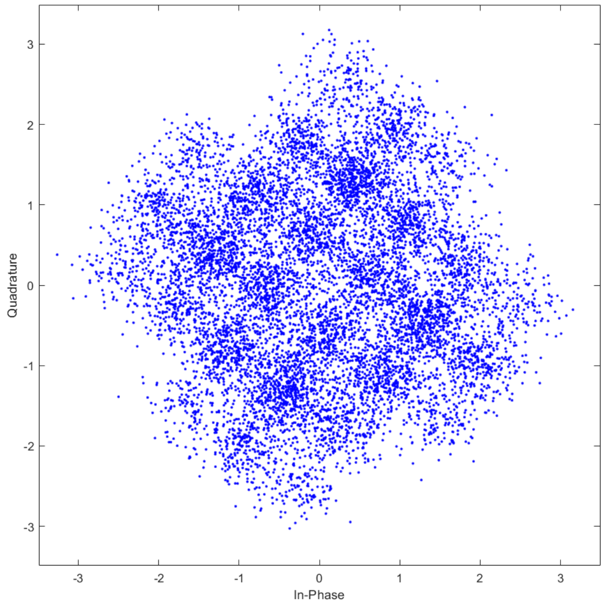
Como produto do código acima as Figuras 1 e 2 foram geradas, deixando explícito os efeitos do canal e do ruído na constelação dos dados.

Figura 1 – Constelação dos dados após canal H (antes do AWGN).



Fonte: O autor.

Figura 2 – Constelação dos dados antes do filtro (após distorções do canal H e AWGN).



Fonte: O autor.

Posteriormente foi aplicado o algoritmo *Recursive Least Squares* (RLS) para treinar o filtro linear. Para a escolha do hiper-parâmetro λ foi realizada uma busca em grade entre os valores 0.95 e 0.99. Como métricas de avaliação foram utilizadas:

- *Symbol Error Rate* (SER);
- *Bit Error Rate* (BER);
- Média do Módulo do Erro (MME).

As duas funções auxiliares abaixo foram criadas para tornar o código mais legível:

```
function    [Y, w, e] = RLS(d, X, n_coef, lambda, delta)
    w = zeros(n_coef,1);
    n = length(d); m = n_coef;
    S = delta*eye(m);
    Y = zeros(n-m+1,1); % valor predito a priori
    e = zeros(n-m+1,1); % erro a priori
    for N=1:n-m+1
        x_in = X(N:N+m-1,1); % entrada do filtro
        Num = (lambda^-1)*S*x_in;
        Den = (1 + (lambda^-1) * x_in'*S*x_in);
        k = Num/Den;
        Y(N,1) = w'*x_in;
        e(N,1) = d(N,1) - Y(N,1);
        w = w + k*e(N,1)';
        S = (lambda^-1)*S - (lambda^-1)*k*x_in'*S;
    end
```

```
end


function [SER, BER, MME] = evalMetrics(d,Y,e,C)
    k = log2(C);
    dataSent_symbols     = qamdemod(d, C);
    dataReceived_symbols = qamdemod(Y, C);

    dataReceived_binary = de2bi(dataReceived_symbols, k);
    dataSent_binary     = de2bi(dataSent_symbols, k);

    % Taxa de erro de simbolos (Symbol Error Rate):
    SER=sum((dataReceived_symbols == dataSent_symbols(1:length(dataReceived_symbols))) == 0)/length(dataReceived_symbols);

    % Taxa de erro de bits (Bit Error Rate):
    BER=sum(sum(dataReceived_binary == dataSent_binary(1:length(dataReceived_binary),:) == 0))/numel(dataReceived_binary);

    % Média dos Módulos dos Erros
    MME = mean(abs(e));
end
```

O código abaixo gerou a Figura 3 e apresentou o λ_{opt} para cada uma das métricas utilizadas:

```
% Parâmetros do RLS:
n_coef = 10;
delta = 1e-2;
lambdas = linspace(0.95, 0.99, 100);

% Aplicação do RLS e avaliação para cada valor de lambda:
tic
SER=zeros(size(lambdas)); BER=zeros(size(lambdas)); MME=zeros(size(lambdas));
for i=1:length(lambdas)
    [Y, w, e]= RLS(d,X,n_coef,lambdas(i),delta);
    [SER(i), BER(i), MME(i)] = evalMetrics(d,Y,e,C);
end
toc

figure; plot(lambdas,1e2*SER); hold on; plot(lambdas, 1e2*BER); plot(lambdas,MME); legend('100*SER','100*BER','MME');
xlim([0.95 0.99]); xlabel('\lambda'); ylabel('Métricas'); legend('Location','northwest');
[minSER, iSER] = min(SER);
```

```
[minBER, iBER] = min(BER);
[minMME, iMME] = min(MME);
fprintf(...
    'SER_{minimum} = %e => lambda = %.6f\nBER_{minimum} = %e => lambda = %.6f\nMME_{minimum} = %e => lambda = %.6f', ...
    minSER, lambdas(iSER), minBER, lambdas(iBER), minMME, lambdas(iMME))
```

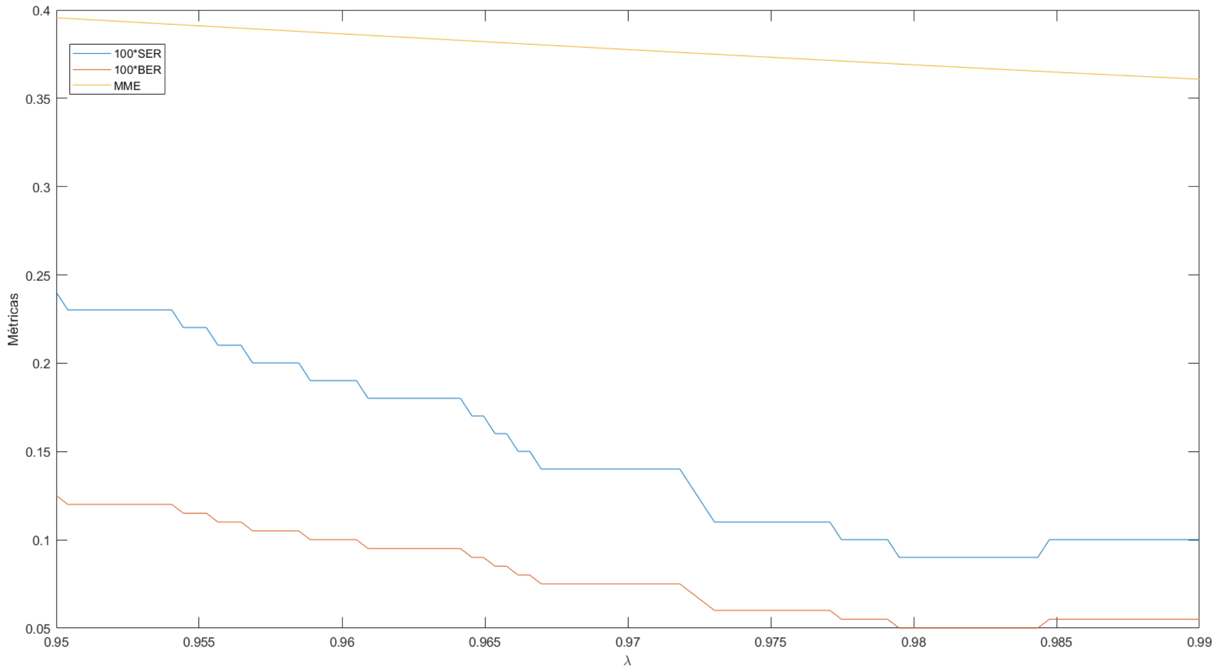
Elapsed time is 64.911534 seconds.

$\text{SER}_{\text{minimum}} = 9.008107e-04 => \lambda = 0.979495$

$\text{BER}_{\text{minimum}} = 5.004504e-04 => \lambda = 0.979495$

$\text{MME}_{\text{minimum}} = 3.608388e-01 => \lambda = 0.990000$

Figura 3 – Gráfico das métricas em função do valor de λ.



Fonte: O autor.

Como as métricas SER e BER concordaram com o valor de λ_{opt}, esse valor foi adotado nas simulações do filtro com RLS. O código abaixo foi utilizado para gerar a constelação dos dados após o filtro, Figura 4, e os gráficos do logaritmo do módulo do erro em função das iterações, Figuras 5 e 6.

```
lambda_opt = lambdas(iSER);
n_coef = 10;
[Y_10, w_10, e_10]= RLS(d,X,n_coef,lambda_opt,delta);
[SER_10, BER_10, MME_10] = evalMetrics(d,Y_10,e_10,C);

```

```
fprintf('Valor das três métricas quando len(w)=%d: \nSER = %e\nBER = %e\nMME = %e', ...
        n_coef, SER_10, BER_10, MME_10);

% Plota a constelação de Y
scatterplot(Y_10);
name = sprintf("Constelação depois do filtro. \n(len(w)=%d; lambda = %.4f; delta = %.2f)", 10, lambda_opt, delta);
title(name); grid on;

% erro ao longo das iterações
semilogy(abs(e_10));
xlabel('Iteração'), ylabel('Logarítmo do módulo do erro')
title('Erro ao longo das iterações [len(w) = 10]'); grid on;
semilogy(abs(e_10));
xlabel('Iteração'), ylabel('Logarítmo do módulo do erro')
title('Erro ao longo das iterações (com zoom) [len(w) = 10]'); grid on; xlim([0 600]);
```

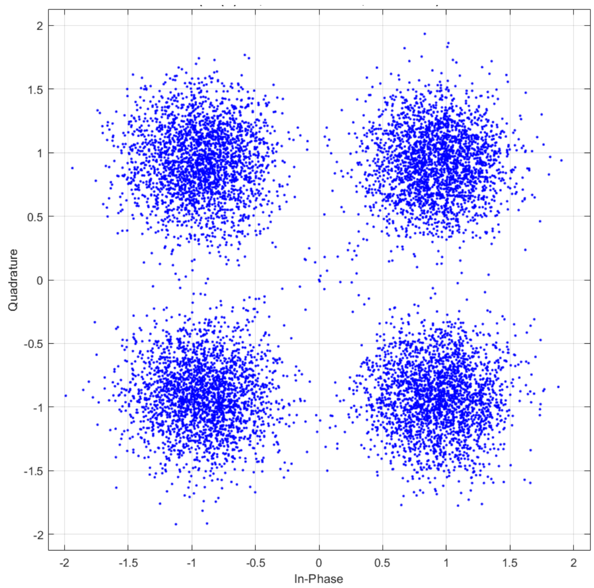
Valor das três métricas quando len(w)=10:

SER = 9.008107e-04

BER = 5.004504e-04

MME = 3.694145e-01

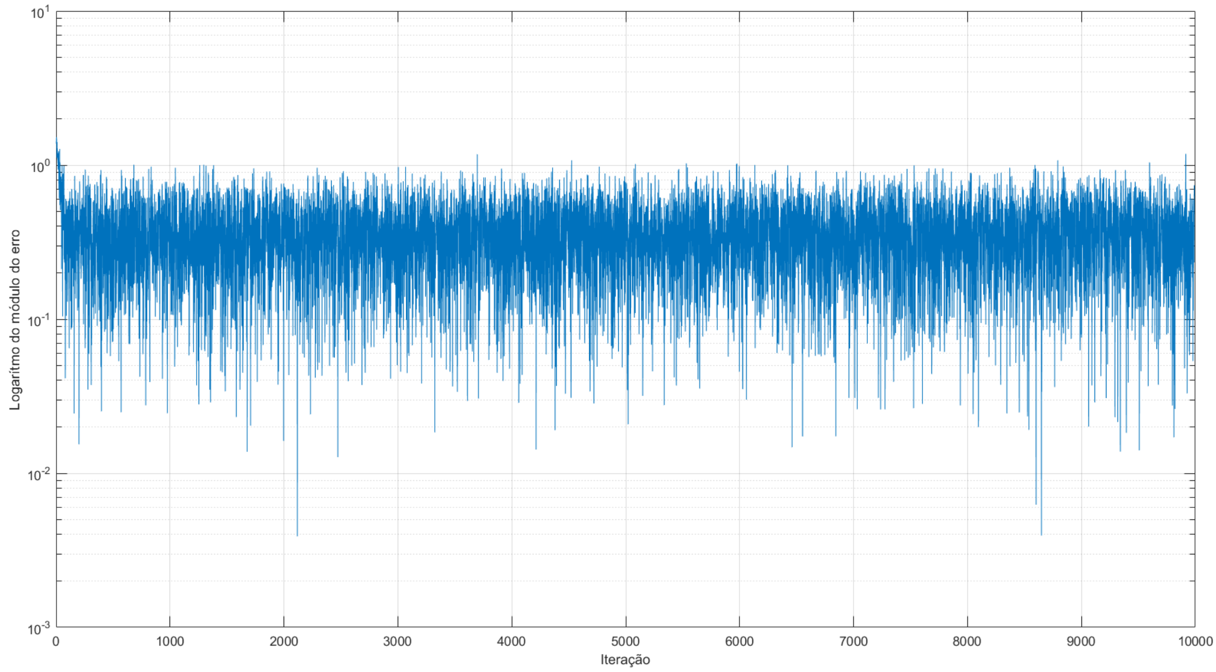
Figura 4 – Constelação dos dados após o filtro com RLS [$n_{coef} = 10$; $\lambda = 0.9795$; $\delta = 0.01$].



Fonte: O autor.

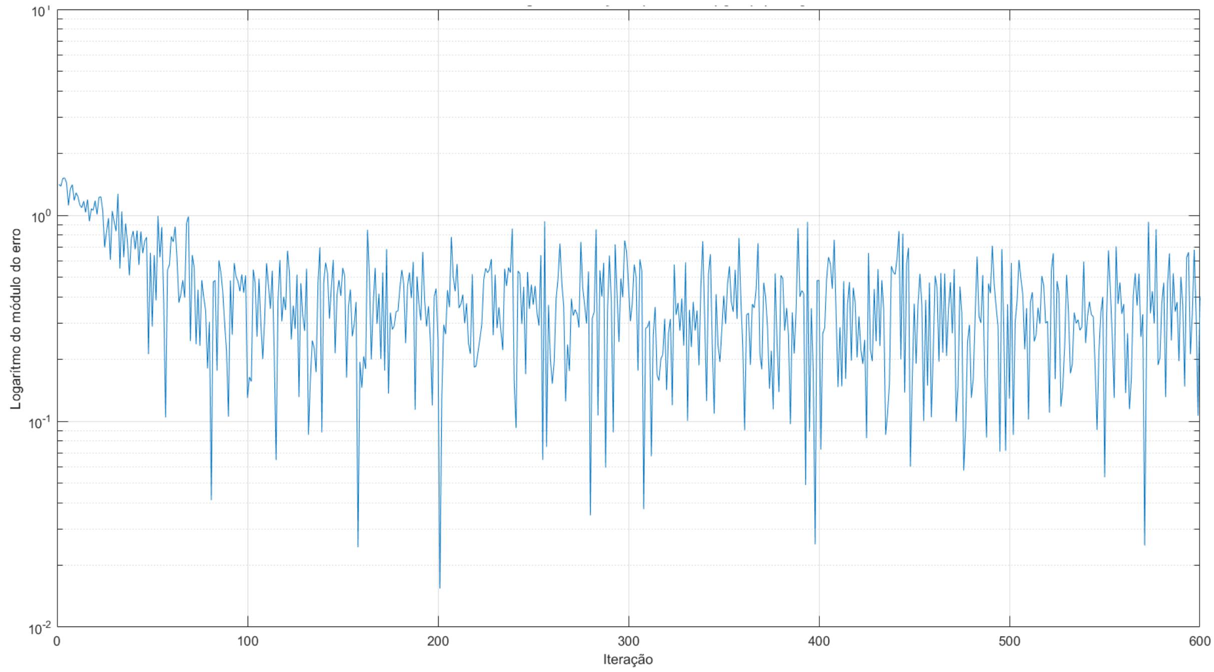
Pela Figura 4 podemos ver que o filtro equalizou bem o canal e que após aproxi-

Figura 5 – Gráfico do logaritmo do módulo do erro ao longo das iterações [$n_{coef} = 10$].



Fonte: O autor.

Figura 6 – Gráfico do logaritmo do módulo do erro ao longo das iterações (com zoom) [$n_{coef} = 10$].



Fonte: O autor.

madamente 100 iterações o algoritmo convergiu, como pode ser visto na Figura 6, restando a oscilação natural do desajuste dos valores do filtro.

O próximo passo foi **aumentar número de coeficientes do filtro para 20**, mantendo constante os outros hiper-parâmetros. O código abaixo avaliou o desempenho do novo filtro e gerou as Figuras 7, 8 e 9:

```
n_coef = 20;
[Y_20, w_20, e_20]= RLS(d,X,n_coef,lambda_opt,delta);
[SER_ITL, BER_20, MME_ITL] = evalMetrics(d,Y_20,e_20,C);

fprintf('Valor das três métricas quando len(w)=%d: \nSER = %f\nBER = %f\nMME = %f', ...
        20, SER_20, BER_20, MME_20);

% Plota a constelação de Y
scatterplot(Y_20);
name = sprintf("Constelação depois do filtro. \n(len(w)=%d; lambda = %.4f; delta = %.2f)", n_coef, lambda_opt, delta);
title(name); grid on;

semilogy(abs(e_20));
xlabel('Iteração'), ylabel('Logarítmo do módulo do erro')
title('Erro ao longo das iterações [len(w)=20]'); grid on;

semilogy(abs(e_20));
xlabel('Iteração'), ylabel('Logarítmo do módulo do erro')
title('Erro ao longo das iterações (com zoom) [len(w)=20]'); grid on; xlim([0 600]);
```

Valor das três métricas quando len(w)=20:

SER = 0.002605

BER = 0.001403

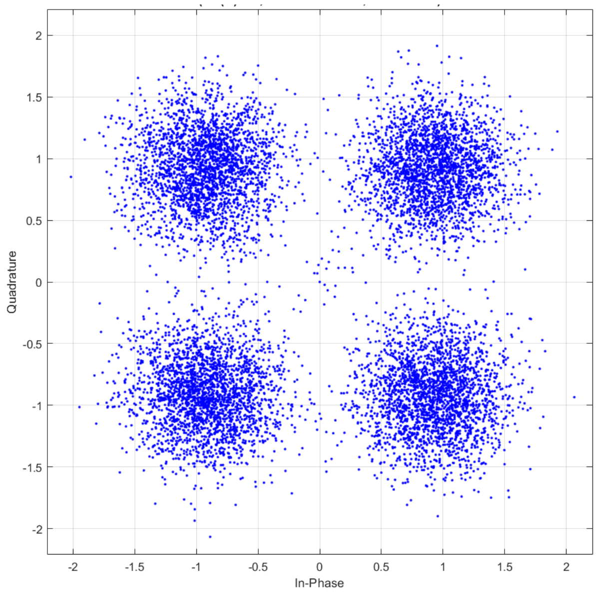
MME = 0.387526

Pelas três métrica de avaliação pode-se dizer que o filtro teve pior desempenho quando seu número de coeficientes aumentou para 20. Embora seja sutil é possível perceber que as constelações na Figura 7 apresentam uma dispersão maior do que o da Figura 4. Já os gráficos do erro ao longo das iterações não diferem muito para esses dois valores de coeficientes do filtro, a convergência na Figura 9 acontece em aproximadamente 100 iterações como na Figura 6.

Como formar de incrementar a análise foi estudado o efeito do número de coeficientes do filtro nas três métricas de avaliação. O código abaixo faz uma busca em grade pelo número de coeficientes (de 2 a 100) e calcula o valor das métricas para cada caso, gerando o gráfico da Figura 10:

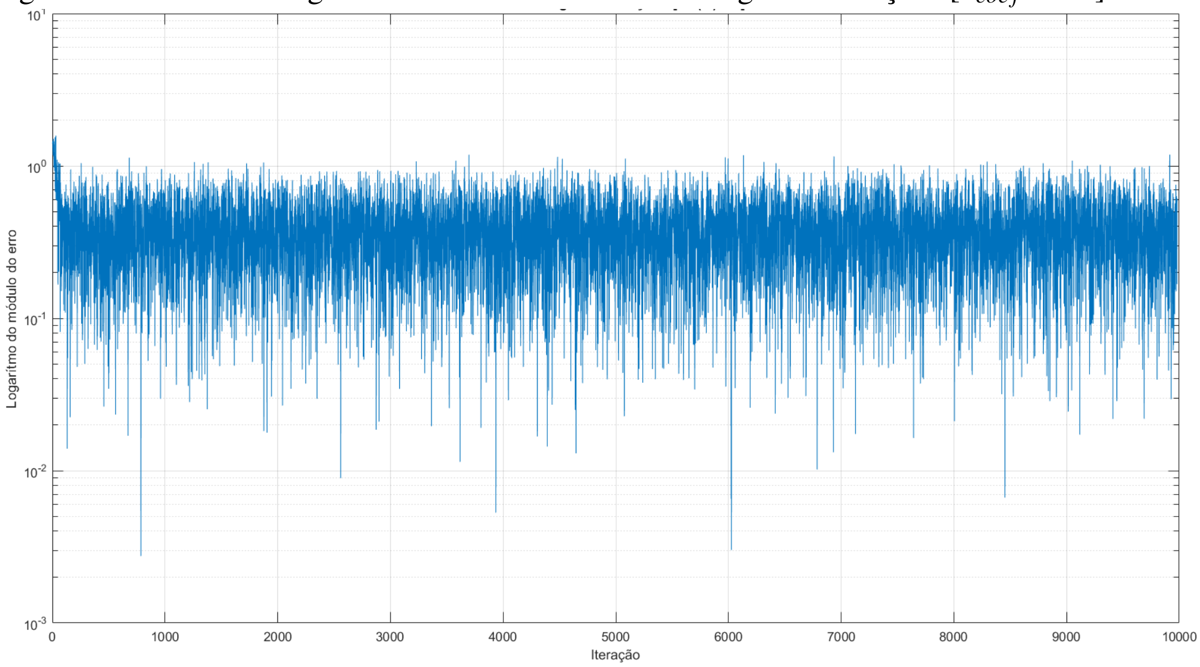
```
% Parâmetros:
n_coefs = 2:100; % de 2 a 100 coeficientes
% Aplicação do RLS e avaliação das métricas para cada valor de n_coef:
```

Figura 7 – Constelação dos dados após o filtro com RLS [$n_{coef} = 20$; $\lambda = 0.9795$; $\delta = 0.01$].



Fonte: O autor.

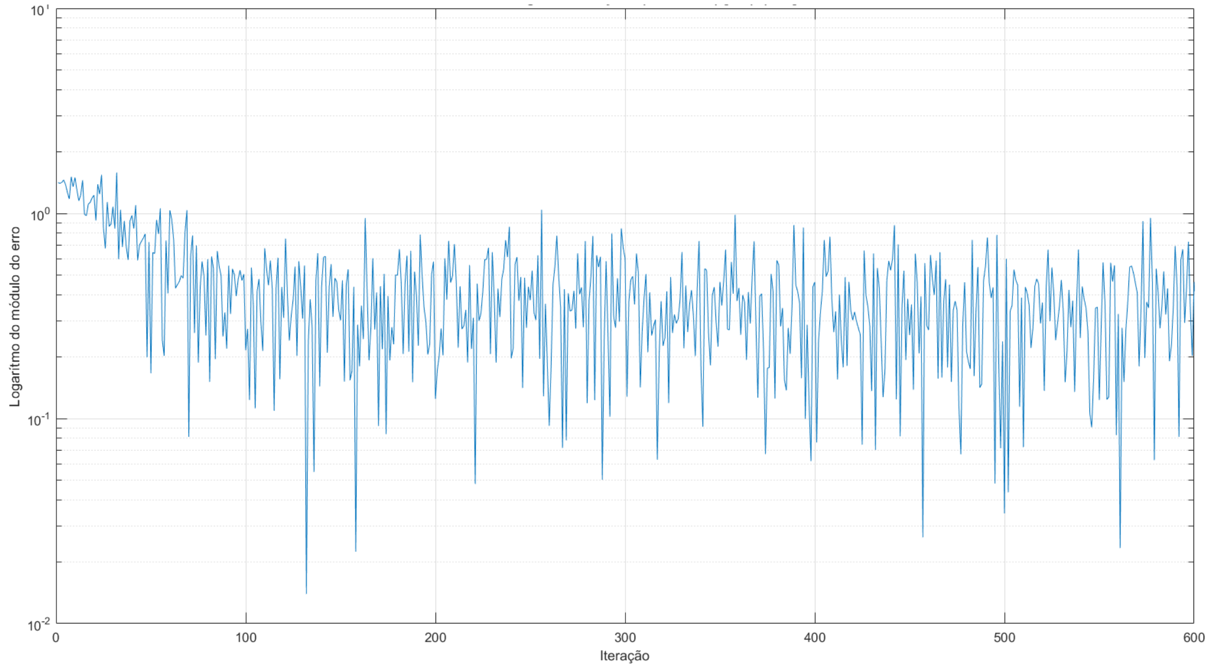
Figura 8 – Gráfico do logaritmo do módulo do erro ao longo das iterações [$n_{coef} = 20$].



Fonte: O autor.

```
tic
SER=zeros(size(n_coefs)); BER=zeros(size(n_coefs)); MME=zeros(size(n_coefs));
for i=1:length(n_coefs)
    [Y, w, e]= RLS(d,X,n_coefs(i),lambda_opt,delta);
    [SER(i), BER(i), MME(i)] = evalMetrics(d,Y,e,C);
end
toc

```

Figura 9 – Gráfico do logaritmo do módulo do erro ao longo das iterações (com zoom) [$n_{coef} = 20$].



Fonte: O autor.

```
figure; plot(n_coefs,1e2*SER,'LineWidth',2); hold on; %xlim([n_coefs(1) n_coefs(end)]);
plot(n_coefs, 1e2*BER,'LineWidth',2); plot(n_coefs,MME,'LineWidth',2); legend('100*SER','100*BER','MME');
xlabel('Número de coeficientes do filtro'); ylabel('Métricas'); grid on;
title('Efeito do número de coeficientes do filtro nas métricas de avaliação')

[minSER, iSER] = min(SER);
[minBER, iBER] = min(BER);
[minMME, iMME] = min(MME);
fprintf(...
    'SER_{minimum} = %e => n_coef = %d\nBER_{minimum} = %e => n_coef = %d\nMME_{minimum} = %e => n_coef = %d', ...
    minSER, n_coefs(iSER), minBER, n_coefs(iBER), minMME, n_coefs(iMME))
```

Elapsed time is 846.398283 seconds.

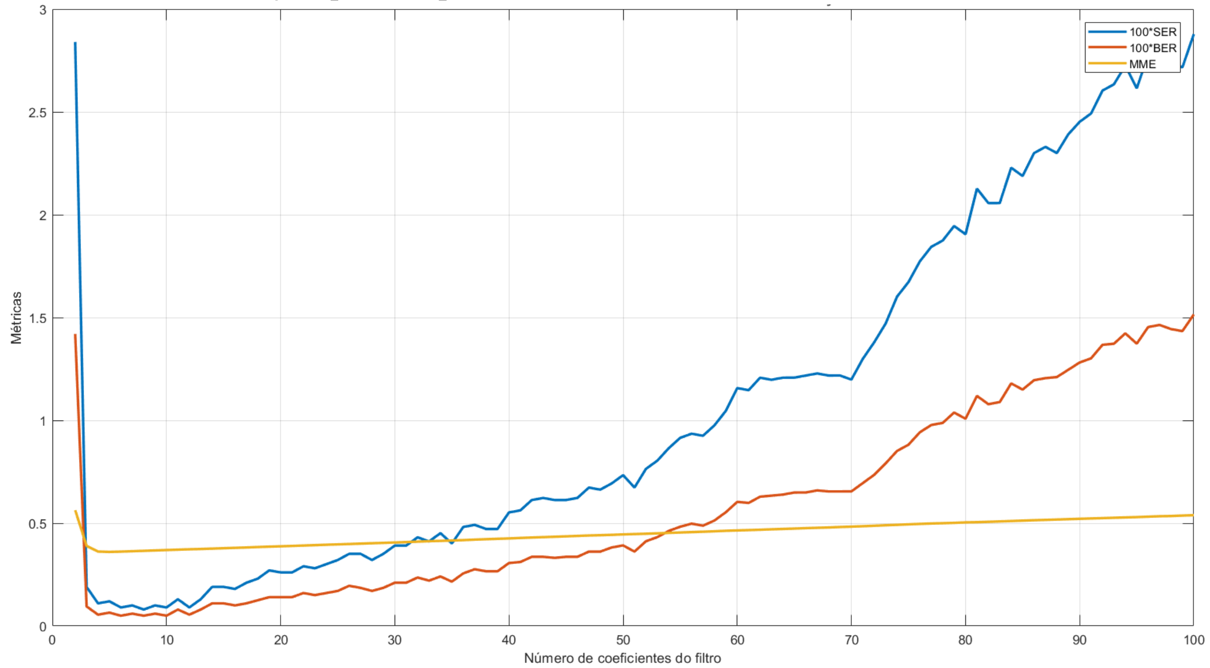
$SER_{minimum} = 8.005604e-04 => n_{coef} = 8$

$BER_{minimum} = 5.002501e-04 => n_{coef} = 6$

$MME_{minimum} = 3.603195e-01 => n_{coef} = 5$

Podemos ver na Figura 10 que há uma tendência de piora na performance do filtro com o aumento do número de coeficientes na faixa estudada (de 10 a 20). De forma a deixar mais explícito o decaimento de performance foram feitos os gráficos das constelações após o

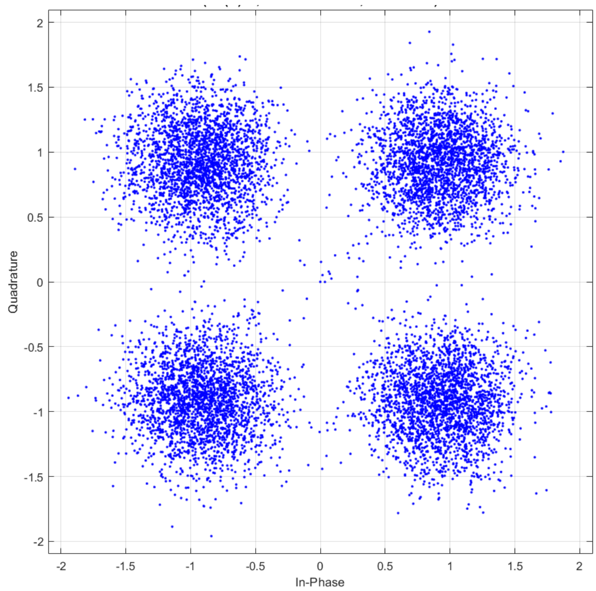
Figura 10 – Gráfico mostrando a relação entre o número de coeficientes do filtro e as métricas de avaliação quando aplicado RLS [$\lambda = 0.9795$; $\delta = 0.01$].



Fonte: O autor.

filtro para os casos em que $n_{coef} = 8$, valor ótimo segundo o SER, e $n_{coef} = 100$, que apresenta uma performance deplorável segundo a Figura 10. Os resultados podem ser vistos nas Figuras 11 e 12.

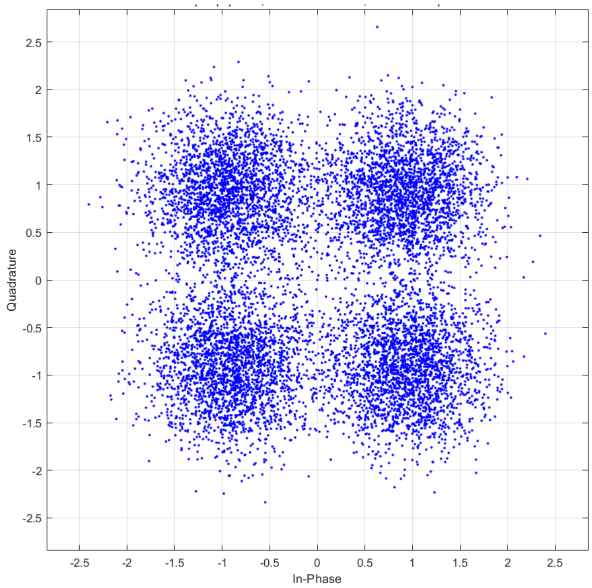
Figura 11 – Constelação dos dados após o filtro com RLS [$n_{coef} = 8$; $\lambda = 0.9795$; $\delta = 0.01$].



Fonte: O autor.

Agora ficou clara a dispersão maior da constelação de dados, ou queda de perfor-

Figura 12 – Constelação dos dados após o filtro com RLS [$n_{coef} = 100$; $\lambda = 0.9795$; $\delta = 0.01$].



Fonte: O autor.

mance, ocasionada pelo aumento do número de coeficientes do filtro.

2 QUESTÃO 2

2.1 Enunciado

Repita o problema 1 utilizando um método de filtragem por meio do aprendizado de teoria da informação (ITL). Utilize um *kernel* gaussiano para compor a sua função de custo. Compare os resultados da equalização com o RLS.

2.2 Resolução

Para essa questão foi utilizado o *Error Correntropy Criterion* (ECC), com *kernel* gaussiano e janela unitária. Consequentemente a função custo utilizada foi:

$$J(n) = \frac{1}{\sqrt{2\pi\sigma}} e^{-(d(n)-y(n))^2/2\sigma^2} \tag{2.1}$$

Onde $y(n)$ é a saída do filtro no instante n.

Portanto temos como regra de atualização, utilizando gradiente ascendente:

$$w(n+1) = w(n) + \frac{\eta}{\sqrt{2\pi\sigma^3}} e^{-e(n)^2/2\sigma^2} e(n)x(n) \tag{2.2}$$

Onde $e(n)$ é o erro na iteração n:

$$e(n) = d(n) - y(n) \tag{2.3}$$

A função exibida no código abaixo foi criada para aplicar o gradiente ascendente com EEC:

```
function [Y, w, e] = gaEEC(d,X,n_coef,eta,sigma)
    w = zeros(n_coef,1);
    n = length(d); m = n_coef;
    Y = zeros(n-m+1,1); % valor predito a priori
    e = zeros(n-m+1,1); % erro a priori
    for N=1:n-m+1
        x_in = X(N:N+m-1,1); % entrada do filtro
        Y(N,1) = w'*x_in;
        e(N,1) = d(N,1) - Y(N,1);
        deltaJ = 1/(sqrt(2*pi)*sigma^3) * exp(-e(N,1)^2/(2*sigma^2)) * e(N,1) * x_in';
        w = w + eta*deltaJ';
    end
end
```

Para escolhermos o valor de η foi realizada uma busca em grade pelo hiper-parâmetro, de forma similar à busca pelo valor de λ no RLS. O código abaixo implementa tal busca e gera a Figura 13:

```
% Hiper-parâmetros
etas = linspace(2.75e-2,1e-5,100);
n_coef = 10;
sigma = 1;

% Aplicação do gaECC e avaliação para cada valor de eta:
tic
SER_ITL=zeros(size(etas)); BER_ITL=zeros(size(etas)); MME_ITL=zeros(size(etas));
for i=1:length(etas)
    [Y, ~, e] = gaEEC(d,X,n_coef,etas(i),sigma);
    [SER_ITL(i), BER_ITL(i), MME_ITL(i)] = evalMetrics(d,Y,e,C);
end
toc

figure; semilogx(etas,1e2*SER_ITL,'LineWidth',2); hold on; semilogx(etas,
    1e2*BER_ITL,'LineWidth',2);
semilogx(etas,MME_ITL,'LineWidth',2); legend('100*SER','100*BER','MME'); grid on;
    xlim([etas(end) etas(1)]);
xlabel('\eta'); ylabel('Métricas');
title('Valor das métricas em função do eta [len(w)=10]')

[minSER_ITL, iSER_ITL] = min(SER_ITL);
[minBER_ITL, iBER_ITL] = min(BER_ITL);
[minMME_ITL, iMME_ITL] = min(MME_ITL);
fprintf(...
    ['SER_ITL{minimum} = %e => eta = %.4e\n' ...
     'BER_ITL{minimum} = %e => eta = %.4e\n' ...
     'MME_ITL{minimum} = %e => eta = %.4e'], ...
    minSER_ITL, etas(iSER_ITL), minBER_ITL, etas(iBER_ITL), minMME_ITL,
        etas(iMME_ITL))
```

Elapsed time is 439.741252 seconds.

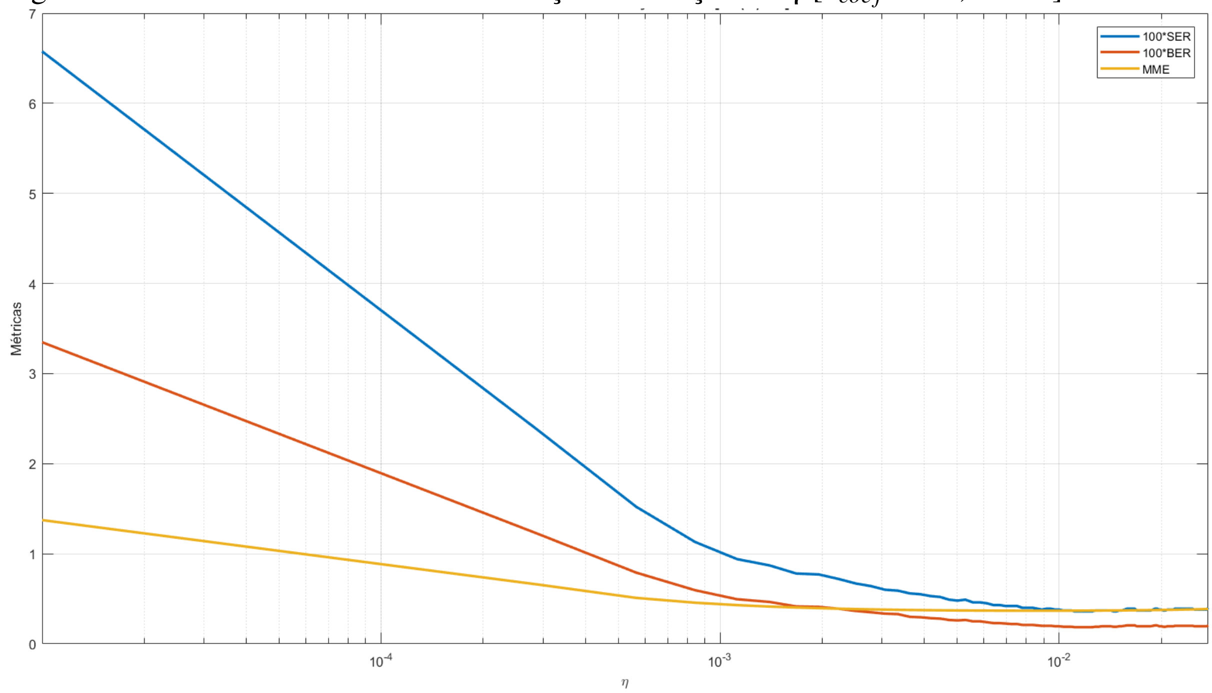
$SER\_ITL_{minimum} = 3.603243e-03 => \eta = 1.4727e-02$

$BER\_ITL_{minimum} = 1.851666e-03 => \eta = 1.2505e-02$

$MME\_ITL_{minimum} = 3.677712e-01 => \eta = 8.6180e-03$

Como valor ótimo de η foi escolhido o que minimiza o SER. O código baixo gerou os gráficos da constelação de dados e erro ao longo das iterações quando usa-se o η_{opt} (os

Figura 13 – Gráfico das métricas de avaliação em função de η [$n_{coef} = 10$; $\sigma = 1$].



Fonte: O autor.

resultados podem ser vistos nas Figuras 14, 15 e 16).

```
eta_opt = etas(iSER_ITL);
[Y, ~, e] = gaEEC(d,X,n_coef,eta_opt,sigma);
figure; scatterplot(Y);
name = sprintf("Constelação depois do filtro (ITL). \n[len(w)=%d; eta=%.2e; sigma=%.2f]", ...
    n_coef, eta_opt, sigma);
title(name); grid on;

figure; semilogy(abs(e));
xlabel('Iteração'), ylabel('Logarítmo do módulo do erro')
title(sprintf('Erro ao longo das iterações (ITL) [len(w)=%d]', n_coef)); grid on;
figure; semilogy(abs(e)); xlim([0 600]);
xlabel('Iteração'), ylabel('Logarítmo do módulo do erro')
title(sprintf('Erro ao longo das iterações (ITL) (com zoom) [len(w)=%d]', n_coef)); grid on;

[SER_ITL, BER_ITL, MME_ITL] = evalMetrics(d,Y,e,C);

fprintf('Valor das três métricas quando len(w)=%d: \nSER = %f\nBER = %f\nMME = %f', ...
        n_coef, SER_ITL, BER_ITL, MME_ITL);
```

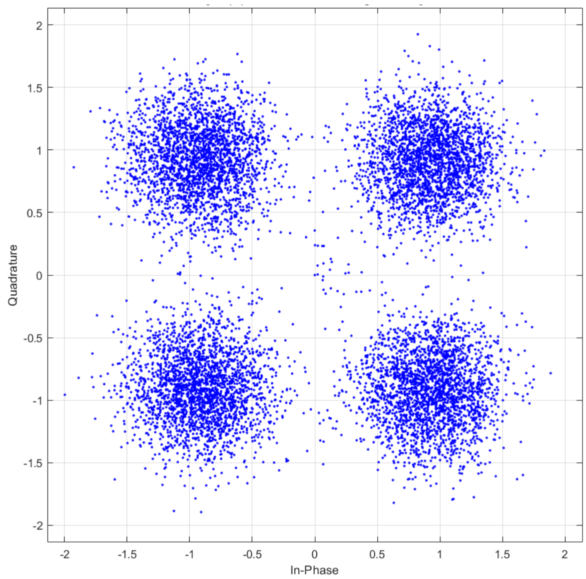
Valor das três métricas quando len(w)=10:

SER = 0.003603

BER = 0.001902

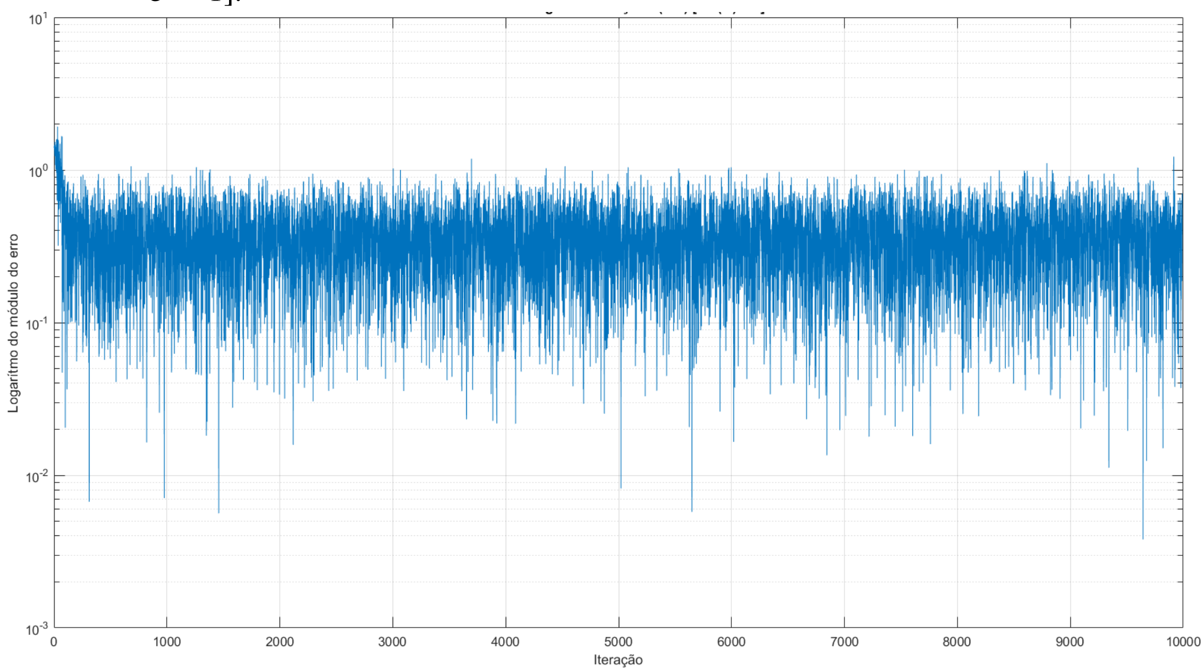
MME = 0.370957

Figura 14 – Constelação dos dados após o filtro com ITL [$n_{coef} = 10$; $\eta = 1.47e-2$ $\sigma = 1$].



Fonte: O autor.

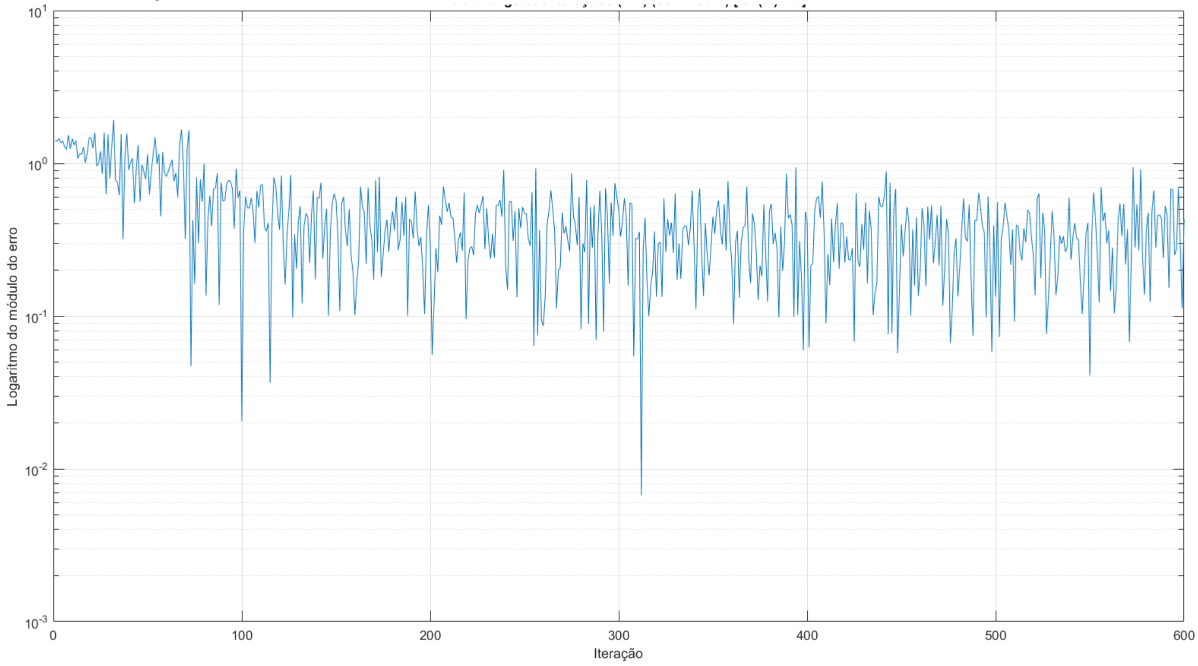
Figura 15 – Logaritmo do módulo do erro ao longo das iterações [$n_{coef} = 10$; $\eta = 1.47e-2$ $\sigma = 1$].



Fonte: O autor.

Podemos ver na Figura 16 que o algoritmo convergiu após, aproximadamente, 200

Figura 16 – Logaritmo do módulo do erro ao longo das iterações (com zoom) [$n_{coef} = 10$; $\eta = 1.47e-2$ $\sigma = 1$].



Fonte: O autor.

iterações, depois disso ocorreu as oscilações naturais do gradiente ascendente.

Posteriormente a **análise foi repetida para o caso em que o número de coeficientes do filtro era 20**.

Durante as simulações um problema foi encontrado: o algoritmo tinha problemas para convergir quando utilizado o η_{opt} encontrado para $n_{coef} = 10$, portanto foi realizada novamente uma busca pelo hiper-parâmetro η. Os resultados podem ser vistos abaixo e na Figura 17.

$SER\_ITL_{minimum} = 6.111612e-03 => \eta = 9.1927e-03$

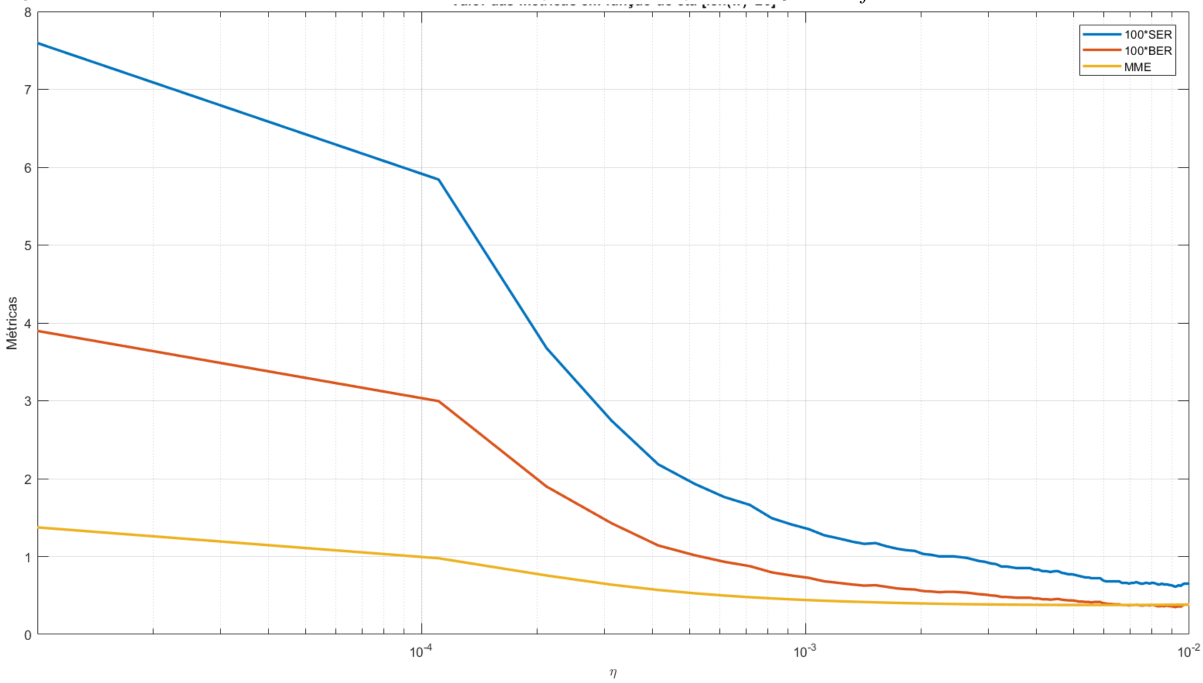
$BER_ITL_{minimum} = 3.506663e-03 => \eta = 9.1927e-03$

$MME_ITL_{minimum} = 3.768924e-01 => \eta = 5.7618e-03$

O valor de η que minimiza SER foi escolhido como ótimo e a constelação dos dados, assim como o erro ao longo das iterações foram gerados para o caso de $n_{coef} = 20$. Os resultados dessas simulações podem ser vistos nas Figuras 18, 19 e 20.

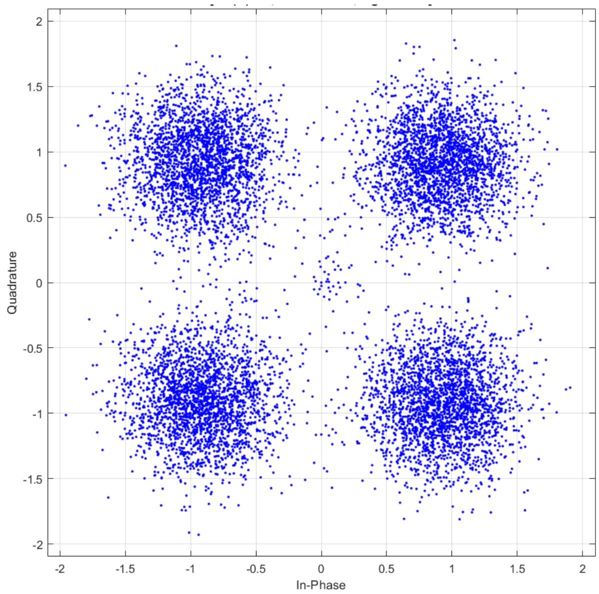
De forma geral o filtro com 10 coeficientes teve um desempenho superior segundo as 3 métricas utilizadas. Em termos de convergência o filtro com 20 coeficientes precisou aproximadamente de 400 iterações, número maior que o filtro com 10 coeficientes por causa do η menor. A constelação do filtro de 20 também apresenta mais pontos no centro, que devem

Figura 17 – Gráfico das métricas de avaliação em função de η [$n_{coef} = 20$; $\sigma = 1$].



Fonte: O autor.

Figura 18 – Constelação dos dados após o filtro ITL [$n_{coef} = 20$; $\eta = 9.19e-3$; $\sigma = 1$].

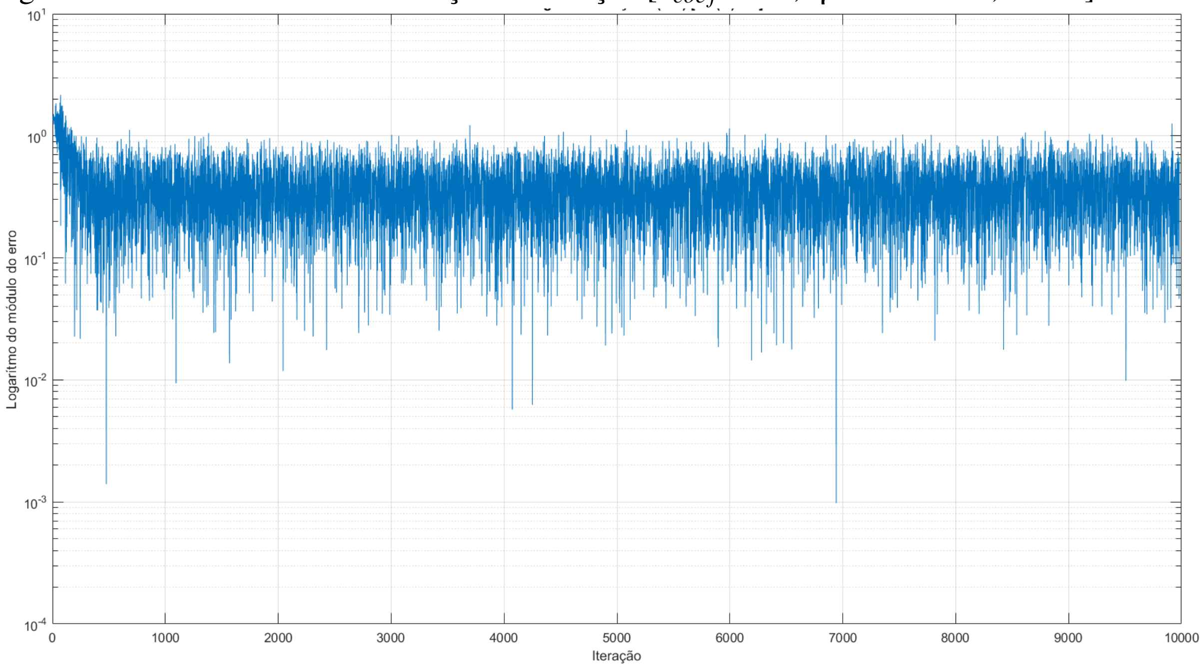


Fonte: O autor.

ser ocasionados pelo fator de passo menor, ou η, que faz com que o algoritmo precise de mais iterações para devidamente equalizar o canal .

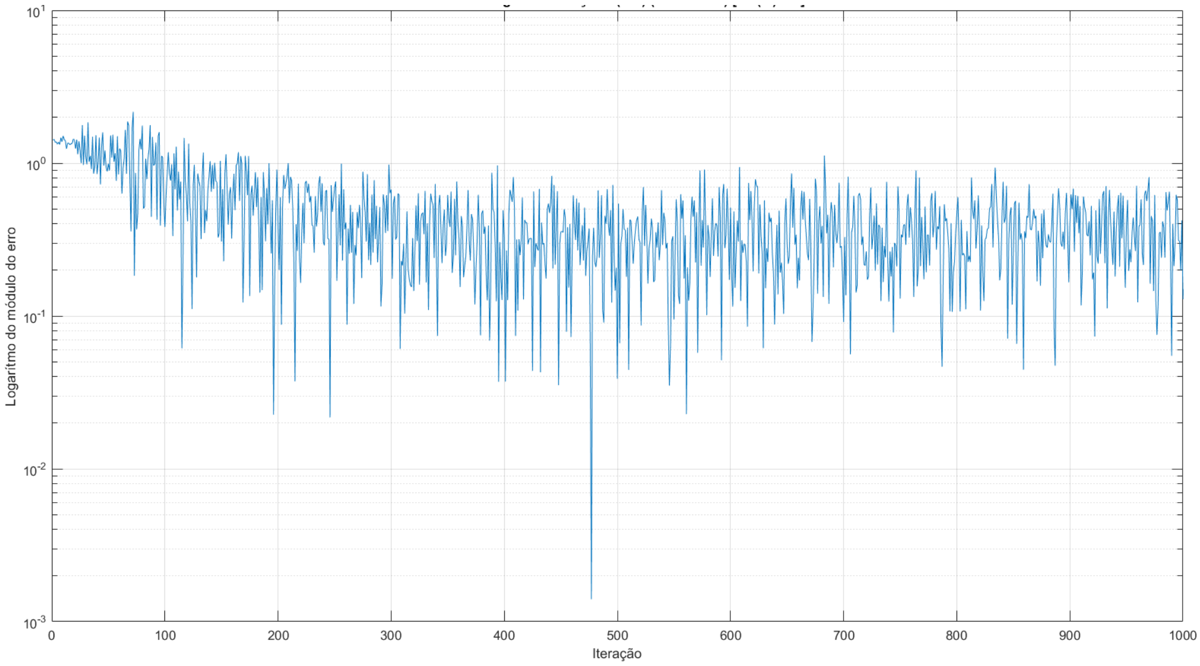
De forma análoga ao caso do RLS variou-se o número de coeficientes do filtro para ver seu impacto nas métricas de avaliação. Para tal simulação foi escolhido um valor de η que levasse à convergência os filtros de 2 a 100 coeficientes. O valor $\eta = 2.5e-3$ foi escolhido

Figura 19 – Gráfico do erro em função da iteração [$n_{coef} = 20$; $\eta = 9.19e-3$; $\sigma = 1$].



Fonte: O autor.

Figura 20 – Gráfico do erro em função da iteração (com zoom) [$n_{coef} = 20$; $\eta = 9.19e-3$; $\sigma = 1$].



Fonte: O autor.

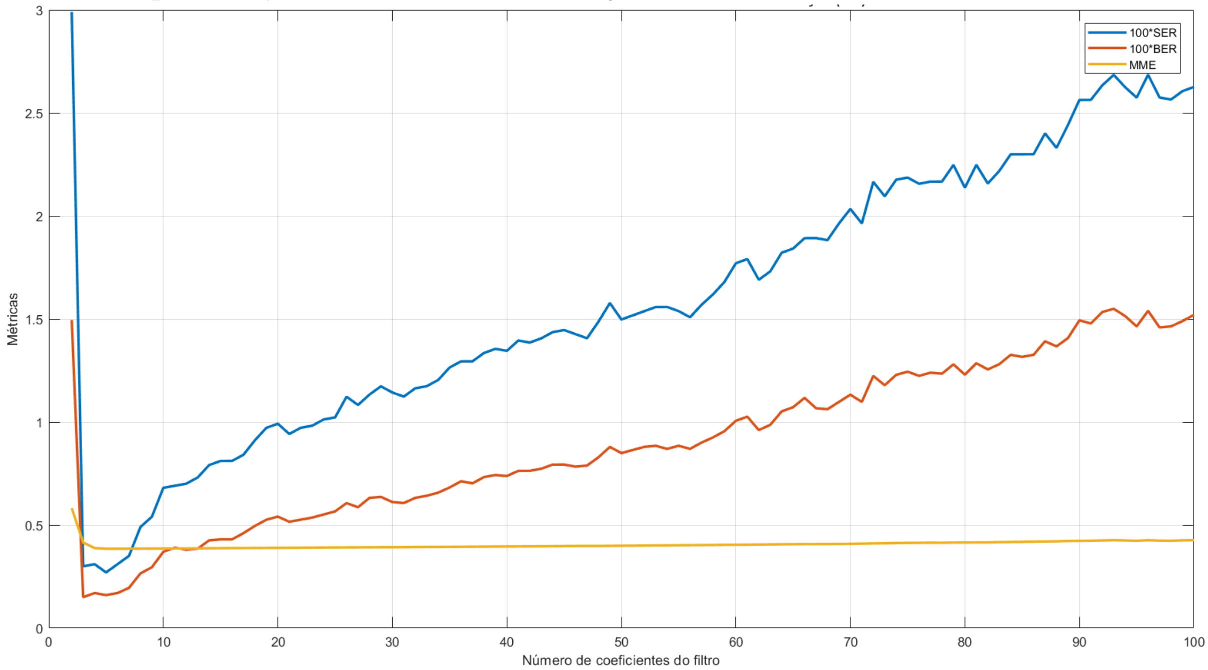
através de tentativa e erro. Os resultados dessa simulação podem ser vistos abaixo e na Figura 21.

$SER\_(ITL)_{minimum} = 2.701080e-03 => n_{coef} = 5$

$BER\_(ITL)_{minimum} = 1.500300e-03 => n_{coef} = 3$

$MME\_(ITL)_{minimum} = 3.850355e-01 => n_{coef} = 6$

Figura 21 – Gráfico das métricas de avaliação em função do número de coeficientes do filtro para o algoritmo baseado em ITL [$\eta = 2.5e-3$; $\sigma = 1$].



Fonte: O autor.

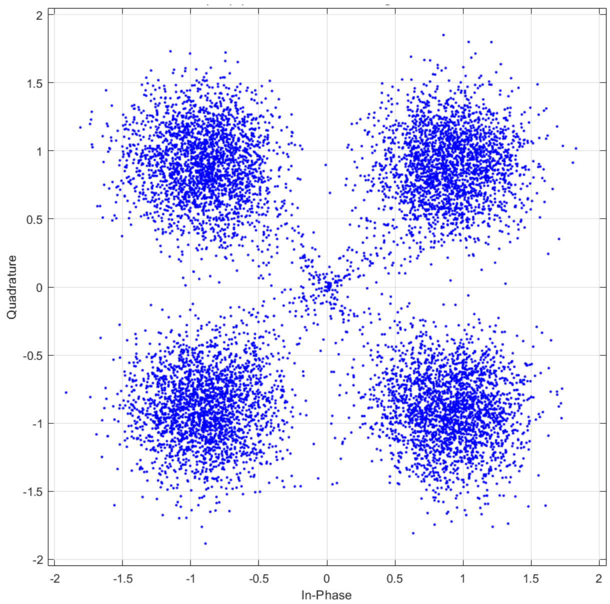
Como no caso do RLS gerou-se os gráficos para as constelações do melhor e pior n_{coef} segundo a métrica SER, os resultados podem ser vistos nas Figuras 22 e 23.

Podemos ver que, como no caso do RLS, o aumento no número de coeficientes do filtro ocasionou redução da performance da equalização no problema estudado.

De forma geral o RLS apresentou melhor desempenho do que o EEC com gradiente ascendente. O SER dos exemplos com RLS sempre foram inferiores do que os baseados em ITL, tal diferença pode ter sido ocasionada por:

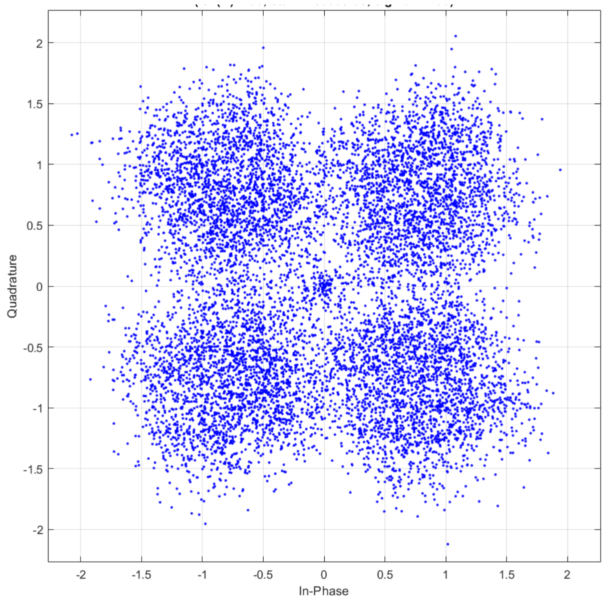
- Mal ajuste dos hiper-parâmetros;
- A baixa complexidade do Gradiente Ascendente + EEC quando comparado ao RLS;
- A natureza gaussiana do ruído aditivo, sendo que o ITL teoricamente apresenta vantagens na presença de ruído não-gaussiano (PRINCIPE, 2010);
- Outra particularidade do problema beneficia o RLS sobre a técnica baseada em ITL utilizada.

Figura 22 – Constelação dos dados do melhor caso com ITL [$n_{coef} = 5$; $\eta = 2.5e-3$; $\sigma = 1$].



Fonte: O autor.

Figura 23 – Constelação dos dados do pior caso com ITL [$n_{coef} = 100$; $\eta = 2.5e-3$; $\sigma = 1$].



Fonte: O autor.

REFERÊNCIAS

PRINCIPE, J. C. **Information Theoretic Learning: Renyi's Entropy and Kernel Perspectives**. 1st. ed. [S.l.]: Springer Publishing Company, Incorporated, 2010. ISBN 1441915699, 9781441915696.