MAGNO TEÓFILO MADEIRA DA SILVA

**EQUALIZAÇÃO NÃO-LINEAR DE CANAIS
DE COMUNICAÇÃO**

Dissertação apresentada à Escola
Politécnica da Universidade de
São Paulo para obtenção do
título de Mestre em Engenharia.

São Paulo
2001

MAGNO TEÓFILO MADEIRA DA SILVA

**EQUALIZAÇÃO NÃO-LINEAR DE CANAIS
DE COMUNICAÇÃO**

Dissertação apresentada à Escola
Politécnica da Universidade de
São Paulo para obtenção do
título de Mestre em Engenharia.

Área de Concentração:
Sistemas Eletrônicos

Orientador:
Prof. Dr. Max Gerken

São Paulo
2001

Silva, Magno Teófilo Madeira da

Equalização não-linear de canais de comunicação. São Paulo, 2001.

177p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Telecomunicações e Controle.

1. Equalização não-linear 2. Redes neurais artificiais 3. Algoritmos adaptativos I. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Telecomunicações e Controle II. t

Agradecimentos

Ao amigo e orientador Max Gerken pelas idéias e permanente incentivo.

À Maria D. Miranda pelas discussões freqüentes a respeito de equalização não-linear que tiveram uma grande importância no desenvolvimento deste trabalho.

Aos meus familiares pelo apoio e por tudo que representam para mim.

Aos amigos e colegas do LCS que de forma direta ou indireta contribuíram para a realização deste trabalho.

À FAPESP pelo financiamento deste projeto de pesquisa.

Resumo

É investigado o uso de redes neurais aplicadas à equalização de canais de comunicação, sendo consideradas três tipos de redes: MLP (*Multilayer Perceptron*), RBF (*Radial Basis Function*) e RNN (*Recurrent Neural Network*). Os equalizadores não-lineares baseados nestas redes foram comparados com o equalizador linear transversal e com os equalizadores ótimos segundo os critérios de Bayes e da máxima verossimilhança. Nestas comparações foram utilizados um alfabeto binário e um quaternário transmitidos em modelos de canais cuja resposta ao pulso unitário é finita. Além das versões usuais de equalizadores, foram consideradas versões com realimentação de decisões sempre que isso se mostrou adequado. O treinamento desses equalizadores foi feito de forma supervisionada, ou seja, na fase de treinamento a sequência de símbolos transmitida era conhecida no receptor. Além disso, foi realizado um estudo comparativo dos algoritmos de treinamento das redes. Neste âmbito, foi obtido um algoritmo do tipo acelerador para o treinamento de redes MLP. Com o intuito de se obter uma estrutura não-linear menos complexa e mais flexível, foi proposto ainda um equalizador híbrido constituído de uma combinação do equalizador linear e da rede RNN que faz uso de realimentação de decisões. Resultados de simulações indicam que o seu uso pode ser vantajoso tanto para canais não-lineares como lineares.

Abstract

Equalization of communication channels using neural networks is investigated by considering three kinds of networks: MLP (Multilayer Perceptron), RBF (Radial Basis Function) and RNN (Recurrent Neural Network). The performance of the nonlinear equalizers based on these networks are compared with the linear transversal equalizer and the optimal equalizers given by the bayesian and maximum likelihood criteria. Binary and quaternary alphabets are used and transmitted over finite pulse response channel models. Decision feedback is considered whenever it is worthwhile. The training of these equalizers is considered in the supervised form and a comparison of some training algorithms has been performed. In this scope, a new algorithm based on parameter acceleration is introduced for the training of MLP networks. Moreover, a hybrid equalizer composed of a linear transversal equalizer and a RNN network is proposed. It is a simple and flexible nonlinear structure making use of decision feedback. Simulation results show that it may be advantageously used to equalize linear and nonlinear channels.

Sumário

Lista de abreviaturas	vii
Lista de símbolos	ix
1 Introdução	1
1.1 A equalização adaptativa	1
1.2 Tópicos abordados nesta dissertação	5
1.3 Resumo dos capítulos seguintes	7
2 Descrição das estruturas utilizadas	9
2.1 Esquema de equalização utilizado	9
2.2 Equalizador Linear Transversal (LTE)	9
2.2.1 Algoritmo LMS	11
2.2.2 Algoritmo RLS	11
2.2.3 Algoritmo EF-LSL modificado	12
2.2.4 Algoritmo Acelerador para filtragem linear	14
2.3 A rede MLP (<i>Multilayer perceptron</i>)	14
2.3.1 Algoritmo BP	17
2.3.2 Algoritmo RLS-I	23
2.3.3 Algoritmo RLS-II	26
2.3.4 Algoritmo Acelerador (AC)	30
2.4 A rede RBF (<i>Radial Basis Function</i>)	35
2.4.1 O treinamento da rede RBF	37

2.5	A rede neural recorrente (RNN)	40
2.5.1	Algoritmo RTRL	42
2.6	Equalizadores ótimos	46
2.6.1	Critério de Bayes	46
2.6.2	Critério da máxima verossimilhança	52
2.7	<i>Decision Feedback Equalizer</i> (DFE)	55
2.7.1	DFE visto como classificador	57
2.8	Equalização e classificação	62
2.9	Canais não-lineares	67
3	Resultados Experimentais	69
3.1	Canais utilizados nas simulações	70
3.2	Velocidade de convergência dos algoritmos de treinamento	72
3.2.1	Redes MLP	72
3.2.2	Redes RBF	77
3.2.3	Redes recorrentes	80
3.3	Comparação de desempenho dos equalizadores	81
3.3.1	Resultados das simulações com 2-PAM	82
3.3.2	Resultados das simulações com 4-QAM	96
4	Equalizadores Híbridos	103
4.1	Equalizador híbrido proposto	105
4.2	Resultados experimentais com o equalizador híbrido	110
4.2.1	Canais usados nas simulações	110
4.2.2	Resultados das simulações com 2-PAM	112
4.2.3	Resultados das simulações com 4-QAM	116
4.3	Conclusões a respeito do equalizador híbrido	119
5	Conclusões	120
	Referências Bibliográficas	124

Apêndices	130
A Lema de inversão de matrizes	130
B Algoritmo RLS-I	132
C Algoritmo Acelerador Real	137
C.1 O algoritmo Acelerador real de tempo contínuo modificado	137
C.2 O algoritmo Acelerador real para treinamento de redes MLP	138
C.2.1 Cálculo progressivo	138
C.2.2 Cálculo regressivo	141
C.3 Observações sobre convergência e inicialização	143
D Algoritmo Acelerador Complexo	144
D.1 O algoritmo Acelerador complexo de tempo contínuo modificado	144
D.2 O algoritmo Acelerador complexo para treinamento de redes MLP . . .	145
D.2.1 Cálculo progressivo	145
D.2.2 Cálculo regressivo	150
D.3 O algoritmo Acelerador complexo não-matricial	151
E Algoritmo RTRL	157
E.1 O algoritmo RTRL real	157
E.2 O algoritmo CRTRL	160

Lista de abreviaturas

AC	Acelerador
AV	Algoritmo de Viterbi
BP	Backpropagation (Retropropagação)
CAC	Acelerador Complexo
CBP	Complex Backpropagation (Retropropagação complexa)
CDMA	Code Division Multiple Access
CRTRL	Complex Real-Time Recurrent Learning
DF	Decision Feedback (Realimentação de decisões)
DFE	Decision Feedback Equalizer
EF-LSL	Error Feedback Least Squares Lattice
FIR	Finite Impulse Response (Resposta ao pulso unitário finita)
ISI	Intersymbol Interference (Interferência intersimbólica)
LMS	Least Mean Square
LTE	Linear Transversal Equalizer (Equalizador linear transversal)
LSL	Least Squares Lattice
MAP	Maximum a Posteriori Probability
ML	Maximum Likelihood (Máxima verossimilhança)
MLP	Multilayer Perceptron
MLSE	Maximum Likelihood Sequence Estimator
MSE	Mean Square Error (Erro quadrático médio)
NL	Operação não-linear

PAM	Pulse Amplitude Modulation
PSK	Phase Shift Keying
QAM	Quadrature Amplitude Modulation
RBF	Radial Basis Function
RLS	Recursive Least Squares
RNN	Recurrent Neural Network (Rede neural recorrente)
RTRL	Real Time Recurrent Learning
SER	Symbol Error Rate (Taxa de erro de símbolo)
SNR	Signal to Noise Ratio (Relação sinal-ruído)
UCL	Unsupervised Clustering Learning

Lista de símbolos

Símbolos gerais

n	índice de tempo
$H(z)$	função de transferência do canal
$H_l(z)$	parte linear do canal descrita por função de transferência
h_k	k-ésima amostra da resposta ao pulso unitário do canal
D_k	ganho do canal não-linear na harmônica k
a	símbolo transmitido
$\hat{a}$	estimativa do símbolo transmitido
x	sinal de saída do canal
η	ruído branco gaussiano
u	sinal de entrada do receptor
y	sinal de saída do equalizador
M	número de entradas do equalizador
M_f	número de entradas do equalizador com DF referentes à saída do canal
M_b	número de entradas do equalizador com DF referentes à realimentação de decisões
M_r	número de entradas referentes à realimentação de decisões na RNN do equalizador híbrido
N_h	número de coeficientes do canal $H(z)$

τ_d	atraso em número de amostras
$\mathbf{w}$	vetor de pesos
$\mathbf{w}^H$	hermitiano (transposto do complexo conjugado) do vetor $\mathbf{w}$
$\mathbf{w}^*$	complexo conjugado do vetor $\mathbf{w}$
$\mathbf{w}^T$	transposto do vetor $\mathbf{w}$
$f_d(\cdot)$	função de decisão
$\text{sgn}(\cdot)$	função sinal
μ	passo de aprendizagem
β	<i>momentum</i> constante
d	sinal desejado
$\varphi(\cdot)$	função de ativação do neurônio
$\phi(\cdot)$	função de ativação no domínio complexo
$\mathbf{I}$	matriz identidade
$\mathbf{R}$	matriz de autocorrelação
$\mathbf{S}_w$	matriz de dispersão intra-classes
$E[\cdot]$	operador esperança
$\xi(\cdot)$	função custo

Símbolos usados no algoritmo RLS

Q	medida de desempenho
$\mathbf{g}$	ganho de Kalman
$\mathbf{P}$	inversa da matriz de autocorrelação
λ	fator de esquecimento
δ	constante positiva pequena

Símbolos usados no algoritmo EF-LSL

m	índice de ordem
ξ_i^f	energia da predição progressiva
ξ_i^b	energia da predição regressiva
η_i	erro da predição progressiva
ψ_i	erro da predição regressiva
T_i^f	coeficiente de reflexão da predição progressiva
T_i^b	coeficiente de reflexão da predição regressiva
γ_i	fator de conversão
α_i	erro de estimação <i>a priori</i>
T_i^d	coeficiente de regressão

Símbolos usados no algoritmo Acelerador

m_1, m_2, m_3	parâmetros de aceleração
$\mathbf{q}$	derivada do vetor de pesos
e_a	erro de estimação <i>a priori</i>
γ	escalar usado na versão não matricial

Símbolos usados na rede MLP

$v_j^{(l)}$	nível de atividade interna do neurônio j da camada l
y_j^l	saída do neurônio j da camada l ; j -ésima entrada da camada $l + 1$
$y_j^{(0)}$	j -ésima entrada da rede ($y_j^{(0)} = u_j$)
$\theta_j^{(l)}$	nível de ajuste (<i>bias</i>) do neurônio j da camada l

$w_{km}^{(l)}$	peso que liga o neurônio k da camada l ao neurônio m da camada $l - 1$
$e_k^{(l)}$	erro do neurônio k da camada l
$\delta_j^{(l)}$	gradiente local do neurônio j da camada l
N_l	número de neurônios da camada l
N_0	número de entradas da rede $N_0 = M$
L	última camada da rede

Símbolos usados na rede RBF

$\mathbf{r}_k$	centro da gaussiana do neurônio k
σ_k^2	variância da gaussiana do neurônio k
s_k	distância entre o sinal de entrada e o centro da gaussiana do neurônio k
$f(s_k)$	saída do neurônio k

Símbolos usados no Algoritmo RTRL

Π	sensibilidade no domínio real
$P_{RR}, P_{RI}, P_{IR}, P_{II}$	sensibilidades no domínio complexo

Símbolos usados no equalizador bayesiano

$p(\cdot)$	densidade de probabilidade
$P(\cdot)$	probabilidade
$\mathbf{H}$	matriz de Toeplitz do canal

N_x número de estados desejados do canal

Símbolos usados no equalizador de ML

$W(z^{-1})$ filtro casado branqueador

s saída do filtro casado branqueador

$R(z)$ função de autocorrelação

$F(z)$ parcela de fase mínima da fatoração de $R(z)$

$F(z^{-1})$ parcela de fase não-mínima da fatoração de $R(z)$

Capítulo 1

Introdução

1.1 A equalização adaptativa

A equalização adaptativa é uma técnica largamente utilizada em sistemas de comunicação digital para combater a deterioração do sinal na entrada do receptor, provocada por imperfeições no canal de transmissão. Normalmente assume-se o modelo [1] mostrado na Figura 1.1 e descrito a seguir. Considera-se que uma seqüência de símbolos $a(n)$ é transmitida através de um canal que fornece na sua saída uma versão distorcida da mesma. O sinal recebido $u(n) = x(n) + \eta(n)$ consiste de duas parcelas: o sinal de saída do canal $x(n)$ e o ruído $\eta(n)$. Este sinal é aplicado ao equalizador que deve fornecer uma estimativa da seqüência de símbolos originalmente transmitida. O bloco “canal” da Figura 1.1 representa não só o canal físico de transmissão mas também o sistema de transmissão/modulação e o sistema de recepção/demodulação efetivamente presentes em qualquer sistema de comunicações prático. Assim, denomina-se aqui como canal um modelo de tempo discreto para o sistema de transmissão, o canal físico e o sistema de recepção.

Na maioria das aplicações, as distorções introduzidas por um canal são suficientemente bem modeladas por uma resposta ao pulso unitário finita, caso em que a função de transferência é dada por $H(z) = \sum_{k=0}^{N_h-1} h_k z^{-k}$, sendo N_h o comprimento da resposta

Figura 1.1: Modelo de um sistema de comunicações de tempo discreto.

ao pulso unitário do canal e h_k os coeficientes que caracterizam o mesmo. Considerando inicialmente um modelo deste tipo, existem os seguintes aspectos que devem ser considerados em um problema de equalização:

- A interferência intersimbólica (ISI) causada pelo efeito dispersivo do canal. Considerando o modelo linear apresentado, a ISI é o resultado da convolução do sinal transmitido pelo canal com a sua resposta ao pulso unitário. Este efeito dispersivo pode também ser originário da existência de diversos percursos de transmissão percorridos pelo sinal. Neste caso, fala-se de um efeito de multipercurso. Em comunicações sem fio, o sinal pode ser recebido por um percurso direto entre as antenas de transmissão e recepção e também por reflexões de outros objetos, como morros, prédios, etc. Neste caso, o canal de transmissão pode ser representado por vários canais em paralelo, gerando uma variação do modelo anterior como pode ser visto em [14]. Além disso, obtém-se muitas vezes do processo de modelagem canais que não são de mínima fase, o que dificulta a equalização. Deve-se considerar também canais que são variantes no tempo, como é o caso de canais de telefonia móvel que apresentam o efeito de desvanecimento (*fading*).
- A presença de ruído na entrada do receptor que vai afetar o desempenho do equalizador;

Considerando inicialmente a equalização de um canal fixo, não-variante no tempo, costumam-se utilizar diferentes critérios para obter equalizadores ótimos:

- Critérios que minimizam momentos de 2^a ordem como o da minimização do erro quadrático médio ou o critério dos mínimos quadrados;
- Critérios originários da teoria da informação como o da máxima verossimilhança ou da máxima entropia;
- Critério de Bayes (para maximização da probabilidade *a posteriori* de símbolo transmitido dado o sinal recebido).

O primeiro tipo de critério leva a soluções para o equalizador que consistem de um filtro transversal linear (LTE - *Linear Transversal Equalizer*) seguido de um elemento não-linear de decisão. Considerando o problema de equalização como um problema de classificação de diferentes saídas do canal, isto é, de diferentes estados, observa-se que este tipo de solução é linear pois consiste na divisão do espaço de estados por hiperplanos. Aqui já se observa uma limitação fundamental deste tipo de equalizador pois quando o canal não é de mínima fase, o equalizador ótimo segundo o critério de Bayes é não-linear no sentido que as hipersuperfícies que dividem o espaço de estados de forma ótima não são simples hiperplanos. Desta forma, considerando a equalização como um problema de classificação, o equalizador linear é sub-ótimo.

Em relação ao critério da máxima verossimilhança, que pertence ao conjunto de critérios citados em segundo lugar, Forney [23] demonstrou que o equalizador ótimo é formado por um filtro casado com o canal de transmissão seguido do algoritmo de Viterbi [24]. O filtro casado deve ser também branqueador para que o ruído do sinal aplicado ao algoritmo de Viterbi seja não-correlacionado. Este equalizador é não-linear e busca a sequência de máxima verossimilhança. Uma variação deste esquema foi apresentada por Ungerboeck [25]. Neste trabalho, foi feita uma modificação no algoritmo de Viterbi o que permitiu que o ruído fosse correlacionado. Para se projetar o equalizador ótimo apresentado em [23], o conhecimento prévio do canal se faz necessário. Em [26] foi utilizado um algoritmo adaptativo para estimação do canal e se observou que o resultado da equalização se afastava um pouco do ótimo. Além disso, o algoritmo

de Viterbi apresenta uma complexidade computacional que cresce exponencialmente com a ordem do modelo do canal, o que pode ser um problema na prática.

Dependendo do canal, o equalizador linear transversal (LTE), bastante conhecido e utilizado, poderá estar muito longe do ótimo segundo o critério de Bayes ou do ótimo segundo o critério da máxima verossimilhança. Por este motivo, surgiram variações do mesmo, como o *Decision Feedback Equalizer* (DFE) que pode proporcionar melhoras significativas nos resultados da equalização. O DFE nada mais é do que um LTE em que são também utilizadas decisões passadas. Isso faz com este tipo de sistema tenha uma estrutura onde a não-linearidade do decisor está posicionada em uma malha de realimentação constituindo um sistema não-linear realimentado (Figura 1.2).

Figura 1.2: *Decision Feedback Equalizer* (DFE).

Diante disso, surgiram vários estudos na direção de se obter um equalizador intermediário entre o ótimo e o LTE. Várias estruturas não-lineares têm sido estudadas, como as baseadas em séries de Volterra e as que utilizam redes neurais. As redes neurais, apesar de serem computacionalmente complexas, apresentam vantagens sobre outras estruturas não-lineares devido ao seu alto grau de paralelismo o que as tornam atraentes para implementação em circuitos integrados [45].

Entretanto, não se deve esquecer que a grande vantagem do LTE / DFE é a sua simplicidade que permite o desenvolvimento de algoritmos de baixa complexidade para a adaptação em tempo real dos coeficientes. A variação dos parâmetros do canal com o tempo ou a utilização de um mesmo dispositivo equalizador com diferentes canais leva à necessidade de se ajustar os coeficientes do equalizador de forma automática. Neste

contexto, o desenvolvimento de equalizadores ótimos segundo os critérios de Bayes ou da máxima verossimilhança que sejam menos complexos é um problema de interesse. Da mesma forma, o desenvolvimento de algoritmos eficientes de ajuste ou treinamento para estes equalizadores é um problema que também tem a sua importância.

O treinamento de equalizadores pode ser feito de forma supervisionada ou autodidata. No treinamento supervisionado, o sinal transmitido é conhecido no receptor pelo menos em uma fase inicial que é chamada de fase de treinamento. No treinamento autodidata, faz-se uma estimativa do sinal transmitido baseando-se em estatísticas do sinal de saída do canal e de decisões já tomadas. Em sistemas que utilizam comunicações móveis como sistemas CDMA (*Code Division Multiple Access*) é desejável ajustar os parâmetros do equalizador sem a necessidade de se conhecer a sequência transmitida, proporcionando dessa forma um uso mais apropriado da banda de transmissão [49].

1.2 Tópicos abordados nesta dissertação

Nesta dissertação de mestrado foi investigado o uso de redes neurais aplicadas à equalização de canais de comunicação. Foram utilizadas três tipos de redes: MLP (*Multilayer Perceptron*), RBF (*Radial Basis Function*) e RNN (*Recurrent Neural Network*). Os desempenhos dos equalizadores não-lineares baseados nestas redes foram comparados com o equalizador linear transversal (LTE) e com os equalizadores ótimos segundo o critério de Bayes e o critério da máxima verossimilhança. O uso de realimentação de decisões também foi explorado nestas estruturas. O treinamento desses equalizadores foi feito de forma supervisionada, ou seja, na fase de treinamento a sequência de símbolos transmitida era conhecida no receptor. Os modelos dos canais utilizados englobam um modelo de tempo discreto para o sistema de transmissão, o canal físico e o sistema de recepção. Além disso, foram considerados canais do tipo FIR, ou seja, canais cuja resposta ao pulso unitário é finita.

De modo geral um dos grandes problemas no uso de redes neurais do tipo MLP

para equalização está na utilização de algoritmos de treinamento baseados no gradiente estocástico. Embora esses algoritmos representem uma poderosa ferramenta para treinamento de redes do tipo MLP, eles apresentam uma baixa velocidade de convergência. O algoritmo mais conhecido para o treinamento de redes multi-camada é o *Backpropagation* ou algoritmo de retropropagação.

O objetivo desses algoritmos é minimizar uma função custo, atualizando os fatores de ponderação (pesos) na direção do erro mínimo. No caso de um combinador linear, a superfície de erro no espaço de parâmetros apresenta somente um mínimo. Este não é o caso do problema em questão onde podem ocorrer mínimos locais para onde o algoritmo de retropropagação pode convergir e ficar “encalhado”. Além disso, segundo Hecht-Nielsen [50], um outro problema associado à retropropagação é a existência de regiões na superfície de erro onde o gradiente é pequeno. Considerando uma taxa de aprendizado fixa, como é o caso, isto significará necessariamente uma convergência lenta.

Em aplicações de filtragem adaptativa, o algoritmo RLS é tipicamente uma ordem de magnitude mais rápido que o algoritmo LMS [1]. Lembrando que o algoritmo de retropropagação pode ser entendido como uma generalização do LMS para o caso de redes não-lineares multi-camada, chega-se de imediato à idéia de desenvolver algoritmos de treinamento usando o princípio da retropropagação mas baseando-se no algoritmo RLS ou similares, como o algoritmo de Kalman estendido ou ainda o algoritmo acelerador [5]. Pode-se, assim, esperar um ganho na velocidade de treinamento da mesma ordem de grandeza do ganho na rapidez de convergência entre os algoritmos LMS e RLS. O preço pago por este ganho é sem dúvida um aumento na complexidade computacional do algoritmo de treinamento em comparação com o de retropropagação convencional.

Diante disso, foram estudados dois algoritmos baseados no método dos mínimos quadrados recursivo para o treinamento de redes MLP. Foi também desenvolvido um algoritmo para treinamento de redes MLP baseado no algoritmo Acelerador [5], [6],

[7], utilizado em filtragem adaptativa linear. Este algoritmo também foi utilizado no treinamento de redes RBF. Ele apresenta algumas vantagens em comparação com os algoritmos de treinamento convencionais devido à maior velocidade de convergência às custas de um moderado aumento na complexidade computacional. Este aumento é inferior ao resultante do uso de algoritmos de treinamento baseados no algoritmo RLS.

Com relação à rede do tipo RBF, um dos grandes problemas associados ao seu uso para equalização está na correta determinação dos centros e dos raios das funções de base radial quando não se conhece o modelo do canal. Técnicas baseadas em análise de agrupamentos (*cluster analysis*) são importantes na determinação destes centros, possibilitando a redução do número de neurônios e o uso de uma RBF em tempo real [15].

A fim de possibilitar o uso dessas estruturas para sinais complexos, como é o caso das modulações QAM, PSK, etc., foram utilizadas versões complexas dos algoritmos no treinamento das redes consideradas.

O estudo de todas estas estruturas possibilitou ainda o projeto de um equalizador híbrido, no qual um equalizador linear é utilizado em conjunto com uma estrutura não-linear. Com isso, procura-se atingir um bom desempenho com uma estrutura não-linear de menor complexidade computacional.

Cabe observar que os equalizadores fracionários não foram considerados nesta dissertação. Às custas de um aumento da taxa de amostragem, estes equalizadores apresentam em muitas situações um desempenho superior aos dos convencionais conseguindo compensar distorções provocadas por canais mais severos [53]. A investigação de seu uso em conjunto com estruturas não-lineares foi reservada para trabalhos futuros.

1.3 Resumo dos capítulos seguintes

No Capítulo 2, é feita uma descrição detalhada das estruturas utilizadas e seus algoritmos de treinamento. As deduções dos algoritmos mais conhecidos foram omitidas,

sendo apresentadas referências onde podem ser encontradas. Por outro lado, as deduções dos menos conhecidos estão nos apêndices. Foi feita ainda uma descrição dos equalizadores ótimos segundo o critério de Bayes e o critério da máxima verossimilhança. Interpretando os equalizadores como classificadores, foi feita uma comparação dos resultados da classificação do LTE, da MLP e da RBF com o classificador ótimo segundo o critério de Bayes.

No Capítulo 3, são mostrados os resultados das simulações. Inicialmente foi feita uma comparação das velocidades de convergência dos algoritmos utilizados no treinamento da rede MLP. A importância da utilização de algoritmos para adaptação dos centros e variâncias das funções de base radial da rede RBF pôde ser verificada. Uma comparação de desempenho das estruturas utilizadas com os equalizadores ótimos foi feita através de curvas de taxa de erro de símbolo (SER), considerando três canais para o caso real e três para o caso complexo.

No Capítulo 4, um equalizador híbrido é proposto e em seguida são apresentados os resultados das simulações considerando os casos real e complexo, canais com poucos coeficientes e também um canal mais realista.

No Capítulo 5, são apresentadas as conclusões do trabalho.

Capítulo 2

Descrição das estruturas utilizadas

2.1 Esquema de equalização utilizado

As estruturas estudadas nesta dissertação seguem o esquema básico de equalização da Figura 2.1, sendo $H(z)$ a transformada-Z da resposta ao pulso unitário finita (FIR) do canal e η o ruído branco gaussiano adicionado à sequência $x(n)$ para obter a sequência de observação $u(n)$. O problema a ser considerado é usar no instante n a informação das amostras $u(n), u(n-1), \dots, u(n-M+1)$, que podem ser agrupadas no vetor $\mathbf{u}(n) = [u(n) \ u(n-1) \ \dots \ u(n-M+1)]^T$, para estimar o símbolo transmitido $a(n-\tau_d)$. Este símbolo é usado como sinal de referência sendo então comparado com o sinal de saída do equalizador $y(n)$ na fase de treinamento, obtendo-se, dessa forma, o sinal de erro $e(n)$, que por sua vez é responsável pela atualização dos parâmetros do algoritmo de treinamento na busca do erro mínimo global da função custo.

Os inteiros M e τ_d na Figura 2.1 representam o número de entradas e o atraso do equalizador em número de amostras respectivamente.

2.2 Equalizador Linear Transversal (LTE)

O problema de equalização tratado com o LTE se resume em usar um algoritmo adaptativo tipo gradiente estocástico como o LMS (*Least Mean Square*) ou um algoritmo

Figura 2.1: Esquema básico empregado para equalização adaptativa.

dos mínimos quadrados recursivo, como o RLS (*Recursive Least Squares*) ou ainda algumas variações deste último, como algoritmos da família LSL (*Least Squares Lattice*), para obter uma estimativa do símbolo transmitido $a(n - \tau_d)$ [1]. Nos algoritmos LMS e RLS esse problema se resume a encontrar os pesos w_k e estimar $a(n - \tau_d)$ como

$$\hat{a}(n - \tau_d) = f_d \left(\sum_{k=0}^{M-1} w_k^* u(n - k) \right) = f_d(\mathbf{w}^H(n) \mathbf{u}(n)), \quad (2.1)$$

sendo

$\mathbf{w}(n) = [w_0(n) \ w_1(n) \ \cdots \ w_{M-1}(n)]^T$, $\mathbf{u}(n) = [u(n) \ u(n-1) \ \cdots \ u(n-M+1)]^T$ e $f_d(\cdot)$ a função de decisão. No caso binário $f_d(\cdot) = \text{sgn}(\cdot)$, com

$$\text{sgn}(y) = \begin{cases} -1 & y \leq 0 \\ 1 & y > 0. \end{cases} \quad (2.2)$$

Por outro lado, em alguns algoritmos, os pesos do filtro transversal não são utilizados diretamente como é o caso do algoritmo EF-LSL. Neste caso, a estimativa $\hat{a}(n - \tau_d)$ é obtida de forma diferente como será mostrado posteriormente.

Dentre os diversos algoritmos de filtragem adaptativa linear que existem na literatura foram considerados neste trabalho os seguintes: LMS, RLS, EF-LSL modificado e o algoritmo Acelerador. As deduções dos três primeiros podem ser encontradas em

[1], as modificações introduzidas no algoritmo EF-LSL em [2] e a dedução do algoritmo Acelerador em [5] e [6]. Tabelas dos mesmos e números de operações por iteração estão mostrados nas sub-seções seguintes.

2.2.1 Algoritmo LMS

O LMS (*Least Mean Square*) é um algoritmo do tipo gradiente estocástico que possui uma baixa complexidade computacional quando comparado com outros algoritmos [1]. Considerando sinais reais, ele possui $2M+1$ multiplicações e $2M$ somas a cada iteração. Por outro lado, possui uma velocidade de convergência também baixa. Suas operações estão mostradas na Tabela 2.1 sendo μ o passo de aprendizagem do algoritmo e d o sinal desejado. No caso da equalização supervisionada, este sinal corresponde ao símbolo transmitido com um atraso de τ_d amostras, ou seja $d(n) = a(n - \tau_d)$.

Inicialização do algoritmo:

$$\mathbf{w}(0) = \mathbf{0}$$

Para cada instante de tempo, $n = 1, 2, \dots$, calcule:

$$y(n) = \mathbf{w}^H(n)\mathbf{u}(n)$$

$$e(n) = d(n) - y(n)$$

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu\mathbf{u}(n)e^*(n)$$

Tabela 2.1: Algoritmo LMS.

2.2.2 Algoritmo RLS

O RLS (Recursive Least Squares) é um algoritmo baseado no método dos mínimos quadrados recursivo [1]. Considerando sinais reais, ele possui complexidade computacional de $5M^2 + 4M + 1$ multiplicações, 1 divisão e $4M^2$ somas a cada iteração. Em comparação com o LMS, embora seja bem mais complexo computacionalmente, pode apresentar uma velocidade de convergência bem superior quando o sinal de entrada é

mal condicionado. Suas operações estão mostradas na Tabela 2.2, sendo λ o fator de esquecimento, $\mathbf{P}$ a inversa da matriz de autocorrelação, $\mathbf{g}$ o ganho de Kalman e d o sinal desejado.

Inicialização do algoritmo:

$$\mathbf{P}(0) = \delta^{-1} \mathbf{I}, \delta = \text{constante positiva pequena}$$

$$\mathbf{w}(0) = \mathbf{0}$$

Para cada instante de tempo, $n = 1, 2, \dots$, calcule:

$$\mathbf{g}(n) = \frac{\lambda^{-1} \mathbf{P}(n-1) \mathbf{u}(n)}{1 + \lambda^{-1} \mathbf{u}^H(n) \mathbf{P}(n-1) \mathbf{u}(n)}$$

$$e(n) = d(n) - \mathbf{w}^H(n-1) \mathbf{u}(n)$$

$$\mathbf{w}(n) = \mathbf{w}(n-1) + \mathbf{g}(n) e^*(n)$$

$$\mathbf{P}(n) = \lambda^{-1} \mathbf{P}(n-1) - \lambda^{-1} \mathbf{g}(n) \mathbf{u}^H(n) \mathbf{P}(n-1)$$

Tabela 2.2: Algoritmo RLS.

2.2.3 Algoritmo EF-LSL modificado

O LSL *a priori* com realimentação de erro (EF-LSL) [2] é apresentado na Tabela 2.3, sendo que $(\xi_i^f(n), \eta_i(n), T_i^f(n))$ e $(\xi_i^b(n), \psi_i(n), T_i^b(n))$ representam as energias, os erros de predição e os coeficientes de reflexão das predições progressiva e regressiva respectivamente. As variáveis $(\gamma_i(n), \alpha_i(n), T_i^d(n))$ são respectivamente o fator de conversão, o erro de estimação *a priori* e o coeficiente de regressão; δ é uma constante positiva pequena, λ é o fator de ponderação (ou esquecimento) e $(b_i(n), \bar{b}_i(n), f_i(n), \bar{f}_i(n))$ são variáveis intermediárias introduzidas em [2] para diminuir a complexidade computacional do algoritmo original desenvolvido por Ling *et al.* em [3]. Deve-se notar ainda que na Tabela 2.3, as amostras do sinal de entrada consideradas no instante n são $u(n), u(n-1), \dots, u(n-M+1)$ e $d(n)$ é o sinal desejado.

Inicialização ($n = 0$): De $m = 1$ até M faça $\{x_{m-1}^f(n) = \delta; x_{m-1}^b(n) = \lambda\delta; \Gamma_m^f(n+1) =$ $= \Gamma_m^b(n+1) = \Gamma_m^d(n) = \psi_{m-1}(n+1) = 0\}$
Para $n = 1, 2, 3, \dots$ faça: Para $m = 0$ faça: $\{\gamma_m(n) = 1; \eta_m(n+1) = u(n+1);$ $\psi_m(n) = u(n); \alpha_m(n) = d(n)\}$ De $m = 1$ até M faça { A.1 a C.5 }, em seguida faça { D.1 }
A.1) $b_{m-1}(n) = \gamma_{m-1}(n)\psi_{m-1}(n)$ A.2) $\mathcal{B}_{m-1}(n) = x_{m-1}^b(n-1) + b_{m-1}(n)\psi_{m-1}(n)$ A.3) $\bar{b}_{m-1}(n) = b_{m-1}(n)/\mathcal{B}_{m-1}(n)$ A.4) $x_{m-1}^b(n) = \lambda\mathcal{B}_{m-1}(n)$ A.5) $\Gamma_m^f(n+1) = \Gamma_m^f(n) + \bar{b}_{m-1}(n)\eta_m(n+1)$
B.1) $\psi_m(n+1) = \psi_{m-1}(n) - \Gamma_m^b(n)\eta_{m-1}(n+1)$ B.2) $\eta_m(n+1) = \eta_{m-1}(n+1) - \Gamma_m^f(n)\psi_{m-1}(n)$ B.3) $\alpha_m(n) = \alpha_{m-1}(n) - \psi_{m-1}(n)\kappa_m^d(n-1)$ B.4) $\gamma_m(n) = \gamma_{m-1}(n) - \bar{b}_{m-1}(n)b_{m-1}(n)$ B.5) $\kappa_m^d(n) = \kappa_m^d(n-1) + \bar{b}_{m-1}(n)\alpha_m(n)$
C.1) $f_{m-1}(n+1) = \gamma_{m-1}(n)\eta_{m-1}(n+1)$ C.2) $\mathcal{F}_{m-1}(n+1) = x_{m-1}^f(n) + f_{m-1}(n+1)\eta_{m-1}(n+1)$ C.3) $\bar{f}_{m-1}(n+1) = f_{m-1}(n+1)/\mathcal{F}_{m-1}(n+1)$ C.4) $x_{m-1}^f(n+1) = \lambda\mathcal{F}_{m-1}(n+1)$ C.5) $\Gamma_m^b(n+1) = \Gamma_m^b(n) + \bar{f}_{m-1}(n+1)\psi_m(n+1)$
D.1) $\hat{a}(n - \tau_d) = f_d(d(n) - \alpha_M(n))$

Tabela 2.3: Algoritmo EF-LSL modificado.

Este algoritmo é interessante porque dentre todos os algoritmos RLS rápidos recursivos na ordem para os quais a estabilidade numérica não foi demonstrada, é o que apresenta o melhor desempenho numérico. Por meio de simulações, pode ser verificado que apresenta um comportamento estável e resultados precisos em precisão finita para um grande número de situações práticas, apesar da inexistência de uma prova formal de sua estabilidade numérica [2]. O algoritmo apresentado na Tabela 2.3 possui complexidade computacional de $13M$ multiplicações, $2M$ divisões e $9M$ somas a cada iteração, considerando sinais reais.

2.2.4 Algoritmo Acelerador para filtragem linear

O algoritmo Acelerador foi obtido a partir da discretização por três formas distintas de um algoritmo de tempo contínuo utilizado em controle adaptativo [6], [4]. Como pode ser visto em [5] e [6] a discretização feita pelo método de Euler reverso apresentou o melhor desempenho levando em conta a complexidade computacional e a estabilidade.

Às custas de um moderado aumento da complexidade computacional, este algoritmo apresenta uma velocidade de convergência maior que a do LMS. Suas operações necessárias por iteração são mostradas na Tabela 2.4 sendo μ o passo de aprendizagem, m_1 , m_2 e m_3 parâmetros de aceleração que devem ser escolhidos a fim de satisfazer $4m_1^2m_2m_3 \geq 1$ com $4m_1^2m_2m_3 \approx 1$. Nesta condição, obtida a partir do critério de estabilidade de Liapunov, o algoritmo apresenta a maior velocidade de convergência segundo [6] e [7]. Considerando sinais reais, a complexidade computacional deste algoritmo é de $6M + 3$ multiplicações, $5M$ somas e 1 divisão por iteração.

2.3 A rede MLP (*Multilayer perceptron*)

Nesta seção será feita uma rápida descrição da rede neural do tipo MLP e de quatro algoritmos de treinamento: *Backpropagation* (BP) que é baseado no algoritmo do gradiente estocástico, dois outros baseados no método dos mínimos quadrados recursivo e

Inicialização do algoritmo:

$$\mathbf{w}(0) = \mathbf{0}$$

$$\alpha = 1 + 2\mu m_1 m_2$$

$$b = 2\mu m_1^2 m_3 + m_1 \mu^2$$

Para cada instante de tempo, $n = 1, 2, \dots$, calcule:

$$e_a(n) = \mathbf{w}^H(n-1)\mathbf{u}(n) - d(n)$$

$$\gamma(n) = \frac{b\mathbf{u}^H(n)\mathbf{q}(n-1) + \mu m_1 \alpha e_a^*(n)}{\alpha + b\mathbf{u}^H(n)\mathbf{u}(n)}$$

$$\mathbf{q}(n) = \frac{1}{\alpha} [\mathbf{q}(n-1) - \gamma(n)\mathbf{u}^H(n)]$$

$$\mathbf{w}(n) = \mathbf{w}(n-1) + \mu \mathbf{q}(n)$$

Tabela 2.4: Algoritmo Acelerador para filtragem linear.

finalmente um baseado no algoritmo Acelerador.

Tipicamente uma rede MLP consiste de um conjunto de nós (neurônios) que estão dispostos em camadas, mais especificamente em uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída [8]. O sinal de entrada se propaga pela rede progressivamente, camada por camada. A estrutura de uma rede MLP é mostrada na Figura 2.2. Em cada nó, é calculada a soma ponderada das entradas adicionadas a um nível de ajuste (*bias*). Este resultado passa então por uma função não-linear $\varphi(\cdot)$ chamada de função de ativação. Como pode ser visto ainda na Figura 2.2, todos os nós de uma camada são conectados aos nós da camada seguinte, mas não há conexão entre os nós pertencentes à mesma camada.

Na literatura, é comum chamar os nós de neurônios, a função de ativação de função neuronal e a ligação entre nós de sinapses.

Uma rede MLP tem três características que merecem ser mencionadas:

- O modelo de cada neurônio da rede inclui uma não-linearidade na saída. O ponto importante a ser enfatizado é que é conveniente que a não-linearidade seja uma função diferenciável em todos os pontos. Normalmente se usa a função sigmoidal

Figura 2.2: Arquitetura de uma rede MLP com duas camadas ocultas e com destaque do nó k da camada l .

definida como

$$y_k = \varphi(v_k) = \frac{1 - e^{-cv_k}}{1 + e^{-cv_k}}, \quad (2.3)$$

sendo v_k o nível de atividade interna do neurônio k , y_k a saída do mesmo e c uma constante inteira positiva, normalmente tomada igual a 1 ou 2. Esta função está esquematizada na Figura 2.3;

- A rede contém uma ou mais camadas ocultas que estão diretamente conectadas às entradas ou às saídas. Esses neurônios ocultos habilitam a rede a aprender tarefas complexas extraindo aspectos mais significativos dos padrões de entrada;
- A rede exibe um alto grau de conectividade pois cada neurônio de uma camada está completamente conectado aos neurônios das camadas adjacentes. Uma mu-

dança na conectividade da rede requer uma mudança de suas conexões sinápticas ou de seus pesos.

Figura 2.3: Função sigmoidal e sua derivada com $c = 1$ e $c = 2$.

2.3.1 Algoritmo BP

O algoritmo de retropropagação (*Backpropagation*) se baseia na minimização de uma função custo definida pela equação

$$\xi(n) = \frac{1}{2} \sum_{k=1}^{N_L} |e_k(n)|^2, \quad (2.4)$$

sendo N_L o número de neurônios da camada de saída da rede e $e_k(n)$ o sinal de erro calculado nesta camada na iteração n (ver Equação (2.7) a seguir).

Algoritmo *Backpropagation* Real

O algoritmo *Backpropagation* para sinais reais pode ser resumido pelos seguintes passos:

1. Inicialização

Os pesos e os níveis de ajuste (*bias*) devem ser inicializados com números aleatórios uniformemente distribuídos num intervalo de pequenos valores como por exemplo $[-10^{-2}, 10^{-2}]$.

2. Cálculo progressivo

As saídas de cada neurônio da camada l devem ser calculadas como

$$y_k^{(l)}(n) = \varphi(v_k^{(l)}(n)), \quad (2.5)$$

com

$$v_k^{(l)}(n) = \sum_{m=0}^{N_{l-1}} w_{km}^{(l)}(n) y_m^{(l-1)}(n), \quad (2.6)$$

sendo $y_m^{(l-1)}(n)$ a saída do neurônio m da camada $l-1$ na n -ésima iteração e $w_{km}^{(l)}$ o peso sináptico que liga o neurônio k da camada l ao neurônio m da camada $l-1$. Deve-se notar que os níveis de ajuste (*bias*) estão embutidos no vetor de pesos considerando $y_0^{(l-1)}(n) = -1$ e $w_{k0}^{(l)}(n) = \theta_k^{(l)}(n)$, sendo $\theta_k^{(l)}$ o valor do nível de ajuste (*bias*) do neurônio k da camada l . A função não-linear é a que está definida na Equação (2.3).

A k -ésima entrada da rede no instante n é dada por $y_k^{(0)}(n) = u_k(n)$, sendo u_k o k -ésimo elemento da seqüência de entrada $u(n), u(n-1), \dots, u(n-M+1)$. A saída no instante n do neurônio k pertencente à camada de saída ($l=L$) é representada por $y_k^{(L)}(n)$.

Feitas estas observações, calcula-se o sinal de erro do neurônio k da camada de saída ($l=L$) como

$$e_k(n) = d_k(n) - y_k^{(L)}(n), \quad (2.7)$$

sendo $d_k(n)$ o sinal desejado para o neurônio de saída k no instante n . No problema de equalização este é o símbolo $a(n - \tau_d)$ considerando um atraso de τ_d amostras.

3. Cálculo regressivo

A derivada da função custo com relação a $w_{km}^{(l)}(n)$ pode ser escrita como

$$\frac{\partial \xi(n)}{\partial w_{km}^{(l)}(n)} = -\delta_k^{(l)}(n) y_m^{(l-1)}(n), \quad (2.8)$$

sendo que $\delta_k^{(l)}(n)$ é definido em [8] como gradiente local e deve ser calculado como explicado a seguir.

Para um neurônio k da camada de saída ($l = L$), deve-se utilizar

$$\delta_k^{(L)}(n) = e_k^{(L)}(n) \varphi'(v_k^{(L)}(n)), \quad (2.9)$$

sendo $\varphi'(\cdot)$ a derivada da função neuronal e $v_k^{(L)}$ definido como na Equação (2.6).

Para um neurônio k da camada oculta $l \neq L$, a expressão é dada por

$$\delta_k^{(l)}(n) = \varphi'(v_k^{(l)}(n)) \sum_m \delta_m^{(l+1)}(n) w_{mk}^{(l+1)}(n). \quad (2.10)$$

Com o gradiente local, atualiza-se os pesos a partir da equação

$$w_{km}^{(l)}(n+1) = w_{km}^{(l)}(n) + \beta [w_{km}^{(l)}(n) - w_{km}^{(l)}(n-1)] + \mu \delta_k^{(l)}(n) y_m^{(l-1)}(n), \quad (2.11)$$

sendo β um *momentum* constante, escolhido no intervalo $[0, 1]$ (usado para aumentar a velocidade de convergência do algoritmo) e μ o passo de aprendizagem que deve ser escolhido também no intervalo $]0, 1]$.

4. Atualizados os pesos, deve-se voltar ao passo 2 até que o erro mínimo pré-especificado seja atingido.

Detalhes da dedução deste algoritmo podem ser encontrados na referência [8].

Fazendo-se uma análise da complexidade computacional do algoritmo BP real na fase de treinamento de uma rede MLP com L camadas seguindo a configuração $(N_1, N_2, \dots, N_L)$, obtém-se os números de operações por iteração da Tabela 2.5. Nesta tabela, N_0 representa o número de entradas da rede ($N_0 = M$) e NL indica o cálculo da função não-linear $\varphi(\cdot)$ e de sua derivada.

Operações	Nº de operações por iteração
$\times$	$N_A + 5N_B$
NL	$2N_B$
$+$	$N_A + N_B + 2N_L$

Tabela 2.5: Complexidade computacional do algoritmo BP real na fase de treinamento de uma rede MLP $(N_1, N_2, \dots, N_L)$, sendo $N_A = 3N_0N_1 + 4(N_1N_2 + N_2N_3 + \dots + N_{L-1}N_L)$ e $N_B = N_1 + N_2 + \dots + N_L$.

Deve-se notar que o número de multiplicações, somas e cálculos de não-linearidades da Tabela 2.5 são da fase de treinamento. Supondo agora a rede treinada, obtém-se o número de operações da Tabela 2.6.

Operações	Nº de operações por iteração
$\times$	$N_C + 5N_B$
NL	N_B
$+$	N_C

Tabela 2.6: Complexidade computacional de de uma rede MLP $(N_1, N_2, \dots, N_L)$ treinada com sinais reais, sendo $N_C = N_0N_1 + N_1N_2 + N_2N_3 + \dots + N_{L-1}N_L$ e $N_B = N_1 + N_2 + \dots + N_L$.

Algoritmo *Backpropagation* Complexo

O algoritmo *Backpropagation* complexo é uma generalização do algoritmo LMS complexo com uma apropriada escolha da função de ativação não-linear. Uma das difi-

dades encontradas na extensão do algoritmo BP real ao domínio complexo está relacionada com a correta escolha dessa função [1]. A função de ativação não-linear para o caso real (Equação (2.3)) quando estendida a todo o plano complexo apresenta singularidades, ou seja

$$\frac{1 - e^{-v}}{1 + e^{-v}} \rightarrow \infty \text{ para } v = \pm j(2k + 1)\pi, \text{ com } k \text{ inteiro.}$$

Como a implementação prática do algoritmo BP complexo requer que a função neuronal seja limitada, o uso desta função fica prejudicada, embora alguns autores proponham a sua utilização [9].

Em [10] Benvenuto e Piazza propuseram a seguinte função de ativação

$$\phi(z) = \varphi(z_R) + j\varphi(z_I), \quad (2.12)$$

sendo z uma variável complexa $z = z_R + jz_I$ e $\varphi(\cdot)$ a função sigmoideal utilizada no domínio real (Equação (2.3)). Se por um lado esta função de ativação não apresenta singularidades, por outro ela não é uma função complexa analítica. Este fato faz com que o algoritmo *Backpropagation* complexo não seja uma simples extensão do caso real. Gráficos da parte real e do módulo dessa função estão mostrados na Figura 2.4-a e b respectivamente.

Considerando a função custo definida pela Equação (2.4) e a função de ativação $\phi(\cdot)$, o correspondente algoritmo CBP pode ser resumido pelas equações mostradas a seguir.

Cálculo Progressivo: Para $l = 1, \dots, L$ e $k = 1, \dots, N_l$, calcule:

$$v_k^{(l)}(n) = \sum_{m=0}^{N_{l-1}} w_{km}^{(l)}(n) y_m^{(l-1)}(n) \quad (2.13)$$

e

$$y_k^{(l)}(n) = \phi(v_k^{(l)}(n)). \quad (2.14)$$

Figura 2.4: Parte real (a) e módulo (b) da função $\phi(z)$.

Cálculo Regressivo: Para $l = L, \dots, 1$ e $k = 1, \dots, N_l$, calcule:

1.

$$e_k^{(l)}(n) = \begin{cases} d_k(n) - y_k^{(l)}(n) & \text{para } l = L \\ \sum_{m=1}^{N_{l+1}} w_{mk}^{*(l+1)}(n) \Delta_m^{(l+1)}(n) & \text{para } l = L-1, \dots, 1, \end{cases} \quad (2.15)$$

sendo

$$\Delta_m^{(l)}(n) = e_{mR}^{(l)}(n) \varphi'(v_{mR}^{(l)}(n)) + j e_{mI}^{(l)}(n) \varphi'(v_{mI}^{(l)}(n)). \quad (2.16)$$

2.

$$w_{km}^{(l)}(n+1) = w_{km}^{(l)}(n) + 2\mu \Delta_k^{(l)}(n) y_m^{*(l-1)}(n), \quad \text{para } m = 0, 1, \dots, N_{l-1}. \quad (2.17)$$

Como no algoritmo BP real, o nível de ajuste (*bias*) de cada neurônio está embutido no vetor de pesos, ou seja $y_0^{(l-1)}(n) = -1$ e $w_{k0}^{(l)}(n) = \theta_k^{(l)}(n)$. Os pesos devem ser inicializados com números aleatórios complexos e uniformemente distribuídos num intervalo de pequenos valores. Maiores detalhes na dedução deste algoritmo podem ser encontrados em [10].

2.3.2 Algoritmo RLS-I

Na formulação de algoritmos baseados no método dos mínimos quadrados recursivo é mais conveniente utilizar a notação matricial. Para isso, considerou-se o esquema do nó k da camada l mostrado na Figura 2.5. A partir desta figura definiram-se os seguintes vetores:

$$\mathbf{y}^{(l-1)}(n) = [-1 \quad y_1^{(l-1)}(n) \quad y_2^{(l-1)}(n) \quad \cdots \quad y_{N_{l-1}}^{(l-1)}(n)]^T, \quad (2.18)$$

$$\mathbf{w}_k^{(l)}(n) = [w_{k0}^{(l)}(n) \quad w_{k1}^{(l)}(n) \quad w_{k2}^{(l)}(n) \quad \cdots \quad w_{kN_{l-1}}^{(l)}(n)]^T. \quad (2.19)$$

Deve-se notar que a primeira posição do vetor $\mathbf{w}_k^{(l)}$ é o nível de ajuste (*bias*) do neurônio k da camada l e para isso a primeira posição do vetor $\mathbf{y}^{(l-1)}$ é mantida constante e igual a -1.

Figura 2.5: Modelo do k-ésimo neurônio da camada l .

Em implementações recursivas do método dos mínimos quadrados para o treinamento de redes neurais é considerada a seguinte medida de desempenho:

$$Q(n) = \sum_{m=1}^n \lambda^{n-m} \sum_{k=1}^{N_L} [e_k^{(L)}(m)]^2, \quad (2.20)$$

sendo N_L o número de neurônios da camada de saída, λ o fator de ponderação (ou esquecimento) $0 \ll \lambda < 1$ e $e_k^{(L)}$ o erro do k -ésimo neurônio de saída [11]. Explicitando o cálculo do erro em (2.20) obtém-se:

$$Q(n) = \sum_{m=1}^n \lambda^{n-m} \sum_{k=1}^{N_L} [d_k^{(L)}(m) - \varphi(\mathbf{y}^{(L-1)T}(m) \mathbf{w}_k^{(L)}(n))]^2, \quad (2.21)$$

sendo $\mathbf{y}^{(L-1)}$ o vetor de saídas da camada $L-1$ definido na Equação (2.18), $d_k^{(L)}$ o sinal desejado para o neurônio de saída k , $\mathbf{w}_k^{(L)}$ o vetor de pesos do neurônio k da camada de saída (L) definido na Equação (2.19) e $\varphi(\cdot)$ a função de ativação não-linear definida na Equação (2.3).

O índice de desempenho $Q(n)$ é então minimizado tomando-se a derivada parcial de $Q(n)$ em relação a $\mathbf{w}_i^{(l)}$ e igualando a zero. Procedendo então os cálculos, obtém-se as equações do algoritmo RLS-I que podem ser vistas na Tabela 2.7. No Apêndice B encontram-se maiores detalhes da dedução deste algoritmo.

O algoritmo da Tabela 2.7 garante a simetria da matriz de autocorrelação inversa. Deve-se notar ainda que, considerando o sinal $\tilde{\mathbf{y}}_k^{(l-1)}$ como a entrada do nó k da camada l , o algoritmo obtido para a atualização dos pesos deste nó é o mesmo RLS de filtragem adaptativa linear [1] que garante a simetria da matriz de autocorrelação inversa, com o sinal de erro definido como na Tabela 2.7. Além disso, $\mathbf{P}_k^{(l)}$ é a inversa da matriz de autocorrelação do nó k da camada l e $\mathbf{g}_k^{(l)}$ é o ganho de Kalman deste mesmo nó.

Os pesos e os níveis de ajuste (*bias*) devem ser inicializados como no algoritmo *Backpropagation*. Notar que os níveis de ajuste estão incluídos no vetor de pesos. As matrizes de autocorrelação inversa devem ser inicializadas como $\mathbf{P}_k^{(l)}(0) = \delta^{-1} \mathbf{I}$ sendo δ uma constante positiva, pequena e não nula e $\mathbf{I}$ a matriz identidade.

Fazendo uma análise da complexidade computacional do algoritmo RLS-I para o treinamento de redes MLP com L camadas seguindo a configuração $(N_1, N_2, \dots, N_L)$ e

Inicialização: $\mathbf{P}_k^{(l)}(0) = \delta^{-1}\mathbf{I}$, $l = 1, 2, \dots, L$ e $k = 1, 2, \dots, N_l$ $\delta =$ constante positiva pequena   Os pesos devem ser inicializados com números aleatórios uniformemente distribuídos num intervalo de pequenos valores.   Para $l = 1, 2, \dots, L$ e $k = 1, 2, \dots, N_l$, calcule: $v_k^{(l)}(n) = \mathbf{y}^{(l-1)T}(n)\mathbf{w}_k^{(l)}(n)$ $y_k^{(l)}(n) = \varphi(v_k^{(l)}(n))$ Para $l = L, L-1, \dots, 1$ e $k = 1, 2, \dots, N_l$, calcule: $e_k^{(l)}(n) = \begin{cases} d_k^{(l)}(n) - y_k^{(l)}(n) & l = L \\ \sum_{m=1}^{N_{l+1}} \varphi'(v_m^{(l+1)}(n))\mathbf{w}_{mk}^{(l+1)}(n)e_m^{(l+1)}(n) & l \neq L \end{cases}$ $\tilde{\mathbf{y}}_k^{(l)}(n) = \varphi'(v_k^{(l)}(n))\mathbf{y}^{(l)}(n)$ $\mathbf{g}_k^{(l)}(n) = \mathbf{P}_k^{(l)}(n-1)\tilde{\mathbf{y}}_k^{(l-1)}(n)$ $\alpha_k^{(l)}(n) = \lambda + \tilde{\mathbf{y}}_k^{(l-1)T}(n)\mathbf{g}_k^{(l)}(n)$ $\mathbf{P}_k^{(l)}(n) = \lambda^{-1} \left[\mathbf{P}_k^{(l)}(n-1) - \frac{\mathbf{g}_k^{(l)}(n)\mathbf{g}_k^{(l)T}(n)}{\alpha_k^{(l)}(n)} \right]$ $\mathbf{w}_k^{(l)}(n) = \mathbf{w}_k^{(l)}(n-1) + \mathbf{g}_k^{(l)}(n) \frac{e_k^{(l)}(n)}{\alpha_k^{(l)}(n)}$

Tabela 2.7: Algoritmo RLS-I.

considerando sinais reais, obtém-se os números de operações por iteração da Tabela 2.8. Nesta tabela, N_0 é o número de entradas da rede e NL representa o cálculo da função não-linear ou da sua derivada. Considerando a rede MLP treinada, a complexidade computacional é a mesma exposta na Tabela 2.6 pois ela independe do algoritmo de treinamento.

O algoritmo RLS-I está deduzido no Apêndice B para sinais reais. Embora apresente uma velocidade de convergência bem superior que a do algoritmo BP, possui uma complexidade computacional muito maior. Para obter sua versão complexa, seria necessário considerar a função de ativação definida na Equação (2.12) e deduzir o

algoritmo de forma similar à do caso real. Isso, por sua vez, dá origem a um algoritmo cuja complexidade computacional prejudica seu uso na prática o que fez com que descartássemos a utilização do mesmo no caso complexo.

Operações	Número de operações por iteração
$\times$	$4N_A + N_B + 10N_C - N_1$
$\div$	N_C
NL	$2N_C$
$+$	$2N_A + 2N_C + N_D + 2N_L$

Tabela 2.8: Complexidade computacional do algoritmo RLS-I para sinais reais, na fase de treinamento de uma rede MLP $(N_1, N_2, \dots, N_L)$, sendo $N_A = N_0^2 N_1 + N_1^2 N_2 + N_2^2 N_3 + \dots + N_{L-1}^2 N_L$, $N_B = 12N_0 N_1 + 13(N_1 N_2 + N_2 N_3 + \dots + N_{L-1} N_L)$, $M_C = M_1 + M_2 + \dots + M_L$, $M_D = 6N_0 N_1 + 7(N_1 N_2 + N_2 N_3 + \dots + N_{L-1} N_L)$.

2.3.3 Algoritmo RLS-II

Um outro algoritmo baseado no método dos mínimos quadrados recursivo para treinamento de redes do tipo MLP foi proposto em [12]. Tal algoritmo é computacionalmente mais simples pois utiliza uma matriz de autocorrelação inversa por camada e não por neurônio como é feito no algoritmo RLS-I. Além disso, como função de ativação, ao invés da função sigmoideal (Figura 2.3), foi utilizada uma aproximação da mesma como pode ser visto na Figura 2.6 para $c_l = 1$ e $c_l = 2$. Deve-se notar que o parâmetro c_l da função de ativação é dependente da camada l , ou seja, para cada camada pode-se utilizar uma função neuronal diferente.

A dedução do algoritmo está intimamente ligada à função de ativação considerada, como pode ser observado em suas equações.

O algoritmo RLS-II pode ser resumido pelos seguintes passos:

1. Os pesos e os níveis de ajuste (*bias*) devem ser inicializados com pequenos números aleatórios de distribuição uniforme e média nula. Os níveis de ajuste

Figura 2.6: Função não-linear e sua derivada com $c_l = 1$ e $c_l = 2$.

(*bias*) estão incluídos no vetor de pesos. As matrizes de autocorrelação inversa devem ser inicializadas como $\mathbf{P}^{(l)}(0) = \delta^{-1}\mathbf{I}$ sendo δ uma constante positiva, pequena e não nula e $\mathbf{I}$ a matriz identidade.

2. O sinal de entrada deve ser propagado progressivamente a fim de se obter as saídas da rede em cada iteração utilizando-se as seguintes equações

$$v_k^{(l)}(n) = \mathbf{y}^{(l-1)T}(n)\mathbf{w}_k^{(l)}(n) \quad (2.22)$$

e

$$y_k^{(l)}(n) = \begin{cases} -1 & v_k^{(l)}(n) \leq -c_l \\ \frac{1}{c_l} v_k^{(l)}(n) & -c_l < v_k^{(l)}(n) < c_l \\ 1 & v_k^{(l)}(n) \geq c_l. \end{cases} \quad (2.23)$$

Os vetores $\mathbf{y}^{(l-1)}(n)$ e $\mathbf{w}_k^{(l)}(n)$ estão definidos nas equações (2.18) e (2.19) respectivamente.

3. O sinal de erro deve ser obtido na saída de cada neurônio. Para os neurônios da camada de saída da rede, o erro é formado pela comparação de sua saída com o sinal desejado

$$e_k^{(L)}(n) = d_k^{(L)}(n) - y_k^{(L)}(n). \quad (2.24)$$

Para as demais camadas, o erro é obtido através de uma propagação regressiva, utilizando-se a seguinte equação

$$e_k^{(l)}(n) = \frac{1}{c_{l+1}} \sum_{m=1}^{N_{l+1}} w_{mk}^{(l+1)}(n) e_m^{(l+1)}(n). \quad (2.25)$$

4. Neste passo deve-se calcular as matrizes $\mathbf{P}^{(l)}$ (inversas da matrizes de autocorrelação) e os vetores $\mathbf{g}^{(l)}$ (ganhos de Kalman) para $l = 1, 2, \dots, L$, usando as seguintes equações

$$\mathbf{P}^{(l)}(n) = \lambda^{-1} \left[I - \mathbf{g}^{(l)}(n) \mathbf{y}^{(l-1)T}(n) \right] \mathbf{P}^{(l)}(n-1) \quad (2.26)$$

e

$$\mathbf{g}^{(l)}(n) = \frac{\mathbf{P}^{(l)}(n-1) \mathbf{y}^{(l-1)}(n)}{\lambda + \mathbf{y}^{(l-1)T}(n) \mathbf{P}^{(l)}(n-1) \mathbf{y}^{(l-1)}(n)}. \quad (2.27)$$

Nas equações acima, λ representa o fator de ponderação (ou esquecimento).

5. Se $v_k^{(l)}(n) \notin]-c_l, c_l[$ os pesos não devem ser alterados, no entanto, se $v_k^{(l)}(n) \in]-c_l, c_l[$ os pesos do nó k da camada l devem ser atualizados segundo

a equação

$$\mathbf{w}_k^{(l)}(n) = \mathbf{w}_k^{(l)}(n-1) + \mathbf{g}^{(l)}(n) \left[y_k^{(l)}(n) + e_k^{(l)}(n) - \frac{1}{c_l} \mathbf{y}^{(l-1)T}(n) \mathbf{w}_k^{(l)}(n-1) \right]. \quad (2.28)$$

O passo 5 deve ser repetido até que todos os pesos da rede estejam atualizados. Depois deve-se voltar ao passo 2 até que se atinja um erro mínimo pré-estabelecido.

A complexidade computacional deste algoritmo na fase de treinamento para redes MLP com L camadas seguindo a configuração $(N_1, N_2, \dots, N_L)$ pode ser expressa pelos números de operações por iteração da Tabela 2.9.

Operações	Número de operações por iteração
$\times$	$N_A + 6N_B + N_C + N_D + 5L$
$\div$	L
NL	$2(N_1 + N_2 + \dots + N_L)$
$+$	$N_A + 5N_B + N_C + N_E + 2L$

Tabela 2.9: Complexidade computacional do algoritmo RLS-II para sinais reais, na fase de treinamento de uma rede MLP $(N_1, N_2, \dots, N_L)$, sendo $N_A = N_0^3 + N_1^3 + \dots + N_{L-1}^3$, $N_B = N_0^2 + N_1^2 + \dots + N_{L-1}^2$, $N_C = 3N_0N_1 + 4(N_1N_2 + N_2N_3 + \dots + N_{L-1}N_L)$, $N_D = 12N_0 + 17N_1 + 18(N_2 + \dots + N_L)$, $N_E = 6N_0 + 8(N_1 + N_2 + \dots + N_{L-1}) + 3N_L$.

Como o algoritmo RLS-I, o algoritmo RLS-II aqui exposto é para sinais reais. Sua versão complexa não foi deduzida porque também apresenta uma elevada complexidade computacional o que inviabiliza sua utilização em tempo real. Maiores detalhes da dedução deste algoritmo podem ser encontrados na referência [12].

2.3.4 Algoritmo Acelerador (AC)

O algoritmo Acelerador de parâmetros de tempo discreto para filtragem adaptativa linear está descrito na Seção 2.2.4.

No caso da filtragem linear foi verificado que ele apresenta uma velocidade de convergência maior que o LMS. Além disso possui uma complexidade computacional menor que a do RLS [6]. O interesse do uso deste algoritmo para o treinamento de uma rede do tipo MLP é verificar se este comportamento observado na filtragem linear se repete quando o mesmo é usado para filtragem adaptativa não-linear. Assim, deduziu-se um algoritmo para filtragem não-linear tomando como base o algoritmo Acelerador linear da Seção 2.2.4.

As deduções das versões real e complexa do algoritmo Acelerador para o treinamento de redes neurais do tipo MLP estão apresentadas nos Apêndices C e D respectivamente.

Uma parte deste estudo foi apresentado em forma de um artigo no XVIII Simpósio Brasileiro de Telecomunicações-SBrT (Referência [51]).

Algoritmo Acelerador Real

O algoritmo Acelerador real para treinamento de redes do tipo MLP em sua forma eficiente está apresentado na Tabela 2.10. Deve-se notar que a propagação regressiva do erro é feita da mesma forma que nos algoritmos RLS-I e *Backpropagation*, só que aqui se faz a retropropagação do erro *a priori* ($e_{a,i}^{(l)}$).

As equações do cálculo regressivo deste algoritmo foram obtidas a partir da minimização da medida de desempenho $\xi(n)$ definida na Equação (2.4). O cálculo progressivo foi obtido a partir da introdução da função de ativação não-linear $\varphi(\cdot)$ no algoritmo apresentado na Seção 2.2.4, seguida de aproximações por série de Taylor de 1ª ordem.

Para garantir a estabilidade do algoritmo Acelerador segundo Liapunov, os valores de m_1 , m_2 e m_3 devem satisfazer

$$4m_1^2m_2m_3 \geq 1. \quad (2.29)$$

Uma das dificuldades deste algoritmo, é o ajuste dos parâmetros μ , m_1 , m_2 e m_3 a fim de se obter uma alta velocidade de convergência. De [5], [6] e [7], sabe-se que a velocidade de convergência do algoritmo Acelerador é máxima quando os parâmetros m_1 , m_2 e m_3 estão próximos da condição de estabilidade crítica ($4m_1^2m_2m_3 = 1$). Além disso, dentro de certos limites a velocidade de convergência é diretamente proporcional a m_1 e μ . Mais precisamente, considerando um passo de aprendizagem (μ) fixo e as condições $4m_1^2m_2m_3 = 1$ e $m_2 = m_3$, não se consegue aumentar a velocidade de convergência do algoritmo após um certo limite máximo para m_1 , sendo que de certa forma a velocidade de convergência fica limitada pelo passo de aprendizagem. Além disso, em situações bastante ruidosas, em que a relação sinal-ruído é inferior a 20 dB, valores muito grandes de m_1 provocam uma maior variância do erro médio quadrático, o que é indesejável. Dessa forma, um procedimento que se mostrou útil na prática foi o seguinte:

- usar m_2 com a mesma ordem de grandeza de m_3 , podendo inclusive serem iguais a $1/2m_1$;
- fixar um valor de μ ;
- achar um valor de m_1 que garanta a maior velocidade de convergência sem provocar grandes variações do erro.

Quanto à inicialização do algoritmo, os pesos e os níveis de ajuste (*bias*) devem ser inicializados com números aleatórios uniformemente distribuídos num intervalo de pequenos valores. Notar que os níveis de ajuste (*bias*) estão incluídos no vetor de pesos. Os vetores $\mathbf{q}_k^{(l)}$ com $k = 1, 2, \dots, N_l$ e $l = 1, 2, \dots, L$, do instante anterior ao inicial devem ser inicializados com o vetor nulo: $\mathbf{q}_k^{(l)}(-1) = \mathbf{0}$.

A complexidade computacional do algoritmo Acelerador real na fase de treinamento para redes MLP com L camadas, seguindo a configuração $(N_1, N_2, \dots, N_L)$, pode ser expressa pelos números de operações por iteração da Tabela 2.11. A função de ativação

$\varphi(\cdot)$ comumente utilizada é do tipo sigmoidal (Equação (2.3)).

Inicialização: $\mathbf{q}_k^{(l)}(-1) = \mathbf{0}$, $l = 1, 2, \dots, L$ e $k = 1, 2, \dots, N_l$.   Os pesos devem ser inicializados com números aleatórios uniformemente distribuídos num intervalo de pequenos valores. $\alpha = 1 + 2\mu m_1 m_2$ Para $l = 1, 2, \dots, L$ e $k = 1, 2, \dots, N_l$: $v_{a,k}^{(l)}(n) = \mathbf{y}^{(l-1)T}(n) \mathbf{w}_k^{(l)}(n-1)$ $y_k^{(l)}(n) = \varphi(v_{a,k}^{(l)}(n))$ $b_k^{(l)}(n) = \mu^2 \varphi'(v_{a,k}^{(l)}(n)) m_1 + 2\mu m_1^2 m_3$ Para $l = L, L-1, \dots, 1$ e $k = 1, 2, \dots, N_l$: $e_{a,k}^{(l)}(n) = \begin{cases} y_k^{(l)}(n) - d_k(n) & l = L \\ \sum_{m=1}^{N_{l+1}} \varphi'(v_{a,m}^{(l+1)}(n)) \cdot \mathbf{w}_{mk}^{(l+1)}(n-1) e_{a,m}^{(l+1)}(n) & l \neq L \end{cases}$ $\gamma_k^{(l)}(n) = \frac{b_k^{(l)}(n) \mathbf{y}^{(l-1)T}(n) \mathbf{q}_k^{(l)}(n-1) + \mu m_1 \alpha e_{a,k}^{(l)}(n)}{\alpha + b_k^{(l)}(n) \ \mathbf{y}^{(l-1)}(n)\ ^2}$ $\mathbf{q}_k^{(l)}(n) = \frac{1}{\alpha} \left[\mathbf{q}_k^{(l)}(n-1) - \gamma_k^{(l)}(n) \mathbf{y}^{(l-1)}(n) \right]$ $\mathbf{w}_k^{(l)}(n) = \mathbf{w}_k^{(l)}(n-1) + \mu \mathbf{q}_k^{(l)}(n)$

Tabela 2.10: Algoritmo Acelerador real para treinamento de redes do tipo MLP.

Operações	N^2 de operações
$\times$	$N_A + 12N_C - N_1$
$\div$	N_C
$+$	$N_B + 4N_C + 2N_L$
NL	$2N_C$

Tabela 2.11: Complexidade Computacional do algoritmo AC real, na fase de treinamento de uma rede MLP $(N_1, N_2, \dots, N_L)$, sendo $N_A = 6N_0N_1 + 7(N_1N_2 + N_2N_3 + \dots + N_{L-1}N_L)$, $N_B = 5N_0N_1 + 6(N_1N_2 + N_2N_3 + \dots + N_{L-1}N_L)$ e $N_C = N_1 + N_2 + \dots + N_L$.

Algoritmo Acelerador Complexo

O algoritmo Acelerador complexo para treinamento de redes neurais do tipo MLP foi obtido a partir da discretização de seu equivalente de tempo contínuo modificado. A modificação em questão consistiu na introdução da função de ativação para sinais complexos definida na Equação (2.12). A dedução deste algoritmo é bastante longa e está apresentada de forma resumida no Apêndice D. Tal algoritmo pode ser sintetizado pelas seguintes equações:

Cálculo Progressivo: Para $l = 1, \dots, L$ e $k = 1, \dots, N_l$, calcule:

$$v_{a,k}^{(l)}(n) = \sum_{m=0}^{N_{l-1}} w_{km}^{(l)*}(n-1)y_m^{(l-1)}(n) \quad (2.30)$$

e

$$y_k^{(l)}(n) = \phi(v_{a,k}^{(l)}(n)), \quad (2.31)$$

sendo $\phi(z) = \varphi(z_R) + j\varphi(z_I)$, $z = z_R + jz_I$ e $\varphi(\cdot)$ uma função não-linear do tipo sigmoidal.

Cálculo Regressivo: Para $l = L, L - 1, \dots, 1$ e $k = 1, \dots, N_l$, calcule:

1.

$$e_{a,k}^{(l)}(n) = \begin{cases} y_k^{(l)}(n) - d_k(n) & \text{para } l = L \\ \sum_{m=1}^{N_{l+1}} w_{mk}^{(l+1)}(n-1) \Delta_m^{(l+1)}(n) & \text{para } l = L-1, \dots, 1, \end{cases} \quad (2.32)$$

sendo

$$\Delta_m^{(l)}(n) = e_{a,mR}^{(l)}(n) \varphi'(v_{a,mR}^{(l)}(n)) + j e_{a,mI}^{(l)}(n) \varphi'(v_{a,mI}^{(l)}(n)). \quad (2.33)$$

2.

$$\mathbf{w}_k^{(l)}(n) = \mathbf{w}_k^{(l)}(n-1) + \mu \mathbf{q}_k^{(l)}(n), \quad (2.34)$$

sendo

$$\mathbf{q}_k^{(l)}(n) = \mathbf{q}_{kR}^{(l)}(n) + j \mathbf{q}_{kI}^{(l)}(n) \quad (2.35)$$

e

$$\begin{pmatrix} \mathbf{q}_{kR}^{(l)}(n) \\ \mathbf{q}_{kI}^{(l)}(n) \end{pmatrix} = (\Gamma_{c,k}^{(l)}(n))^{-1} \times \\ \times \begin{pmatrix} \mathbf{q}_{kR}^{(l)}(n-1) - \mu \mathbf{M}_1(\mathbf{y}_{kR}^{(l-1)}(n) e_{a,kR}^{(l)}(n) + \mathbf{y}_{kI}^{(l-1)}(n) e_{a,kI}^{(l)}(n)) \\ \mathbf{q}_{kI}^{(l)}(n-1) - \mu \mathbf{M}_1(\mathbf{y}_{kI}^{(l-1)}(n) e_{a,kR}^{(l)}(n) + \mathbf{y}_{kR}^{(l-1)}(n) e_{a,kI}^{(l)}(n)) \end{pmatrix}. \quad (2.36)$$

A definição e cálculo da matriz $(\Gamma_{c,k}^{(l)}(n))^{-1}$ bem como a escolha da matriz $\mathbf{M}_1$ estão mostrados no Apêndice D.

Como no caso real, os vetores $\mathbf{q}_k^{(l)}$ para $l = L, L-1, \dots, 1$ e $k = 1, \dots, N_l$, do algoritmo Acelerador complexo do instante anterior ao inicial devem ser inicializados com o vetor nulo. Os pesos devem ser inicializados com valores complexos aleatórios distribuídos uniformemente num intervalo de pequenos valores. Embora não tenha sido feita uma análise teórica da estabilidade deste algoritmo, resultados de simulações

evidenciam que ele é estável quando $4\mathbf{M}_1\mathbf{M}_3\mathbf{M}_1 \geq \mathbf{M}_2^{-1}$ e $\mathbf{M}_2\mathbf{M}_1\mathbf{M}_3 + \mathbf{M}_3\mathbf{M}_1\mathbf{M}_2 \geq \mathbf{M}_1^{-1}/2$. Esta é condição de estabilidade segundo Liapunov para o algoritmo Acelerador matricial no caso linear. As matrizes $\mathbf{M}_i$, $i = 1, 2, 3$, dão origem aos parâmetros de aceleração m_i , $i = 1, 2, 3$, quando se considera uma versão simplificada e não-matricial do algoritmo.

Fazendo algumas simplificações, há a possibilidade de se evitar o cálculo da matriz inversa $(\Gamma_{c,k}(n))^{-1}$, chegando-se num algoritmo não-matricial que também está mostrado no Apêndice D.

2.4 A rede RBF (*Radial Basis Function*)

Uma rede MLP treinada com o algoritmo *Backpropagation* pode ser considerada como uma aplicação de um método de otimização conhecido como aproximação estocástica [8]. O projeto de uma rede RBF, por sua vez, pode ser encarado como um problema de aproximação num espaço de alta dimensão. De acordo com este último ponto de vista, o aprendizado é equivalente a encontrar uma superfície num espaço multi-dimensional que possibilita uma melhor aproximação de uma função descrita de alguma forma por dados de treinamento. A generalização para novos dados é equivalente ao uso desta superfície multi-dimensional a fim de interpolá-los [8]. Este ponto de vista é a motivação do método que utiliza as funções de base radial. No contexto de redes neurais, os neurônios ocultos proporcionam um conjunto de funções que constituem uma base para representar os padrões de entrada no espaço gerado pelas funções de base da camada oculta. Como funções de base são utilizadas funções com simetria radial, o que fornece à rede o nome de RBF.

A rede RBF consiste usualmente de três camadas de neurônios sendo que a primeira alimenta a segunda diretamente (pesos iguais a 1) e a camada de saída é apenas um combinador linear [8], [13], [14] como está esquematizado na Figura 2.7.

A camada oculta implementa um mapeamento não-linear que é calculado em dois

Figura 2.7: Arquitetura de uma rede RBF com M entradas e uma saída.

passos. No primeiro passo, o sinal de entrada $\mathbf{u}(n) = [u(n) \ u(n-1) \ \cdots \ u(n-M+1)]^T$ é comparado com um conjunto de vetores de referência $\mathbf{r}_k(n)$ para $k = 1, 2, \dots, N$ sendo N o número de nós (ou neurônios) ocultos. Estes vetores são chamados de centros. A comparação entre o sinal de entrada e os centros pode ser feita através da norma Euclidiana gerando um conjunto de distâncias

$$s_k(n) = \|\mathbf{u}(n) - \mathbf{r}_k(n)\|, \text{ para } k = 1, 2, \dots, N. \quad (2.37)$$

Estas distâncias são então aplicadas a uma função de ativação não-linear radialmente simétrica. Escolhas típicas são gaussianas e *thinplate-spline functions*, respectivamente dadas por

$$f(s_k(n)) = \exp\left(\frac{-s_k^2(n)}{\sigma_k^2(n)}\right) \quad (2.38)$$

e

$$f(s_k(n)) = \frac{s_k^2(n)}{\sigma_k^2(n)} \log\left(\frac{s_k(n)}{\sigma_k(n)}\right). \quad (2.39)$$

Dentre essas duas funções, a função gaussiana é a mais usada na prática. O parâmetro $\sigma_k(n)$ controla o espalhamento da função e conseqüentemente o seu raio de influência. O sinal de saída da rede é calculado como

$$y(n) = \varphi\left[\sum_{k=0}^N w_k(n)f(s_k(n))\right], \quad (2.40)$$

sendo $f(s_0(n)) = 1$ por definição e $\varphi(\cdot)$ a função de ativação, usualmente sigmoideal como definida na Equação (2.3) para sinais reais. Quando os sinais são complexos utiliza-se a função de ativação $\phi(\cdot)$ definida na Equação (2.12). Também neste caso $w_0(n)$ representa o nível de ajuste (*bias*) da rede.

2.4.1 O treinamento da rede RBF

O desempenho da rede RBF está intimamente ligado à correta determinação dos centros e raios das funções de base radial. Em alguns casos, a escolha desses centros baseada na distribuição do sinal de entrada da rede garante bons resultados quando se usa um grande número de neurônios. Essa solução, no entanto, possui uma complexidade muito alta. Dessa forma, técnicas baseadas em análises de agrupamento (*cluster analysis*) foram desenvolvidas a fim de adaptar os centros e variâncias de forma a permitir a redução do número de neurônios e o uso da rede RBF em tempo real [15]. Uma dessas técnicas é a que obtém agrupamentos do sinal de entrada de forma não supervisionada, conhecida como *Unsupervised Clustering Learning* (UCL) [15], [17], que pode ser dividida entre a adaptação dos centros e das variâncias, como está detalhado a seguir.

Adaptação dos centros Para a adaptação dos centros, faz-se o cálculo das distâncias ao quadrado entre os centros e o vetor de entrada da rede. Calculadas estas distâncias,

seleciona-se a menor e adapta-se o correspondente centro para que ele fique mais próximo do vetor de entrada. Isso pode ser escrito matematicamente através da seguinte formulação

$$\left. \begin{aligned} t &= \arg[\min\{s_k^2(n), 1 \leq k \leq N\}] \\ \mathbf{r}_t(n) &= \mathbf{r}_t(n-1) + \mu_r(\mathbf{u}(n) - \mathbf{r}_t(n-1)) \\ \mathbf{r}_k(n) &= \mathbf{r}_k(n-1), 1 \leq k \leq N, k \neq t, \end{aligned} \right\} \quad (2.41)$$

com $s_k(n)$ definido na Equação (2.37) e N sendo o número de neurônios da camada oculta.

Adaptação das variâncias Se os centros estiverem posicionados corretamente nos estados desejados e o vetor de entrada $\mathbf{u}(n)$ é coincidente com k -ésimo estado ($\mathbf{r}_k$) a menos do ruído, então a variância do ruído no instante n é dada por $\sigma^2(n) = E[\|\mathbf{u}(n) - \mathbf{r}_k\|^2/M]$, sendo M o número de entradas da rede e $E[\cdot]$ o operador esperança [15]. Isso sugere a seguinte estimativa para a variância do ruído

$$\hat{\sigma}_k^2(n) = \frac{(n-1)\hat{\sigma}_k^2(n-1) + \frac{s_k^2(n)}{M}}{n}. \quad (2.42)$$

Esta técnica de adaptação dos centros e variâncias das funções de base radial é bastante usada em equalização pois ela também funciona quando os canais são variantes no tempo [15].

Feita a adaptação dos centros e variâncias das funções de base radial ainda é necessário adaptar os pesos da rede. Isso será discutido a seguir.

Adaptação dos pesos Na adaptação dos pesos podem ser usados os tradicionais algoritmos de filtragem adaptativa linear como o LMS e RLS expostos nas seções 2.2.1 e 2.2.2 respectivamente. Se a adaptação dos centros e variâncias não for feita, não há muita diferença entre as velocidades de convergência desses dois algoritmos. Por outro lado, fazendo a adaptação dos centros e variâncias das funções de base radial segundo o algoritmo exposto anteriormente, o RLS possui uma convergência potencialmente mais rápida e uma complexidade computacional muito maior que o LMS.

Um outro algoritmo que pode ser usado é o algoritmo Acelerador, que apresenta um comportamento intermediário entre o LMS e RLS. Neste caso, não se faz modificações no algoritmo utilizado em filtragem linear apresentado na seção 2.2.4. Além disso, como o comportamento deste algoritmo aplicado em filtragem adaptativa linear já é mais conhecido, consegue-se treinar a rede RBF sem grandes problemas na escolha dos parâmetros m_1 , m_2 , m_3 e μ , o que de certa forma, é uma das grandes dificuldades em sua utilização para o treinamento de redes MLP.

Há ainda uma variação do algoritmo LMS que é a consideração de um *momentum* constante β na equação de adaptação dos pesos:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu e(n)f(\mathbf{s}(n)) + \beta[\mathbf{w}(n) - \mathbf{w}(n-1)]. \quad (2.43)$$

A utilização deste *momentum* proporciona em muitos casos uma aceleração na velocidade de convergência deste algoritmo.

Considerando sinais reais, a complexidade computacional de uma rede RBF na fase de treinamento pode ser expressa pelos números de operações por iteração da Tabela 2.12. Nesta tabela, considerou-se o algoritmo LMS com *momentum*. Além disso, pode-se observar que devido à sua elevada complexidade computacional, o algoritmo RLS torna-se inviável para o treinamento dos pesos desta rede. Para efeito de comparação, os números de operações por iteração foram obtidos com e sem o uso do algoritmo de adaptação dos centros e variâncias (UCL) para o algoritmo LMS.

Considerando uma rede RBF treinada e sinais reais, chega-se no número de operações por iteração da Tabela 2.13.

Uma consideração é necessária quando se usa sinais complexos: observando a Equação (2.37), as distâncias $s_k(n)$ são sempre reais, independente se os centros $\mathbf{r}_k(n)$ e o vetor de entrada $\mathbf{u}(n)$ são reais ou complexos. Isso faz com que os algoritmos de treinamento da rede RBF para sinais complexos sejam os mesmos utilizados para sinais reais. A única diferença neste caso, é que os pesos se tornam complexos o que é compatível com uma saída $y(n)$ complexa [16], [17].

	Número de operações por iteração			
Op.	LMS	LMS+UCL	AC+UCL	RLS+UCL
$\times$	$N(M + 3) + 4$	$N(M + 5) + M + 4$	$N(M + 8) + M + 10$	$5N^2 + N(M + 16) + M + 10$
$\div$	-	N	N	$N + 1$
NL	$N + 1$	$N + 1$	$N + 1$	$N + 1$
$+$	$N(2M + 2) + 3$	$N(2M + 3) + M + 3$	$N(2M + 5) + M + 5$	$4N^2 + N(2M + 8) + M + 4$

Tabela 2.12: Complexidade computacional na fase de treinamento de uma rede RBF com uma saída, M entradas, N neurônios, considerando 4 algoritmos de treinamento e sinais reais.

Operações	Nº de operações por iteração
$\times$	$N(M + 1) + 1$
NL	$N + 1$
$+$	$2NM$

Tabela 2.13: Complexidade computacional de uma rede RBF treinada, com uma saída, M entradas, N neurônios e sinais reais.

2.5 A rede neural recorrente (RNN)

Uma RNN tem em geral M entradas externas e N unidades completamente interconectadas [18]. Um exemplo com $N = 3$ está mostrado na Figura 2.8.

No caso de uma RNN, a saída de uma unidade no tempo $n + 1$ não somente depende de entradas externas da rede do instante de tempo anterior ($u(n - k)$, $k = 0, \dots, M - 1$), mas também de saídas anteriores das unidades $y_k(n)$, $k = 1, \dots, N$. A dinâmica de uma RNN é descrita pelas equações

$$v_k(n + 1) = \sum_{m=1}^N w_{km}(n) y_m(n) + \sum_{m=N+1}^{N+M} w_{km}(n) u(n - m + N + 1) \quad (2.44)$$

Figura 2.8: RNN com M entradas, $N = 3$ neurônios e saída y_3 .

e

$$y_k(n+1) = \varphi(v_k(n+1)). \quad (2.45)$$

Nestas expressões $w_{km}(n)$ é o peso da conexão entre as unidades m e k no tempo n e $\varphi(\cdot)$ é usualmente a função de ativação dada pela Equação (2.3) considerando sinais reais. Quando os sinais são complexos, como será visto posteriormente, utiliza-se a função de ativação $\phi(\cdot)$ definida na Equação (2.12).

Vários algoritmos têm sido propostos para o treinamento de RNN's. Em [19] foi feito um estudo comparativo de alguns desses algoritmos e conclui-se que o RTRL (*Real Time Recurrent Learning*) apresenta vantagens no que diz respeito à estabilidade, exatidão e uso em tempo real. Por outro lado, é um algoritmo que possui uma complexidade

computacional bem maior que os demais. Diante dessas características, a fim de se fazer um estudo inicial desta rede aplicada à equalização de canais de comunicação, utilizou-se o algoritmo RTRL em seu treinamento.

2.5.1 Algoritmo RTRL

O algoritmo RTRL, assim como o *Backpropagation*, visa minimizar uma função custo definida como:

$$\xi(n) = \frac{1}{2} \sum_{k \in \mathcal{N}_s} |e_k(n)|^2, \quad (2.46)$$

sendo $\mathcal{N}_s$ o conjunto formado por todos os neurônios de saída da rede (Na Figura 2.8, $\mathcal{N}_s$ contém apenas o neurônio 3) e $e_k(n)$ o sinal de erro calculado nos neurônios de saída para a n -ésima iteração. O algoritmo RTRL é obtido minimizando-se a função custo utilizando o método do gradiente determinístico e introduzindo-se algumas grandezas de sensibilidade. A dedução deste algoritmo para os casos real e complexo está apresentada no Apêndice E.

Algoritmo RTRL Real

Resumidamente, o algoritmo RTRL consiste nos seguintes passos [18], [8]:

1. Os pesos devem ser inicializados com números aleatórios uniformemente distribuídos num intervalo de pequenos valores como por exemplo $[-10^{-2}, 10^{-2}]$.
2. Para cada instante de tempo n , partindo de $n = 0$, deve-se usar as equações (2.44) e (2.45) para calcular os valores de saída das N unidades.
3. O sinal de erro deve ser calculado pela seguinte equação

$$e_k(n) = \begin{cases} d_k(n) - y_k(n) & k \in \mathcal{N}_s \\ 0 & k \notin \mathcal{N}_s, \end{cases} \quad (2.47)$$

sendo $d_k(n)$ o sinal desejado (para equalização é o símbolo transmitido atrasado de τ_d amostras: $a(n - \tau_d)$) e $\mathcal{N}_s$ é o conjunto das unidades cujas saídas são

saídas da rede. No exemplo da Figura 2.8, o conjunto $\mathcal{N}_s$ é formado apenas pelo neurônio 3.

4. Os pesos devem ser atualizados de acordo com a equação

$$w_{km}(n+1) = w_{km}(n) + \Delta w_{km}(n), \quad (2.48)$$

sendo

$$\Delta w_{km}(n) = \mu \sum_{p \in \mathcal{N}_s} e_p(n) \Pi_{km}^p(n) \quad (2.49)$$

e

μ o passo de aprendizagem.

O termo $\Pi_{km}^p(n)$ deve ser calculado como

$$\Pi_{km}^p(n) = \varphi'(v_p(n-1)) \left[\sum_{q \in \mathcal{N}_o} w_{pq}(n-1) \Pi_{km}^q(n-1) + \delta_{kp} u_m(n-1) \right], \quad (2.50)$$

$$\Pi_{km}^p(0) = 0, \quad (2.51)$$

sendo

$$u_m(n) = \begin{cases} u(n-m+N+1) & \text{para } m \in \mathcal{N}_e \\ y_m(n) & \text{para } m \in \mathcal{N}_o, \end{cases} \quad (2.52)$$

$$\delta_{kp} = \begin{cases} 1 & \text{para } k = p \\ 0 & \text{para } k \neq p, \end{cases} \quad (2.53)$$

$\mathcal{N}_o = \{1, 2, \dots, N\}$ o conjunto formado por todas as unidades menos as de entrada da rede,

e

$\mathcal{N}_e = \{N+1, \dots, N+M\}$ o conjunto formado apenas pelas unidades de entrada.

A complexidade computacional do algoritmo RTRL real para uma rede com N unidades completamente interconectadas e M entradas externas pode ser expressa pelos números de operações por iteração da Tabela 2.14.

Operações	Nº de operações por iteração
$\times$	$M[N^2(N + 1) + 2] + (N + 1)(N^3 + 1) + 2N^2$
NL	$2N$
$+$	$MN(N^2 - N - 3) + (1 - N)(1 - N^3) + 3N^2$

Tabela 2.14: Complexidade Computacional do algoritmo RTRL real, no treinamento de uma rede RNN com M entradas e N unidades.

Considerando agora a rede treinada, o número de operações se reduz para os expostos na Tabela 2.15.

Operações	Número de operações por iteração
$\times$	$MN + N^2$
NL	N
$+$	$MN + N(N - 1)$

Tabela 2.15: Complexidade Computacional de uma rede RNN treinada para sinais reais (M entradas e N unidades).

Algoritmo RTRL Complexo

Como na dedução do algoritmo *Backpropagation* complexo, um problema também encontrado no treinamento da rede RNN para sinais complexos é a escolha apropriada da função de ativação [20]. Seguindo a solução sugerida em [10], foi considerada a seguinte função de ativação

$$\phi(v) = \varphi(v_R) + j\varphi(v_I), \quad (2.54)$$

sendo o v um sinal complexo, tal que $v = v_R + jv_I$ e $\varphi(\cdot)$, uma função não-linear do tipo sigmoidal.

O algoritmo RTRL complexo (CRTRL) pode ser resumido pelas equações mostradas a seguir [20].

Cálculo Progressivo Para $k = 1, \dots, N$, calcule:

$$v_k(n+1) = \sum_{m=1}^N w_{km} y_l(n) + \sum_{m=N+1}^{N+M} w_{km} u(n-m+N+1), \quad (2.55)$$

e

$$\begin{aligned} y_k(n+1) &= \phi(v_k(n+1)) \\ &= \varphi(v_{k_R}(n+1)) + j\varphi(v_{k_I}(n+1)). \end{aligned} \quad (2.56)$$

Deve-se notar que estas equações são idênticas às do caso real, com exceção da função de ativação $\phi(\cdot)$.

Cálculo Regressivo Introdz-se inicialmente os termos de sensibilidade P_{RR} , P_{RI} , P_{IR} , P_{II} definidos por:

$$\begin{bmatrix} P_{RRpq}^k & P_{RIpq}^k \\ P_{IRpq}^k & P_{IIpq}^k \end{bmatrix}_n = \begin{bmatrix} \frac{\partial y_{k_R}}{\partial w_{pqR}} & \frac{\partial y_{k_R}}{\partial w_{pqI}} \\ \frac{\partial y_{k_I}}{\partial w_{pqR}} & \frac{\partial y_{k_I}}{\partial w_{pqI}} \end{bmatrix}_n, \quad (2.57)$$

sendo que o subscrito n indica a dependência das variáveis com o índice de tempo. Partindo dessas definições, demonstra-se no Apêndice E que a equação recursiva para o cálculo das sensibilidades é da forma

$$\begin{aligned} &\begin{bmatrix} P_{RRpq}^k & P_{RIpq}^k \\ P_{IRpq}^k & P_{IIpq}^k \end{bmatrix}_{n+1} = \begin{bmatrix} \varphi'(v_{k_R}) & 0 \\ 0 & \varphi'(v_{k_I}) \end{bmatrix}_{n+1} \times \\ &\times \left(\sum_{l=1}^N \begin{bmatrix} w_{klR} & -w_{klI} \\ w_{klI} & w_{klR} \end{bmatrix}_n \begin{bmatrix} P_{RRij}^l & P_{RIpq}^l \\ P_{IRpq}^l & P_{IIpq}^l \end{bmatrix}_n + \delta_{pk} \begin{bmatrix} u_{qR} & -u_{qI} \\ u_{qI} & u_{qR} \end{bmatrix}_n \right) \end{aligned} \quad (2.58)$$

e a atualização dos pesos deve ser feita utilizando-se a seguinte equação

$$\begin{aligned} w_{pq}(n+1) &= w_{pq}(n) + \mu \nabla w_{pq} Q(n) = \\ &= w_{pq}(n) + \mu \sum_{k=1}^N \left([e_{kR} \ e_{kI}]_n \begin{bmatrix} P_{RRpq}^k & P_{RIpq}^k \\ P_{IRpq}^k & P_{IIpq}^k \end{bmatrix}_n \begin{bmatrix} 1 \\ j \end{bmatrix} \right). \end{aligned} \quad (2.59)$$

As variáveis $e_k(n)$ e $u_q(n)$ estão definidas nas equações (2.47) e (2.52), mas lembrando que agora são complexas, ou seja $e_k(n) = e_{kR}(n) + je_{kI}(n)$ e $u_q(n) = u_{qR}(n) + ju_{qI}(n)$.

Como no caso real, os parâmetros P das equações recursivas (2.58) e (2.59) devem ser inicializados com zero e os pesos com números aleatórios complexos uniformemente distribuídos num intervalo de pequenos valores.

2.6 Equalizadores ótimos

2.6.1 Critério de Bayes

O Critério de Bayes, também conhecido como critério MAP (*Maximum a Posteriori Probability*), consiste na determinação do símbolo a_k que maximiza a probabilidade $P(a_k|\mathbf{u}(n))$ [21]. Usando a regra de Bayes, calcula-se a probabilidade de ter sido transmitido a_k dado que se recebeu $\mathbf{u}(n)$ como

$$P(a_k|\mathbf{u}(n)) = \frac{p(\mathbf{u}(n)|a_k)P(a_k)}{p(\mathbf{u}(n))}, \quad (2.60)$$

sendo $\mathbf{u}(n) = [u(n) \ u(n-1) \ \cdots \ u(n-M+1)]^T$. O cálculo dessa probabilidade deve ser feito para todos os símbolos a_k , $k = 1, 2, \dots, K$, sendo K o número de símbolos do alfabeto. Feitos estes cálculos, decide-se pelo símbolo a_k , $k = 1, 2, \dots, K$, cuja probabilidade $P(a_k|\mathbf{u}(n))$ é maior.

O denominador da Equação (2.60) pode ser desconsiderado pois independe de a_k e não influi na decisão do símbolo que maximiza a probabilidade *a posteriori*. Além disso, se os sinais transmitidos forem equiprováveis: $P(a_1) = P(a_2) = \cdots = P(a_K) = 1/K$, o cálculo de $\max\{P(a_k|\mathbf{u}(n))\}$ se reduz ao cálculo de $\max\{p(\mathbf{u}(n)|a_k)\}$ e o critério MAP coincide com o critério da máxima verossimilhança (ML) [21].

Para exemplificar, vamos considerar a transmissão de sinais binários $+1$ e -1 através de um canal real que assume a forma $H(z) = h_0 + h_1z^{-1}$ e um equalizador com $M = 2$ entradas. Assim, a cada instante de tempo n , os possíveis vetores de entrada do

equalizador podem ser obtidos pela seguinte equação

$$\begin{bmatrix} u(n) \\ u(n-1) \end{bmatrix} = \begin{bmatrix} h_0 & h_1 & 0 \\ 0 & h_0 & h_1 \end{bmatrix} \begin{bmatrix} a(n) \\ a(n-1) \\ a(n-2) \end{bmatrix} + \begin{bmatrix} \eta(n) \\ \eta(n-1) \end{bmatrix}, \quad (2.61)$$

que pode ser rescrita de forma mais compacta como

$$\mathbf{u}(n) = \mathbf{H}\mathbf{a}(n) + \boldsymbol{\eta}(n) = \mathbf{x}(n) + \boldsymbol{\eta}(n). \quad (2.62)$$

Generalizando, para o caso em que o vetor $\mathbf{u}(n)$ possui M elementos e a interferência intersimbólica (ISI) se estende sobre N_h símbolos, a matriz de resposta impulsiva $\mathbf{H}$ possui M linhas e $M + N_h - 1$ colunas e o número máximo de possíveis valores de $\mathbf{x}(n) = \mathbf{H}\mathbf{a}(n)$ é dado por $N_x = K^{M+N_h-1}$, sendo N_h o número de coeficientes do canal $H(z)$ e K o número de símbolos do alfabeto. Neste caso, a matriz de Toeplitz $\mathbf{H}$ que descreve o canal tem a seguinte forma

$$\mathbf{H} = \begin{bmatrix} h_0 & h_1 & h_2 & \cdots & h_{N_h-1} & 0 & \cdots & 0 \\ 0 & h_0 & h_1 & \cdots & h_{N_h-2} & h_{N_h-1} & \cdots & 0 \\ \vdots & & & & \vdots & & \vdots & \\ 0 & 0 & 0 & \cdots & 0 & h_0 & \cdots & h_{N_h-1} \end{bmatrix}.$$

Para o exemplo considerado ($M = 2$, $N_h = 2$, $K = 2$), o número de possíveis valores do vetor $\mathbf{x}(n)$ é $N_x = 2^3 = 8$. Esses N_x vetores são denominados de estados desejados do canal. Considerando $\boldsymbol{\eta}(n)$ ruído branco, gaussiano, de média nula e variância σ_η^2 , o sinal $\mathbf{u}(n) = \mathbf{x}(n) + \boldsymbol{\eta}(n)$, por sua vez, consiste de N_x nuvens de pontos (agrupamentos) cujas médias são dadas pelos N_x possíveis valores de $\mathbf{x}(n)$. Os estados desejados do canal $H(z) = 0,5 + 1,0z^{-1}$ para $M = 2$ são mostrados na Tabela 2.16.

Considerando um atraso de τ_d amostras, para se fazer a detecção do sinal transmitido, os estados desejados $\mathbf{x}(n)$, podem ser divididos em K classes dependendo do valor de $a(n - \tau_d)$. Deve-se notar que neste caso, o equalizador faz o papel de classificador.

$a(n)$	$a(n-1)$	$a(n-2)$	$x(n)$	$x(n-1)$
1	1	1	1.5	1.5
1	1	-1	1.5	-0.5
-1	1	1	0.5	1.5
-1	1	-1	0.5	-0.5
1	-1	1	-0.5	0.5
1	-1	-1	-0.5	-1.5
-1	-1	1	-1.5	0.5
-1	-1	-1	-1.5	-1.5

Tabela 2.16: Entrada e estados desejados do canal $H(z) = 0,5 + 1,0z^{-1}$, $M = 2$.

Para o caso binário, pode-se escrever

$$X_{M,\tau_d}^+ = \{\mathbf{x}(n) | a(n - \tau_d) = 1\}, \quad (2.63)$$

$$X_{M,\tau_d}^- = \{\mathbf{x}(n) | a(n - \tau_d) = -1\}. \quad (2.64)$$

No nosso exemplo, os símbolos são considerados equiprováveis, ou seja, cada estado desejado $\mathbf{x}_i^+ \in X_{M,\tau_d}^+$ ou $\mathbf{x}_i^- \in X_{M,\tau_d}^-$ tem probabilidade p_i , igual a $1/N_x$. Os números de estados em X_{M,τ_d}^+ e X_{M,τ_d}^- são denotados respectivamente por N_x^+ e N_x^- , sendo $N_x = N_x^+ + N_x^-$.

Devido ao ruído branco, aditivo, gaussiano e de média nula, o vetor de observação $\mathbf{u}(n)$ é um processo aleatório que possui uma função densidade de probabilidade condicional gaussiana centrada em cada estado desejado $\mathbf{x}$ do canal, sendo que as observações do canal formam agrupamentos cujas médias são estes estados. Determinar o valor do símbolo transmitido $a(n - \tau_d)$ baseado no vetor de observação $\mathbf{u}(n)$ pode ser considerado como um problema de determinação de regiões de decisão no espaço de estados. A teoria de decisão de Bayes proporciona a solução ótima ao problema de decisão e pode ser aplicada na determinação da solução ótima para o equalizador da Figura 2.1 da página 10 [15].

Para o exemplo considerado, é possível verificar que a solução ótima bayesiana é

definida como

$$\hat{a}(n - \tau_d) = \text{sgn}(f_B(\mathbf{u}(n))) = \begin{cases} 1, & f_B(\mathbf{u}(n)) \geq 0 \\ -1, & f_B(\mathbf{u}(n)) < 0, \end{cases} \quad (2.65)$$

sendo que a função $f_B(\cdot)$ é dada por

$$\begin{aligned} f_B(\mathbf{u}(n)) = & \sum_{i=1}^{N_x^+} p_i (2\pi\sigma_\eta^2)^{M/2} \exp\left(-\|\mathbf{u}(n) - \mathbf{x}_i^+\|^2 / 2\sigma_\eta^2\right) - \\ & - \sum_{j=1}^{N_x^-} p_j (2\pi\sigma_\eta^2)^{M/2} \exp\left(-\|\mathbf{u}(n) - \mathbf{x}_j^-\|^2 / 2\sigma_\eta^2\right). \end{aligned} \quad (2.66)$$

Nesta equação, a primeira somatória é relativa à $\mathbf{x}_i^+ \in X_{M,\tau_d}^+$, a segunda à $\mathbf{x}_j^- \in X_{M,\tau_d}^-$ e σ_η^2 é a variância do ruído.

O equalizador ótimo depende tanto da distribuição de ruído quanto dos estados desejados do canal. Multiplicando $f_B(\mathbf{u}(n))$ por uma constante, não há mudança na determinação da solução ótima. Dessa forma, o espaço de estados deve ser dividido em duas regiões limitadas por

$$\{\mathbf{u} | f_B(\mathbf{u}) = 0\}. \quad (2.67)$$

Estas regiões correspondem às duas decisões $a(k - \tau_d) = \pm 1$. A função de decisão bayesiana (Equação (2.66)) é claramente não-linear o que faz com que um equalizador linear não seja uma solução ótima na maioria dos casos [15],[22].

Considerando símbolos equiprováveis, a Equação (2.66) se reduz a

$$f_B(\mathbf{u}(n)) = \sum_{i=1}^{N_x^+} \exp\left(-\|\mathbf{u}(n) - \mathbf{x}_i^+\|^2 / 2\sigma_\eta^2\right) - \sum_{j=1}^{N_x^-} \exp\left(-\|\mathbf{u}(n) - \mathbf{x}_j^-\|^2 / 2\sigma_\eta^2\right). \quad (2.68)$$

Dado o sinal $\mathbf{u}(n)$, num caso em que o alfabeto possui K símbolos, deve-se dividir os estados desejados em K classes, calcular a probabilidade condicional de cada uma separadamente e decidir pela classe cuja probabilidade é maior.

Voltando novamente ao exemplo da Tabela 2.16, pode-se observar na Figura 2.9 os estados $X_{2,1}^+$ e $X_{2,1}^-$ representados por $*$ e $\circ$ respectivamente. Para uma relação

sinal-ruído $\text{SNR} = 15\text{dB}$, 1000 amostras de $\mathbf{u}(n)$, representadas por pontos, podem ser vistas nos gráficos da Figura 2.9. Pode-se verificar que as observações formam agrupamentos ao redor dos estados desejados. A curva que separa o espaço de estados em duas regiões, quando se considera atraso de 2 amostras ($\tau_d = 2$), foi calculada usando a Equação (2.68) e pode ser vista na Figura 2.9-a. Quando $\mathbf{u}(n)$ cai acima da curva, toma-se $\hat{a}(n-2) = 1$, caso contrário faz-se $\hat{a}(n-2) = -1$.

Uma variação do exemplo anterior é quando se considera atraso de uma amostra ($\tau_d = 1$) ou atraso nulo ($\tau_d = 0$). Nestas situações as regiões associadas a $+1$ e -1 se alteram, como pode ser visto na Figura 2.9-b e c, considerando uma relação sinal-ruído de 15dB .

Quando se utiliza um equalizador linear, observa-se que há uma grande importância na escolha do atraso τ_d , ou seja, para um certo atraso, o equalizador funciona melhor que para outros. Isso pode ser observado neste exemplo simples. Quando $\tau_d = 2$, um equalizador linear é capaz de separar as classes -1 e $+1$ facilmente, embora não seja a solução ótima. Já com $\tau_d = 1$, a solução obtida pelo LTE se afasta ainda mais da ótima e finalmente com $\tau_d = 0$, uma única reta não é capaz de separar as classes e conseqüentemente de solucionar o problema de equalização. Deve-se notar que as redes não-lineares também terão mais dificuldade de obter a curva de separação das regiões de decisão para a situação de atraso nulo que para situação de atraso $\tau_d = 1$ ou $\tau_d = 2$. Isso porque o número de vizinhos pertencentes a classes diferentes aumenta quando se diminui o atraso τ_d neste exemplo.

Figura 2.9: Estados desejados, agrupamentos e curva ótima de separação das regiões de decisão segundo Bayes para $H(z) = 0,5 + 1,0z^{-1}$, $M = 2$, $\text{SNR} = 15\text{dB}$, 1000 amostras de $\mathbf{u}(n)$ e a) $\tau_d = 2$, b) $\tau_d = 1$, c) $\tau_d = 0$.

2.6.2 Critério da máxima verossimilhança

Forney em [23] demonstrou que o equalizador ótimo é formado por um filtro casado branqueador, seguido do algoritmo de Viterbi [24]. Este equalizador é não-linear e busca a seqüência de máxima verossimilhança. Na Figura 2.10 é mostrado um esquema de tal equalizador sendo que $a(n)$ representa os símbolos transmitidos, $\eta(n)$ representa um ruído branco, aditivo e gaussiano e $u(n)$ o sinal que chega no receptor. O fato de se usar o filtro casado e de se estimar a seqüência (de símbolos) de máxima verossimilhança, faz com que este equalizador ótimo difira do critério de Bayes (MAP), mesmo quando os símbolos são equiprováveis.

Figura 2.10: Esquema do equalizador ótimo segundo Forney.

Segundo Forney, esta solução apresenta excelentes resultados e é tão boa quanto as obtidas por outros estimadores ótimos de seqüências que utilizam outros critérios. O uso do filtro branqueador está ligado ao funcionamento do algoritmo de Viterbi, pois o ruído da seqüência de entrada desse algoritmo deve ser não-correlacionado. Ungerboeck em [25] propôs uma modificação no algoritmo de Viterbi, permitindo o uso apenas de um filtro casado com o canal. No projeto desses equalizadores ótimos, no entanto, o conhecimento do canal se faz necessário. Em [26], foi utilizado um algoritmo adaptativo para estimação do canal e se observou que o resultado da equalização se afastava um pouco do ótimo. Nosso objetivo aqui, é projetar o equalizador ótimo apresentado em [23] para comparar os resultados obtidos com os dos equalizadores não-lineares baseados em redes neurais que não pressupõem o conhecimento do canal.

Supondo que a função de transferência $H(z)$ de um canal FIR seja conhecida, então

$R(z)$, a transformada-Z da sequência de autocorrelação da correspondente resposta ao pulso unitário $h(n)$, pode ser fatorada como

$$R(z) = H(z)H^*(1/z^*) = F(z)F^*(1/z^*),$$

sendo $F^*(1/z^*)$ a parcela de fase mínima e $F(z)$ a de fase não-mínima. Neste contexto, $R(z)$ representa a cascata do canal com o filtro casado. Para que o ruído da sequência de saída do filtro casado seja não correlacionado deve-se utilizar ainda um filtro branqueador. Considerando agora que $W(z^{-1})$ seja a cascata do filtro casado com o filtro branqueador, o sinal na saída desse filtro será

$$s(n) = W(z^{-1})H(z)a(n) + W(z^{-1})\eta(n). \quad (2.69)$$

Para esclarecer a notação utilizada acima, vamos supor que $H(z) = h_0 + h_1z^{-1}$, então $H(z)a(n) = h_0a(n) + h_1a(n-1)$.

O problema a ser resolvido está na escolha correta do filtro casado branqueador $W(z^{-1})$. Pode-se então considerar as seguintes escolhas:

- $W(z^{-1}) = H^*(1/z^*)/F(z)$. Neste caso obtém-se um filtro passa-tudo não-causal que pode ser aproximado por um sistema causal considerando um certo atraso. O sinal na saída desse filtro será: $s(n) = F^*(1/z^*)a(n) + W(z^{-1})\eta(n)$. Deve-se notar que $W(z)W(z^{-1}) = 1$ e o canal equivalente tem função de transferência $F^*(1/z^*)$ que é de mínima fase;
- $W(z^{-1}) = H^*(z^{*-1})/F^*(1/z^*)$. Neste caso obtém-se um filtro causal e o sinal em sua saída será: $s(n) = F(z)a(n) + W(z^{-1})\eta(n)$. Novamente $W(z)W(z^{-1}) = 1$ e o canal equivalente tem função de transferência $F(z)$ que é de fase não-mínima.

Para nossa finalidade que é a comparação de desempenho com o equalizador ótimo proposto por Forney, basta tomarmos como modelo a versão de fase mínima do canal ($F^*(1/z^*)$), o ruído aditivo branco e a implementação do algoritmo de Viterbi que será discutido a seguir.

O algoritmo de Viterbi (AV) foi inicialmente proposto como uma solução para decodificação de códigos convolucionais por A. J. Viterbi em 1967 [27]. Posteriormente, Omura e Forney [28], [29] mostraram que este algoritmo pode ser considerado como um decodificador de máxima verossimilhança. O AV é freqüentemente visto como um algoritmo que minimiza a probabilidade de erro através da comparação de um conjunto de verossimilhanças de possíveis estados de transição que podem ocorrer e decide qual deles possui a maior probabilidade de ocorrência. Na Figura 2.11 é mostrada uma possível implementação do algoritmo de Viterbi considerando um canal de três coeficientes [30]. Cada amostra do sinal $s(n)$ é comparada com as possíveis amostras de saída do filtro casado branqueador sem ruído ($s_{000}^o, \dots, s_{111}^o$). A partir dessa comparação, calcula-se a probabilidade de ocorrência de cada símbolo possível, considerando aquele de maior probabilidade. A realimentação existente na implementação deste algoritmo leva em conta as probabilidades dos estados de amostras anteriores para buscar a seqüência de máxima verossimilhança. A função $f_n(\cdot)$ depende da distribuição do ruído $\eta(n)$. Neste caso, como o ruído é branco e gaussiano, a função $f_n(\cdot)$ é uma gaussiana com média zero e variância igual à do ruído.

A implementação do algoritmo de Viterbi da Figura 2.11 não é a mais eficiente. No entanto, como se quer usar este equalizador apenas para comparação com os resultados de outros equalizadores, não se levou em conta este aspecto. Uma implementação mais eficiente do AV pode ser encontrada em [24].

No caso de constelações complexas, é fácil estender o esquema da Figura 2.11. Para 4-QAM, por exemplo, o alfabeto é formado por quatro símbolos: $\{-1-j, -1+j, +1-j, +1+j\}$. Considerando um canal com 3 coeficientes há 4^3 estados possíveis, que devem ser comparados com um símbolo de saída do filtro casado branqueador. Num primeiro estágio, sobram 16 estados e num segundo 4 dos quais apenas 1 é escolhido por ter a máxima probabilidade de ocorrência. Considerando um canal com mais coeficientes, existirão mais estágios até que sobre um único símbolo de maior probabilidade. No caso de outras constelações, o número de estados possíveis muda e o

Figura 2.11: Implementação do equalizador ótimo segundo Forney para um canal com três coeficientes e transmissão binária.

penúltimo estágio sempre corresponde ao tamanho do alfabeto.

2.7 *Decision Feedback Equalizer (DFE)*

O DFE baseado em filtragem adaptativa linear é formado por dois filtros adaptativos: um progressivo $H_f(z)$ e outro regressivo $H_b(z)$ como pode ser visto na Figura 2.12.

O uso do DFE surgiu com o intuito de se combater parte da interferência entre símbolos (ISI) utilizando-se decisões já tomadas [31]. Para entender isso de forma mais sistemática, vamos voltar para o modelo de transmissão já abordado anteriormente.

Deseja-se transmitir um símbolo $a(n)$ através de um canal com atraso n_0 , cuja resposta ao pulso unitário é mostrada na Figura 2.13. Considerando $\eta(n)$ um ruído

Figura 2.12: *Decision Feedback Equalizer*(DFE)

aditivo, branco e gaussiano, o sinal que chega no receptor é dado por

$$\begin{aligned}
 u(n) &= \sum_k h(k)a(n-k) + \eta(n) = \\
 &= h(n_0)a(n-n_0) + \sum_{k=0}^{n_0-1} h(k)a(n-k) + \sum_{k=n_0+1}^{+\infty} h(k)a(n-k) + \eta(n). \quad (2.70)
 \end{aligned}$$

Considerando $h(n_0)$ grande o suficiente, o termo $h(n_0)a(n-n_0)$ é o sinal desejado usado para estimar o nível de amplitude transmitida. As somatórias constituem a interferência entre os símbolos vizinhos e $\eta(n)$ é o ruído aditivo. Cada termo de interferência é proporcional à correspondente amostra da resposta ao pulso unitário do canal $h(k)$ como mostrado na Figura 2.13. A interferência intersimbólica é zero se, e somente se, $h(k) = 0, k \neq n_0$.

Figura 2.13: Resposta ao pulso unitário do canal.

A idéia do DFE é usar o filtro transversal regressivo para estimar a interferência causada pelos símbolos vizinhos anteriores ao estimado $a(n - n_0)$. Dessa forma, se os coeficientes do filtro regressivo forem iguais a $h(n_0 + p)$, $p > 0$ e além disso, se as decisões tomadas anteriormente estiverem corretas, consegue-se obter uma boa estimativa da segunda somatória da Equação (2.70), que pode então ser eliminada facilmente. Por outro lado, o filtro progressivo deve tentar eliminar a interferência provocada pelos símbolos posteriores ao símbolo estimado (primeira somatória da Equação (2.70)).

À primeira vista, o bom funcionamento do DFE depende das decisões já tomadas, ou seja quanto mais corretas forem as decisões anteriores, melhor será a equalização. Poder-se-ia pensar então que o DFE não ocasiona melhoras, pois se as decisões já tomadas são corretas, o equalizador baseado no filtro progressivo seria bom o suficiente para proporcionar o desempenho devido. Isso não é verdade, posto que há contribuição do filtro regressivo na tomada das decisões e além disso, com uma boa eliminação da ISI, o erro na estimativa do símbolo transmitido é menor, causando assim uma melhora significativa na equalização.

Nesta dissertação, utilizou-se o esquema do DFE apresentado na Figura 2.14 com uma extensão para filtragem não-linear [32]. Uma interpretação do DFE como classificador será dada posteriormente.

Considerou-se M_f o número de entradas relacionadas com as amostras da saída do canal e $M_b = M - M_f$ o número de entradas que são amostras realimentadas das decisões já tomadas.

2.7.1 DFE visto como classificador

A interpretação de um equalizador como um classificador permite entender porque ocorrem melhoras com o uso da realimentação de decisões. Para exemplificar, será considerado um canal com função de transferência $H(z) = 0,5 + z^{-1}$ e um equalizador com duas entradas ($M = 2$). Supondo transmissão binária sem ruído, as possíveis entradas do equalizador $x(n)$ e $x(n - 1)$, chamadas aqui de estados desejados, são

Figura 2.14: Esquema empregado para o DFE.

calculadas através das combinações de $a(n)$, $a(n-1)$ e $a(n-2)$. Na Tabela 2.16 da página 48 estão listadas todas as possíveis entradas do equalizador na situação de ruído nulo.

Considerando um equalizador de 1ª ordem ($M = 2$), SNR=10 dB e o canal da Tabela 2.16, obtém-se a curva de separação do equalizador ótimo segundo Bayes [14] mostrada na Figura 2.15-a para atraso nulo ($\tau_d = 0$) e na Figura 2.16-a para atraso de uma amostra ($\tau_d = 1$). Nota-se nestas figuras que a curva ótima de separação segundo Bayes é não-linear.

Supondo que a decisão $\hat{a}(n-1)$ esteja correta, pode-se utilizá-la juntamente com $x(n)$, $x(n-1)$ para estimar $a(n)$ obtendo assim, as regiões de separação das figuras 2.15-b e 2.15-c que se referem ao canal $H(z) = 0,5 + z^{-1}$ com atraso nulo. Utilizando agora a decisão $\hat{a}(n-2)$ e os estados $x(n)$, $x(n-1)$ para estimar $a(n-1)$, obtém-se as regiões de separação das figuras 2.16-b e 2.16-c que também se referem ao canal $H(z) = 0,5 + z^{-1}$ mas com atraso de uma amostra.

Observando estas figuras, conclui-se que o DFE garante um melhor desempenho

na classificação devido à separação dos estados desejados do canal $H(z) = 0,5 + z^{-1}$ em dois planos distintos, conforme a decisão passada. Dessa forma, no caso em que se considerou atraso nulo sem o DFE, um equalizador linear transversal não consegue proporcionar uma separação coerente das regiões de decisão. Por outro lado, fazendo uso da decisão passada $\hat{a}(n-1)$, ainda no caso de atraso nulo, o equalizador linear passa a ser ótimo para uma relação sinal-ruído de 10 dB. No caso do atraso de uma amostra, a divisão das regiões de decisão com o uso do DFE, é mais simples que na situação em que ele não é usado. Deve-se notar ainda nestas figuras as regiões de erro quando se usa uma decisão anterior errada (áreas hachuradas). Neste caso, a probabilidade de erro é de 50% no caso de atraso nulo e de aproximadamente 25% no caso de atraso de uma amostra.

Figura 2.15: Estados desejados; — curva ótima de separação das regiões de decisão, 2-PAM, SNR=10dB, $\tau_d = 0$, $H(z) = 0,5 + z^{-1}$, a) Sem o uso do DF, $M = 2$; b) Com o uso do DF, $M_f = 2$, $M_b = 1$, $\hat{a}(n-1) = 1$; c) Com o uso do DF, $M_f = 2$, $M_b = 1$, $\hat{a}(n-1) = -1$; regiões de erro causadas por decisões anteriores erradas estão hachuradas.

Figura 2.16: Estados desejados; — curva ótima de separação das regiões de decisão, 2-PAM, SNR=10dB, $\tau_d = 1$, $H(z) = 0,5 + z^{-1}$, a) Sem o uso do DF, $M = 2$; b) Com o uso do DF, $M_f = 2$, $M_b = 1$, $\hat{a}(n-2) = 1$; c) Com o uso do DF, $M_f = 2$, $M_b = 1$, $\hat{a}(n-2) = -1$; regiões de erro causadas por decisões anteriores erradas estão hachuradas.

2.8 Equalização e classificação

A equalização de canais de comunicação pode ser considerada como um problema de desconvolução e o equalizador é projetado para que a combinação da sua resposta ao pulso unitário com a do canal esteja o mais próximo possível de um simples atraso. No entanto, essa aproximação se torna bastante complicada quando há não-linearidades no canal. Por outro lado, a equalização pode ser encarada como um problema de classificação e o equalizador é projetado como um dispositivo de decisão a fim de reconstruir a sequência de símbolos transmitida da forma mais precisa possível [33].

Um equalizador linear clássico faz o cálculo de uma projeção linear $\mathbf{y}(n) = \mathbf{u}^T(n) \cdot \mathbf{w}$, sendo $\mathbf{w}$ o vetor de coeficientes do equalizador. Neste caso, procura-se a melhor projeção das amostras de $\mathbf{u}(n)$ na direção de $\mathbf{w}$. No sentido de classificação, isso corresponde a encontrar o discriminante linear de Fisher $\mathbf{w}$ [34], [35], [36]. A Figura 2.17 ilustra duas possíveis direções de $\mathbf{w}$. Nesta figura, o vetor da esquerda é claramente melhor que o outro [35]. Neste caso, o filtro linear e o decisor constituem um dispositivo de classificação que tenta fazer uma separação linear entre as classes correspondentes aos níveis +1 (*) ou -1 (o).

Figura 2.17: Projeção das amostras (estados) em duas diferentes direções de $\mathbf{w}$, para $M = 2$.

Encontrar o discriminante linear de Fischer é um critério ótimo no sentido de se obter o vetor de pesos que satisfaz $\mathbf{w} = \mathbf{S}_w^{-1}(\mathbf{m}_1 - \mathbf{m}_2)$, sendo $\mathbf{S}_w$ a matriz de dispersão intra-classes e $\mathbf{m}_1$ e $\mathbf{m}_2$ os centros de massa de cada classe [34]. Neste caso, considerou-se a existência apenas de duas classes. As definições da matriz $\mathbf{S}_w$ e dos vetores $\mathbf{m}_1$ e $\mathbf{m}_2$, bem como a generalização desse critério para mais de duas classes podem ser encontradas na referência [34].

Por outro lado, considerando a equalização como um problema de desconvolução, existe um outro critério ótimo que é a solução de Wiener em que $\mathbf{w} = \mathbf{R}^{-1}\mathbf{p}$, sendo $\mathbf{R}$ a matriz de autocorrelação do sinal $\mathbf{u}$ e $\mathbf{p}$ um vetor de correlação cruzada entre o sinal $\mathbf{u}$ e o sinal desejado. Maiores detalhes das definições da matriz $\mathbf{R}$ e do vetor $\mathbf{p}$ podem ser encontrados na referência [1]. A solução obtida pelo equalizador linear transversal através dos algoritmos LMS, RLS ou equivalentes tende à esta solução.

Tanto o critério do discriminante de Fischer quanto a solução de Wiener são critérios lineares ótimos. O primeiro pode ser usado quando se interpreta o LTE como um classificador, o segundo quando a equalização linear é tratada como um problema de desconvolução e se deseja minimizar um erro quadrático.

As redes neurais, por sua vez, têm uma grande capacidade de classificação. É sob este enfoque que muitos autores têm utilizado essas estruturas aplicadas à equalização de canais de comunicação. A rede MLP é uma das redes mais usadas em comunicações digitais. Pode ser demonstrado que uma rede MLP de duas camadas com função de ativação sigmoidal e um grande número de neurônios pode aproximar arbitrariamente bem funções contínuas [37]. Por outro lado, uma rede RBF tem um desempenho intermediário entre o equalizador de Wiener e o equalizador bayesiano [36].

Encarando o problema de equalização como classificação, vamos exemplificar considerando novamente o canal $H(z) = 0,5 + z^{-1}$ cuja solução ótima segundo o critério de Bayes está mostrada na Seção 2.6.1 para três atrasos diferentes: $\tau_d = 0$, $\tau_d = 1$ e $\tau_d = 2$ e transmissão binária. Aplicando o LTE e uma rede MLP com 4 camadas, ambos com duas entradas ($M = 2$) e atraso $\tau_d = 1$, obtém-se os resultados expostos

nas figuras 2.18-a e b respectivamente. Deve-se notar que para efeito de comparação, a curva de separação ótima segundo Bayes foi traçada nestes gráficos. Nessas condições a rede MLP se aproxima muito do equalizador linear, o que é algo curioso. Mudando a configuração de (2,9,3,1) para (2,15,5,1) ocorre uma mudança de inclinação da “reta” de separação das regiões de decisão. Neste caso, a solução ótima obtida pelo discriminante linear de Fisher é coincidente com a obtida pelo LTE e foi omitida na Figura 2.18-a.

Na Figura 2.18-c podem ser vistas curvas de separação das regiões de decisão obtidas por uma rede RBF, considerando atraso de 1 amostra ($\tau_d = 1$). Deve-se notar que tais curvas estão bem próximas da curva ótima segundo Bayes. A curva $(- \cdot -)$ foi obtida utilizando-se uma rede RBF com 8 neurônios cujas funções gaussianas tinham médias iguais aos estados desejados e variância igual a 0.5 (metade da menor distância entre estados $X_{2,1}^+$ e $X_{2,1}^-$ elevada ao quadrado). Já a curva $(\cdot \cdot \cdot)$ foi obtida também com 8 neurônios, mas com as médias das funções gaussianas baseadas em amostras do sinal na entrada do equalizador e variância igual à deste sinal.

Considerando atraso nulo ($\tau_d = 0$), obteve-se a curva de separação ótima (segundo Bayes) da Figura 2.9-c. Treinando uma rede MLP com a configuração (2,9,3,1), obteve-se o resultado da Figura 2.19-a. É interessante notar que mesmo com uma taxa de erro alta e se afastando da curva ótima, o equalizador MLP conseguiu dividir o espaço de estados, tarefa que um equalizador linear não consegue executar. Treinando uma rede RBF, com $M=2$, 8 neurônios na camada oculta e funções gaussianas de médias iguais aos estados desejados e $\sigma^2 = 0.5$, chega-se, para $\tau_d = 0$, no resultado da Figura 2.19-b. A curva de decisão obtida se aproxima bastante da ótima segundo Bayes nessa situação.

Diante dos resultados, pôde-se observar que conhecidos os estados desejados do canal e usando todos eles como centros das funções gaussianas da RBF, obtém-se um desempenho bem próximo do ótimo segundo Bayes. Por outro lado, no caso de $\tau_d = 1$, se fossem usados dois centros baseados nas médias dos estados desejados, a RBF teria um comportamento bem próximo do LTE, o que confirma os resultados apresentados

na referência [36]. Este resultado está de acordo com a literatura em que é reportado que o desempenho da rede RBF está intimamente ligado à quantidade de centros das funções de base radial e à sua correta determinação. Na prática, como não se conhece o canal, se utilizam técnicas de agrupamento para determinação dos centros que se baseiam na distribuição do sinal de entrada do equalizador.

É interessante observar ainda as curvas de separação obtidas pela rede MLP nas figuras 2.18-b e 2.19-a. Em ambos os casos, o comportamento desta rede se afastou do ótimo segundo Bayes. No entanto, devido à grande capacidade na aproximação de funções arbitrárias da rede MLP, talvez seja possível obter uma solução bem próxima da ótima segundo Bayes mudando a configuração dessa rede e treinando-a de forma que a solução não fique presa em mínimos locais. Nos casos citados, a rede pode ter ficado presa nesses mínimos locais proporcionando soluções sub-ótimas. Embora este seja um problema de interesse, não será investigado nessa dissertação.

Figura 2.18: Curvas de separação das regiões de decisão do canal $H(z) = 0,5 + z^{-1}$, $M = 2$, $\tau_d = 1$, $\text{SNR} = 15\text{dB}$, a) LTE ($-\cdot-$), b) MLP ($-\cdot-$) (2,9,3,1); ($\cdots$) (2,15,5,1), c) RBF, 8 neurônios, ($-\cdot-$) centros iguais aos estados desejados e $\sigma^2 = 0,5$; ($\cdots$) centros iguais ao sinal inicial de entrada da rede e $\sigma^2 = 1,25$.

Figura 2.19: Curva de separação das regiões de decisão do canal $H(z) = 0,5 + z^{-1}$, $M = 2$, $\tau_d = 0$, $\text{SNR} = 15\text{dB}$, a) MLP $(- \cdot -)$ (2,9,3,1), b) RBF $(- \cdot -)$ 8 neurônios com gaussianas centradas nos estados desejados e $\sigma^2 = 0,5$.

2.9 Canais não-lineares

O modelo dos canais não-lineares utilizados neste trabalho é mostrado na Figura 2.20 [18], [38].

A sequência transmitida passa através de um canal linear cuja função de transferência é $H_l(z)$, em seguida é adicionada a não-linearidades harmônicas. Os valores

dos ganhos D_2 , D_3 e D_4 determinam quão severa é a distorção não-linear. Esse modelo de canal não-linear é freqüentemente encontrado em canais de satélite, usados para transmissão de dados, especialmente quando os amplificadores do sinal operam nos limites de ganho alto [18].

Figura 2.20: Modelo de um canal não-linear.

Capítulo 3

Resultados Experimentais

No capítulo anterior foi feita uma descrição detalhada de diversos equalizadores e seus algoritmos de treinamento. Os equalizadores ótimos segundo o critério de Bayes e da máxima verossimilhança também foram descritos.

Como já fora discutido anteriormente a solução “ótima” na equalização de diversos canais é não-linear o que faz com que um equalizador linear como o LTE não tenha um desempenho satisfatório. Dessa forma, as redes neurais e estruturas não-lineares mais simples como o DFE têm sido muito utilizadas nestas situações.

O objetivo desse trabalho é fazer um estudo comparativo dessas diversas estruturas e a partir das conclusões desse estudo, escolher uma ou mais estruturas adequadas e se possível propor alguns melhoramentos. Para isso, foram feitas simulações considerando modelos de canais de ordem baixa, mas que apresentam algumas características importantes que causam problemas em aplicações práticas (zeros sobre a circunferência unitária ou próximos dela, não-linearidades, fase não-mínima). Uma comparação teórica entre os desempenhos das diferentes redes neurais é um problema que apresenta dificuldades que ultrapassam o escopo deste trabalho.

Dessa forma, a filosofia do trabalho é considerar alguns modelos de canais simples, característicos e de baixa complexidade (e assim facilitar as simulações) para realizar inicialmente um estudo comparativo e introduzir um novo algoritmo de treinamento

com uma velocidade de convergência maior e um moderado aumento da complexidade computacional em comparação com os algoritmos de treinamento mais conhecidos.

Realizado este estudo, pretende-se escolher uma ou mais estruturas para tentar propor possíveis melhoramentos e/ou projetar um equalizador híbrido que funcione melhor que cada estrutura separadamente.

3.1 Canais utilizados nas simulações

As funções de transferência dos canais usados nas simulações estão na Tabela 3.1. Nas simulações com sinais binários (2-PAM) foram utilizados os canais $\mathcal{H}_1$, $\mathcal{H}_2$ e $\mathcal{H}_3$. Nos experimentos com sinais complexos (4-QAM), por sua vez, utilizaram-se os canais $\mathcal{H}_{1c}$, $\mathcal{H}_{2c}$ e $\mathcal{H}_{3c}$. Os modelos $\mathcal{H}_2$ e $\mathcal{H}_{2c}$ seguem o esquema da Figura 2.20, em que se separa a parte linear da não-linear. Na Tabela 3.1, $x_l(n)$ é a saída da parte linear e a $x(n)$ soma-se ruído para obter a entrada do equalizador.

É importante lembrar que estes modelos de canais representam não só os canais físicos de transmissão mas também o sistema de transmissão/modulação e o sistema de recepção/demodulação efetivamente presentes em qualquer sistema de comunicações prático. Dessa forma, denominou-se aqui como canal um modelo de tempo discreto para o sistema de transmissão, o canal físico e o sistema de recepção.

$\mathcal{H}_1$ (Linear com fase não-mínima [39]) $\mathcal{H}_1(z) = 0,3482 + 0,8704z^{-1} + 0,3482z^{-2}$ Zeros: $\{-0,5001; -1,9996\}$
$\mathcal{H}_2$ (Não-Linear [38]) Parte não-linear: $x(n) = x_l(n) + 0,2x_l^2(n) - 0,1x_l^3(n)$ Parte linear: $\mathcal{H}_{2l}(z) = \mathcal{H}_1(z)$ Zeros da parte linear: $\{-0,5001; -1,9996\}$
$\mathcal{H}_3$ (Linear com zeros no círculo unitário [18]) $\mathcal{H}_3(z) = 1 - 2z^{-1} + z^{-2}$ Zeros: $\{1;1\}$
$\mathcal{H}_{1c}$ (Linear com fase não-mínima [40]) $\mathcal{H}_{1c}(z) = (0,34 - j0,27) + (0,87 + j0,43)z^{-1} + (0,34 - j0,27)z^{-2}$ Zeros: $\{(-0,8248 - j2,3950); (-0,1285 + j0,3733)\}$ Módulo dos zeros: $\{2,5331; 0,3948\}$
$\mathcal{H}_{2c}$ (Não-Linear [40]) Parte não-linear: $x(n) = x_l(n) + 0,1x_l^2(n) + 0,05x_l^3(n)$ Parte linear: $\mathcal{H}_{2cl}(z) = \mathcal{H}_{1c}(z)$ Zeros da parte linear: $\{(-0,8248 - j2,3950); (-0,1285 + j0,3733)\}$ Módulo dos zeros da parte linear: $\{2,5331; 0,3948\}$
$\mathcal{H}_{3c}$ (Linear com zeros próximos da circunferência unitária [17]) $\mathcal{H}_{3c}(z) = (0,4313 + j0,4311) + (0,3663 - j1,0996)z^{-1} + (-0,5282 + j0,0756)z^{-2}$ Zeros: $\{(0,4996 + j1,0000); (0,3503 + j0,7000)\}$ Módulo dos zeros: $\{1,1179; 0,7827\}$

Tabela 3.1: Modelos dos canais de comunicação usados nas simulações.

3.2 Velocidade de convergência dos algoritmos de treinamento

3.2.1 Redes MLP

Com o intuito de comparar os algoritmos de treinamento de redes do tipo MLP considerou-se inicialmente o canal $\mathcal{H}_1$ (Tabela 3.1), transmissão binária (2-PAM), uma rede MLP de 4 camadas com configuração (5, 9, 3, 1), $M = 5$ entradas (4 atrasadores na entrada da rede) e um atraso de três amostras ($\tau_d = 3$). Para uma relação sinal-ruído (SNR) de 20 dB, curvas de erro quadrático médio (MSE) foram obtidas considerando-se os algoritmos de treinamento BP (*Backpropagation*), AC (Acelerador) e RLS-I (Figura 3.1).

Figura 3.1: Curvas de MSE (média de 100 realizações). (i)BP, $\mu = 0,1$; $\beta = 0,3$. (ii)AC, $\mu = 0,1$; $m_1 = 1,5$; $m_2 = 0,2100$; $m_3 = 0,5291$. (iii)RLS-I, $\lambda = 0,995$; $\delta = 2$.

Na Figura 3.1, pode-se observar que a velocidade de convergência do algoritmo BP é bastante lenta. Na verdade, o que acontece é que este algoritmo é sensível às condições iniciais, o que torna sua convergência dependente das mesmas. Já o algoritmo acelerador e o RLS-I não apresentam este tipo de problema, talvez pelo fato de seus passos de adaptação não serem fixos [8]. Na Figura 3.2, são mostrados os sinais de saída dos equalizadores durante o treinamento com os algoritmos BP, AC e RLS-I para $\text{SNR} = 20$ dB. Nesta Figura, considerou-se uma situação em que houve convergência rápida do algoritmo BP e mesmo assim ela é mais lenta que a convergência dos outros dois algoritmos.

Figura 3.2: Sinais de saída dos equalizadores treinados com os algoritmos BP ($\mu = 0,1$; $\beta = 0,3$), AC($\mu = 0,1$; $m_1 = 1,5$; $m_2 = 0,2100$; $m_3 = 0,5291$) e RLS-I ($\lambda = 0,995$; $\delta = 2$) para $\text{SNR}=20$ dB.

Nestes experimentos o passo de aprendizagem do algoritmo *Backpropagation* foi feito igual ao do Acelerador sendo que os parâmetros deste último foram escolhidos

conforme procedimento descrito na Seção 2.3.4.

Para comparar a velocidade de convergência dos três algoritmos de treinamento considerados, é necessário tomar parâmetros compatíveis. Na falta de uma equivalência teórica entre os algoritmos, os parâmetros utilizados foram obtidos de forma experimental. Para o algoritmo RLS-I, considerou-se $\lambda = 0,995$. Na presença de ruído, há algumas situações em que este algoritmo diverge para valores de fator de esquecimento menores. Para o algoritmo BP, considerou-se $\mu = 0,1$, já que valores maiores do passo de aprendizagem levavam o algoritmo mais próximo de situações de divergência e não se observaram modificações significativas no resultado apresentado na Figura 3.1. Isto se deve a sua maior sensibilidade às condições iniciais e ao fato de na Figura 3.1 ser apresentada a média de 100 realizações. Para o algoritmo AC, considerou-se $\mu = 0,1$, de tal forma que o passo de adaptação desse algoritmo fosse o mesmo que o do algoritmo BP. Os parâmetros m_1, m_2 e m_3 foram escolhidos para satisfazer a condição $4m_1^2m_2m_3 = 1$ e evitar uma excessiva variância do erro médio quadrático.

É interessante observar que mesmo com um fator de esquecimento (λ) maior que $1 - \mu$, a velocidade de convergência do algoritmo RLS-I resultou maior que a dos algoritmos BP ou AC.

A Tabela 3.2 mostra o número de operações por iteração dos três algoritmos considerados e do algoritmo RLS-II, para a rede MLP utilizada nos experimentos (com configuração (5, 9, 3, 1)).

Operações	BP	AC	RLS-I	RLS-II
$\times$	465	886	3858	2620
$\div$	-	18	18	4
+	395	649	1917	2258
NL	36	36	36	36

Tabela 3.2: Número de operações por iteração de cada algoritmo de treinamento para uma rede MLP (5,9,3,1).

Figura 3.3: Curvas de MSE (média de 100 realizações). (i)RLS-I, (ii)RLS-II, $\lambda = 0,995$; $\delta = 2$.

Como pode ser observado na Tabela 3.2, o algoritmo Acelerador apresenta um moderado aumento da complexidade computacional em relação ao algoritmo *Backpropagation*. Já o número de multiplicações e somas do algoritmo RLS-I são bem maiores, respectivamente 4,4 vezes e 3,0 vezes que os correspondentes valores do algoritmo AC.

Deve-se notar ainda nesta Tabela, que o algoritmo RLS-II possui uma complexidade cerca de 1,5 vezes menor que o RLS-I em número de multiplicações e 4,5 vezes em número de divisões, podendo a princípio ser considerado como uma boa alternativa para o treinamento. Por outro lado, observando a Figura 3.3, nota-se que embora o algoritmo RLS-II seja mais atraente sob o ponto de vista computacional, ele perde no que diz respeito à qualidade da estimativa do símbolo transmitido, pois o erro médio quadrático é maior que o do RLS-I. Comparando as figuras 3.1 e 3.3, nota-se que a

velocidade de convergência do RLS-II é menor que a do algoritmo Acelerador e como este último possui uma complexidade menor, sua utilização se torna mais interessante.

Diante destas observações o uso de algoritmos baseados no método dos mínimos quadrados recursivo para treinamento de redes do tipo MLP, praticamente não se justifica, embora apresente uma maior velocidade de convergência. Por outro lado, o algoritmo Acelerador mostra ser uma boa alternativa no treinamento dessas redes.

As conclusões anteriores se referem ao caso binário em que foram utilizadas versões dos algoritmos para sinais reais. As versões para sinais complexos dos algoritmos de treinamento têm uma complexidade computacional bem maior, particularmente nos algoritmos do tipo RLS onde o aumento de complexidade em relação ao caso real é brutal. Isso nos levou à não dedução dos algoritmos baseados no método dos mínimos quadrados para sinais complexos. O algoritmo Acelerador, por outro lado, apresentou vantagens quando utilizado no caso real e sua versão complexa (CAC) foi obtida e está descrita na Seção 2.3.4 e no Apêndice D.

A fim de comparar a velocidade de convergência do algoritmo Acelerador complexo com o algoritmo *Backpropagation* complexo (CBP), considerou-se a modulação 4-QAM, o canal $\mathcal{H}_{1c}$, uma rede MLP de três camadas com a configuração (5, 10, 1), $M = 5$ entradas e um atraso $\tau_d = 3$. Para uma relação sinal-ruído de 20dB, curvas de erro quadrático médio foram obtidas (Figura 3.4).

Na Figura 3.4, verifica-se que o algoritmo CAC possui uma velocidade de convergência maior que o algoritmo CBP com *momentum*. No entanto, como pode ser visto no Apêndice D, o algoritmo CAC tem muito mais operações que o algoritmo CBP, o que de certa forma pode tornar seu uso inviável. Mesmo o algoritmo em sua versão não-matricial exige um grande esforço computacional pois é necessário fazer o cálculo de pelo menos 18 escalares por iteração (veja programa para MatLab apresentado no Apêndice D).

Figura 3.4: Curvas de MSE (média de 100 realizações), MLP (5,10,1), canal $\mathcal{H}_{1c}$, $M = 5$, $\tau_d = 3$, (i) CBP ($\mu = 0,1$; $\beta = 0,3$), (ii) CAC ($\mu = 0,1$; $m_1 = 1,5$; $m_2 = 0,21$; $m_3 = 0,53$).

3.2.2 Redes RBF

O treinamento de redes do tipo RBF aplicadas à equalização, geralmente é feito com um algoritmo adaptativo para atualização dos pesos e um algoritmo para adaptação dos centros e variâncias, como foi explicado na Seção 2.4.1. No caso da adaptação dos pesos, pode-se utilizar algoritmos do tipo gradiente estocástico ou eventualmente dos mínimos quadrados recursivo. No entanto, observando a Tabela 2.12 da página 40 verifica-se que a elevada complexidade computacional do algoritmo RLS+UCL torna seu uso inviável.

Dessa forma, uma alternativa é usar o algoritmo acelerador para atualização dos pesos por apresentar um comportamento intermediário entre o LMS e RLS tanto do

ponto de vista de esforço computacional, como de velocidade de convergência.

Para verificar o comportamento desses algoritmos, considerou-se o canal $\mathcal{H}_1$, $N = 64$ neurônios na camada oculta, $M = 5$ entradas e um atraso de $\tau_d = 3$ amostras para transmissão binária. Com um passo de adaptação μ fixo, variaram-se os parâmetros m_1, m_2 e m_3 do algoritmo Acelerador, para que a variância do erro fosse praticamente a mesma em torno da milésima iteração. Além disso, devido à análise de estabilidade deste algoritmo apresentada em [5] e [6], tomou-se o cuidado de considerar a condição $4m_1^2m_2m_3 \approx 1$. Nesta simulação, o passo de aprendizagem do Acelerador (μ) foi feito igual ao do LMS, sendo maior que $1 - \lambda$. Os parâmetros dos algoritmos RLS e LMS foram considerados fixos. As curvas de erro quadrático médio obtidas são mostradas na Figura 3.5 e os parâmetros dos algoritmos de treinamento dos pesos estão na Tabela 3.3.

Algoritmo	Parâmetros
LMS	a) e b) $\mu = 0,05, \beta = 0,05$
AC	a) $m_1 = 10^4, m_2 = 10^{-4}, m_3 = 2,5 \times 10^{-5}, \mu = 0,05$
	b) $m_1 = 6 \times 10^3, m_2 = 3,125 \times 10^{-4}, m_3 = 1,25 \times 10^{-5}, \mu = 0,05$
RLS	a) e b) $\lambda = 0,99, \delta = 0,5$

Tabela 3.3: Parâmetros dos algoritmos de adaptação dos pesos utilizados para a obtenção das curvas de MSE da Figura 3.5.

Na Figura 3.5, obtiveram-se praticamente dois comportamentos extremos do algoritmo Acelerador, ou seja, os parâmetros considerados na Figura 3.5-a tornaram este algoritmo mais lento que o algoritmo LMS. Por outro lado, na Figura 3.5-b, chegou-se a uma situação em que ele apresentou um comportamento intermediário entre os algoritmos LMS e RLS. Considerando ainda outros parâmetros, é possível, deixar o algoritmo Acelerador com uma velocidade de convergência bem próxima da do RLS. Nessa situação, no entanto, a variância do erro deste algoritmo, próximo da milésima iteração, é muito maior que a do erro do algoritmo RLS. De qualquer forma, como no

caso da filtragem adaptativa linear, conclui-se que um bom comportamento do algoritmo Acelerador depende fortemente dos parâmetros m_1, m_2 e m_3 .

Figura 3.5: Curvas de MSE (média de 100 realizações), RBF com 1 saída, $M = 5$ entradas, 64 neurônios ocultos, canal $\mathcal{H}_1$, $\tau_d = 3$, 2-PAM, SNR=20dB, adaptação dos centros e variâncias segundo o algoritmo UCL $\mu_r = 0,05$, e adaptação dos pesos segundo os algoritmos (i) LMS, (ii) AC, (iii) RLS, cujos parâmetros estão mostrados na Tabela 3.3.

Utilizando a Tabela 2.12 da página 40, obtém-se os números de operações por iteração da Tabela 3.4.

No que se refere à complexidade computacional, o desempenho do algoritmo AC+UCL foi intermediário em comparação com os algoritmos LMS+UCL e RLS+UCL, possuindo para este exemplo, apenas 1,30 vezes mais multiplicações e 1,15 vezes mais somas que o primeiro.

O desempenho intermediário do algoritmo Acelerador torna-o atraente no treina-

Operações	LMS+UCL	AC+UCL	RLS+UCL
$\times$	649	847	21839
$\div$	64	64	65
NL	65	65	65
$+$	840	970	17545

Tabela 3.4: Número de operações por iteração dos algoritmos de treinamento para uma rede RBF (5,64) e uma saída.

mento de uma rede RBF. Resultados de simulações evidenciam, que, às custas de um moderado aumento da complexidade computacional, escolhendo-se convenientemente os parâmetros m_1 , m_2 e m_3 , consegue-se uma velocidade de convergência maior que o algoritmo LMS com *momentum*. Por outro lado, na situação de alta velocidade de convergência, embora tenha uma convergência mais lenta que o algoritmo RLS, possui cerca de 25 vezes menos multiplicações e 18 vezes menos somas.

3.2.3 Redes recorrentes

Com relação às redes recorrentes, alguns autores, como por exemplo Parisi *et al.* em [41], propuseram o uso de algoritmos baseados no método dos mínimos quadrados recursivo para seu treinamento. Esses algoritmos, embora mais rápidos, são proporcionalmente bem mais complexos. Por outro lado, o algoritmo RTRL, usado neste trabalho, apresenta uma velocidade de convergência comparável à do algoritmo RLS de filtragem adaptativa linear se os parâmetros do mesmo forem escolhidos de forma apropriada [18].

Em [19] foi feito um estudo comparativo de vários algoritmos propostos na literatura para o treinamento de RNN's. Dentre esses, o RTRL apresenta vantagens com relação à estabilidade, exatidão e possibilidade de uso em tempo real. Em contrapartida, é um dos mais complexos entre os algoritmos baseados no gradiente estocástico. Essa

diferença de complexidade, porém, só se torna importante quando se usa redes com um grande número de neurônios. Para equalização, por sua vez, resultados de simulações mostram que 2 neurônios são suficientes para que se obtenha um bom desempenho com uma rede RNN [18].

Diante destes resultados, optou-se pelo algoritmo RTRL dadas suas vantagens e também por ser um dos mais conhecidos no treinamento de redes recorrentes, não se fazendo como para as outras redes, um estudo comparativo de algoritmos de treinamento.

3.3 Comparação de desempenho dos equalizadores

Com o objetivo de comparar o desempenho dos diversos tipos de equalizadores apresentados no Capítulo 2, foram obtidas experimentalmente curvas de taxa de erro de símbolo para os canais da Tabela 3.1. Para obtenção dessas curvas, os parâmetros dos equalizadores foram mantidos fixos após a convergência e utilizou-se então uma sequência de 10^6 ou 10^7 símbolos, a fim de que o número mínimo de erros para uma dada relação sinal-ruído fosse pelo menos superior a 30. As simulações seguiram os esquemas das figuras 2.1 (página 10) e 2.14 (página 58), sendo que a segunda se refere ao uso de realimentação de decisões. Dois tipos de modulação foram considerados: 2-PAM e 4-QAM, cujos alfabetos são respectivamente dados por: $\{-1, +1\}$ e $\{-1 - j, -1 + j, +1 - j, +1 + j\}$. Os símbolos foram considerados equiprováveis e estatisticamente independentes e o ruído adicionado é branco, gaussiano de média nula e variância σ_η^2 .

Uma parte deste estudo foi apresentado em forma de um artigo no XVIII Simpósio Brasileiro de Telecomunicações-SBrT (Referência [52]).

3.3.1 Resultados das simulações com 2-PAM

As configurações de cada equalizador estão mostradas na Tabela 3.5. Os equalizadores utilizados são de 4ª ordem ($M = 5$) e o atraso é de 3 amostras ($\tau_d = 3$), o que é suficiente para definir a estrutura do LTE. Para a rede MLP utilizou-se a configuração proposta por Gibson *et al.* em [39]. Para a RNN seguiu-se a configuração de [18] tomando-se o cuidado de aumentar o número de entradas externas. Para a rede RBF, por sua vez, considerou-se $N = 256$ neurônios, cujos centros são escolhidos de acordo com o sinal de entrada e as variâncias feitas iguais à variância desse sinal.

No caso de $M = 5$ entradas e um canal de 2ª ordem, o número total de diferentes entradas possíveis (estados desejados) do equalizador é 128 [14]. Dessa forma, se não for utilizado nenhum algoritmo para adaptação dos centros e variâncias da rede RBF, é preciso utilizar um número de neurônios maior que o de estados para obter um bom desempenho. No caso da modulação 2-PAM e dos canais considerados, a configuração da rede RBF pode ser diminuída se for feita a adaptação dos centros e variâncias. Isso será discutido com mais detalhes posteriormente.

Op.	LTE (EF-LSL)	MLP (BP) (5,9,3,1)	RBF (LMS) ($N = 256$)	RNN (RTRL) ($N = 3$)
$\times$	65	465	2052	320
$\div$	10	—	—	—
NL	—	36	257	6
+	45	395	3075	124

Tabela 3.5: Configurações dos equalizadores utilizados e número de operações por iteração de seus algoritmos de treinamento para $M = 5$.

Na Figura 3.6-a, estão as curvas de taxa de erro de símbolo para o canal $\mathcal{H}_1$. Neste caso, o desempenho do equalizador MLP para o intervalo de relação sinal-ruído (SNR) considerado é muito próximo do obtido com o equalizador linear. No entanto, os equalizadores RBF e RNN possuem desempenhos superiores ao do equalizador MLP

e conseqüentemente ao do linear. É importante notar que a rede recorrente consegue resultados melhores que os da RBF, possuindo uma complexidade computacional menor no treinamento. O resultado do equalizador ótimo segundo Bayes ficou bem próximo do resultado do equalizador baseado na rede RBF. Como foi comentado anteriormente, conhecendo-se os estados desejados do canal, é possível implementar um equalizador de Bayes através dessa rede. Neste caso, porém, o canal era desconhecido e os centros foram escolhidos através de um estudo do sinal de entrada da rede o que fez com que os desempenhos não fossem exatamente iguais. Ainda nesta figura, percebe-se que a rede RNN apresenta o desempenho mais próximo do ótimo segundo o critério da máxima verossimilhança, que na legenda da figura foi denominado de MLSE (*Maximum Likelihood Sequence Estimator*).

Considerando o canal $\mathcal{H}_2$, obtém-se as curvas de taxa de erro de símbolo da Figura 3.6-b. Deve-se notar que neste caso, o equalizador MLP apresenta um desempenho superior ao do linear, diferente do que acontece na ausência das não-linearidades. Para este canal, não foi projetado o equalizador ótimo segundo o critério da máxima verossimilhança, devido à dificuldade na escolha do filtro casado branqueador para canais não-lineares. É interessante observar ainda, que o desempenho da rede RBF se afastou um pouco do ótimo segundo Bayes, sendo que o mais próximo do mesmo foi obtido com a RNN.

Quanto ao canal $\mathcal{H}_3$ (Figura 3.6-c), que possui dois zeros na circunferência unitária, observa-se que apenas dois equalizadores apresentaram bons resultados: a rede RNN e o equalizador ótimo segundo o critério da máxima verossimilhança. Para este tipo de canal, parece que a existência da realimentação no equalizador é essencial para garantir uma boa equalização. A realimentação é uma característica presente na rede neural recorrente e no equalizador ótimo segundo o critério da máxima verossimilhança. Neste último, basta observar a realimentação existente no algoritmo de Viterbi, do qual uma possível implementação pode ser vista na Figura 2.11 da página 55.

Figura 3.6: Curvas de taxa de erro de símbolo para $M = 5$, $\tau_d = 3$, 2-PAM, configurações dos equalizadores da Tabela 3.5 e canais a) $\mathcal{H}_1$; b) $\mathcal{H}_2$; c) $\mathcal{H}_3$.

Um problema quando se trabalha com equalização supervisionada é a escolha do atraso τ_d da resposta desejada. Para um canal de 2ª ordem e 5 entradas, o atraso “ótimo” para o equalizador linear transversal é de 3 amostras. Por isso, esse atraso foi considerado para todas as estruturas. Na Figura 3.7, para uma relação sinal-ruído de 12,5 dB, estão mostrados valores de taxa de erro de símbolo em função do atraso para cada equalizador utilizado e os canais $\mathcal{H}_1$, $\mathcal{H}_2$ e $\mathcal{H}_3$. É interessante observar nesta figura que a variação com relação ao atraso é pequena, ou seja, independente do canal, considerando um atraso de 1, 2, 3 ou 4 amostras obtém-se bons resultados em todos os equalizadores. No entanto é possível observar ainda que o atraso de 3 amostras garante quase sempre o melhor desempenho dos equalizadores considerados.

Figura 3.7: Taxa de erro de símbolo em função do atraso para 2-PAM e $SNR = 12,5$ dB.

A partir dessas simulações, observou-se que para os canais considerados, a rede recorrente treinada com o algoritmo RTRL possui um desempenho superior a todos os tipos de equalizadores utilizados. Além disso, como pode ser observado na Tabela 3.5, seu algoritmo de treinamento apresenta a menor complexidade computacional quando comparado com os algoritmos de treinamento das outras redes estudadas. Os bons resultados da rede neural recorrente sugerem a introdução de algum tipo de recorrência nas redes MLP, RBF e na própria RNN visando a melhora de seus desempenhos. Isso pode ser feito de diversas formas, mas a que é mais utilizada quando se trata de equalização é a realimentação de decisões (DF). Fizeram-se então simulações para os canais $\mathcal{H}_1$, $\mathcal{H}_2$ e $\mathcal{H}_3$ da Tabela 3.1 considerando a utilização desta filosofia (Figura 2.14 da página 58) cujos resultados são mostrados na Figura 3.8.

Para efeito de comparação, as estruturas com DF foram simuladas com o mesmo número de entradas das estruturas sem DF. Nestas estruturas M_f é o número de entradas que corresponde à sequência de saída do canal afetada de ruído e M_b é o número de entradas relacionadas à realimentação das decisões já tomadas. Verificou-se experimentalmente que se deve usar $M_f > M_b$.

Comparando as Figuras 3.6 e 3.8, observa-se que para os canais considerados, $\tau_d = 3$ e modulação 2-PAM, a realimentação das decisões já tomadas garante um melhor desempenho na equalização. Há inclusive uma proximidade maior do desempenho das redes com os resultados obtidos pelos equalizadores ótimos. Para o canal $\mathcal{H}_3$, por exemplo, a realimentação de decisões possibilitou uma melhora significativa do desempenho do LTE, das redes MLP e RBF e do equalizador ótimo segundo o critério de Bayes.

Observando ainda a Figura 3.8, verifica-se que a realimentação das decisões na rede RNN não proporciona melhoras em seu desempenho para os canais simulados e o atraso considerado. Ou seja, a realimentação de decisões aparentemente não acrescenta nenhuma informação nova em redes que já utilizam recorrência de estados internos.

Figura 3.8: Curvas de taxa de erro de símbolo para $M_f = 3$, $M_b = 2$, $\tau_d = 3$, 2-PAM, configurações dos equalizadores da Tabela 3.5, utilização de realimentação de decisões e canais a) $\mathcal{H}_1$; b) $\mathcal{H}_2$; c) $\mathcal{H}_3$.

Fazendo novamente o estudo do desempenho das estruturas em função do atraso para $SNR = 12,5$ dB, obtiveram-se os resultados da Figura 3.9. É interessante notar nesta figura que para a configuração $M_f = 3$ e $M_b = 2$ utilizada, há uma maior variação do desempenho em função do atraso. Não se pode mais afirmar que um atraso de 3 amostras garante o melhor resultado para todas as estruturas. Para a RNN aplicada ao canal $\mathcal{H}_3$, por exemplo, um atraso de 4 amostra parece ser melhor que o de 3 utilizado.

Figura 3.9: Taxa de erro de símbolo em função do atraso para 2-PAM e $SNR = 12,5$ dB com utilização de DF.

Quando se usa a realimentação de decisões, torna-se possível a redução do número de neurônios das redes para se obter uma mesma probabilidade de erro. No caso do canal $\mathcal{H}_2$, isso pode ser comprovado observando-se a Tabela 3.6 e a Figura 3.10. Nota-se que com uma configuração mais simples (MLP_2 e RBF_2) e com o uso de realimentação de decisões o desempenho dessas redes ainda é comparável ao das consideradas an-

teriormente. Na Figura 3.10 é mostrada também a taxa de erro de símbolo da rede RNN sem realimentação de decisões que, para este canal, ficou bem próxima das demais obtidas com o uso de DF. Observando a Tabela 3.6, conclui-se que mesmo com o uso de recorrência, as redes MLP e RBF apresentam um treinamento com maior complexidade que a rede RNN.

Para a rede MLP₂ com configuração da Tabela 3.6, embora seu algoritmo de treinamento tenha menos multiplicações em comparação com o algoritmo RTRL, ele perde em número de somas e de cálculos de não-linearidades (NL). No caso da rede RBF, não foi utilizado um algoritmo para adaptação dos centros e variâncias. A utilização de um algoritmo desse tipo, como será visto posteriormente, faz com que esta rede tenha uma complexidade computacional comparável com a das demais.

Op.	MLP ₁ (BP) (5,9,3,1)	MLP ₂ (BP) (5,5,2,1)	RBF ₁ (LMS) ($N = 256$)	RBF ₂ (LMS) ($N = 128$)	RNN (RTRL) ($N=3$)
$\times$	465	288	2052	1028	320
$\div$	—	—	—	—	—
NL	36	26	257	129	6
+	395	238	3075	1539	124

Tabela 3.6: Configurações dos equalizadores utilizados para obtenção da Figura 3.10 e os números de operações no treinamento.

A equalização do canal $\mathcal{H}_3$

O canal $\mathcal{H}_3$, como pode ser observado na Tabela 3.1 da página 71, possui dois zeros na circunferência unitária. A dificuldade de se fazer a equalização considerando este tipo de canal está relacionada com a localização dos estados desejados. Para o canal $\mathcal{H}_3$, utilizando $M = 2$ obtém-se os estados da Tabela 3.7. Pode-se notar nesta tabela que os estados são iguais quando se têm na entrada do canal: $a(n) = a(n-1) = \dots = a(n-k) = -1$ ou $a(n) = a(n-1) = \dots = a(n-k) = 1$, com k inteiro positivo.

Figura 3.10: Curvas de taxa de erro de símbolo para $M = 5$ (RNN), $M_f = 3$, $M_b = 2$ (MLP e RBF), $\tau_d = 3$, 2-PAM, configurações dos equalizadores segundo a Tabela 3.6; canal $\mathcal{H}_2$.

Independente do atraso, é muito difícil para um equalizador sem recorrência decidir entre $+1$ ou -1 nesta situação.

Observando as figuras 3.6-c e 3.8-c verifica-se que somente quando se usa algum tipo de realimentação consegue-se obter um bom desempenho. Nas simulações sem DF, os únicos equalizadores que conseguiram bons resultados para o canal $\mathcal{H}_3$ foram a RNN e o MLSE. A RNN possui realimentação de estados internos e no MLSE há realimentação de probabilidades que pode ser vista através da possível implementação do algoritmo de Viterbi da Figura 2.11. Quando se usa DF, por sua vez, há uma melhora nos desempenhos de todos os equalizadores, inclusive do ótimo segundo o critério de Bayes.

Como foi explicado na Seção 2.7.1, a realimentação de decisões tende a separar os estados em classes distintas de acordo com a decisão passada. Assim os estados coincidentes obtidos quando as entradas são $a(n) = a(n-1) = \dots = a(n-k) = -1$ e $a(n) = a(n-1) = \dots = a(n-k) = 1$, com k inteiro positivo, passam a não existir mais pois são separados em duas classes: uma em que a decisão anterior é -1 e outra em que essa decisão é +1.

$a(n)$	$a(n-1)$	$a(n-2)$	$a(n-3)$	$x(n)$	$x(n-1)$
-1	-1	-1	-1	0	0
-1	-1	-1	1	0	2
-1	-1	1	-1	2	-4
-1	-1	1	1	2	-2
-1	1	-1	-1	-4	2
-1	1	-1	1	-4	4
-1	1	1	-1	-2	-2
-1	1	1	1	-2	0
1	-1	-1	-1	2	0
1	-1	-1	1	2	2
1	-1	1	-1	4	-4
1	-1	1	1	4	-2
1	1	-1	-1	-2	2
1	1	-1	1	-2	4
1	1	1	-1	0	-2
1	1	1	1	0	0

Tabela 3.7: Entrada e estados desejados do canal $\mathcal{H}_3$, $M = 2$.

A adaptação dos centros na rede RBF

A rede RBF usada nas simulações apresentadas anteriormente foi treinada com o algoritmo LMS para atualização apenas dos pesos. Os centros foram escolhidos a partir do sinal de entrada do equalizador e as variâncias das funções gaussianas foram consideradas iguais à variância desse sinal. Essa técnica usada em equalização [42] só garante um bom desempenho dessa rede se for usado um número de neurônios no mínimo igual ao número de estados do problema em questão. Isso porque quando se considera menos neurônios, corre-se o risco de não cobrir todo o espaço de estados, treinando-se então uma rede deficiente para algumas entradas. Na seção anterior, foram feitas simulações com canais de 2ª ordem e equalizadores com $M = 5$ entradas. Há então 128 estados. No entanto, foi preciso considerar uma rede RBF com 256 neurônios, sendo que em muitas situações, quando treinada com 128, não foi observada a devida convergência do algoritmo de treinamento.

Diante dessas observações, percebe-se que o uso de um algoritmo para adaptação dos centros e variâncias das funções de base radial é muito importante porque possivelmente possibilita a redução da complexidade da rede RBF. A fim de comprovar isso foram feitas simulações considerando no treinamento o algoritmo UCL apresentado na Seção 2.4.1.

Na Figura 3.11, podem ser vistas curvas de taxa de erro de símbolo para os canais $\mathcal{H}_1$, $\mathcal{H}_2$ e $\mathcal{H}_3$, $M = 5$ entradas e 32, 64, 128, 256 neurônios. Na Figura 3.12 considerou-se o uso de realimentação de decisões. Nessas figuras, somente para $N = 256$ não se considerou a adaptação dos centros e variâncias.

Observando essas figuras, conclui-se que quando se usa o algoritmo UCL para adaptação dos centros e variâncias, torna-se possível a diminuição do número de neurônios para obter um mesmo desempenho da rede. Na Figura 3.11 verifica-se que $N = 128$ e $N = 64$ proporcionam o mesmo desempenho que o observado para $N = 256$. Porém para 32 neurônios, o resultado é inferior. Dessa forma, neste caso deve-se usar um número de neurônios entre a metade do número de estados e um quarto desse número.

Na Figura 3.12, a conclusão obtida na seção anterior é confirmada. Ou seja, quando se usa realimentação de decisões, torna-se possível, diminuir ainda mais o número de neurônios, tornando a rede mais simples para se obter uma mesma taxa de erros. Nesta figura, o desempenho obtido com $N = 32$ é similar ao obtido com $N = 64$ quando se usa realimentação de decisões, sendo possível até mesmo usar $N < 32$ para se conseguir um resultado semelhante.

Na Tabela 3.8 são apresentados os números de operações por iteração, considerando o algoritmo LMS para adaptação dos pesos e atualização dos centros e variâncias, segundo o algoritmo UCL.

Op.	$N = 256$	$N = 128$	$N = 64$	$N = 32$
$\times$	2052	1289	649	329
$\div$	-	128	64	32
NL	257	129	65	33
+	3075	1672	840	424

Tabela 3.8: Número de operações por iteração do algoritmo LMS+UCL. Para $N = 256$, os números se referem apenas à atualização dos pesos segundo o LMS.

Observando a Tabela 3.8, conclui-se que a utilização dessa técnica para a adaptação dos centros e variâncias faz com que a rede RBF seja considerada como uma solução bastante interessante para o problema de equalização. Quando se usa $N = 32$ e realimentação de decisões, a rede RBF só perde para a RNN em número de operações de seu algoritmo de treinamento, sendo que o desempenho em termos de taxa de erro de símbolo é praticamente o mesmo.

Figura 3.11: Curvas de taxa de erro de símbolo, $M = 5$, $\tau_d = 3$, 2-PAM, RBF treinada com LMS+UCL (exceto para $N = 256$), canais a) $\mathcal{H}_1$; b) $\mathcal{H}_2$; c) $\mathcal{H}_3$.

Figura 3.12: Curvas de taxa de erro de símbolo, $M_f = 4$, $M_b = 1$, $\tau_d = 3$, 2-PAM, RBF treinada com LMS+UCL (exceto para $N = 256$), com realimentação de decisões (DF), canais a) $\mathcal{H}_1$; b) $\mathcal{H}_2$; c) $\mathcal{H}_3$.

3.3.2 Resultados das simulações com 4-QAM

Simulações considerando a modulação 4-QAM foram feitas a fim de testar os algoritmos de treinamento complexos além de verificar se as conclusões obtidas para o caso real (2-PAM) se repetiam. Para isso foram obtidas curvas de taxa de erro de símbolo para 3 canais cujos coeficientes são complexos: $\mathcal{H}_{1c}$, $\mathcal{H}_{2c}$ e $\mathcal{H}_{3c}$ (Tabela 3.1).

As configurações dos equalizadores utilizados são mostradas na Tabela 3.9. Considerou-se $M = 5$ entradas e um atraso de $\tau_d = 3$ amostras, o que é suficiente para definir a estrutura do LTE. Para a rede MLP utilizou-se a configuração (5,10,1) proposta em [43]. Para a rede RNN, seguiu-se a configuração com $N = 3$ neurônios de [18].

No caso da RBF, o número de neurônios utilizados é bem maior que nas outras estruturas. Considerando uma situação sem ruído, para $M = 5$, modulação 4-QAM e um canal de 2ª ordem, o número total de diferentes entradas possíveis (estados desejados) do equalizador é $4^{5+2} = 16384$ [14]. Treinar a rede com esse número de neurônios é algo inviável. Utilizou-se então 512 neurônios e o algoritmo LMS+UCL para o treinamento da rede. Tal algoritmo como vimos possibilita a utilização de estruturas mais simples (com menos neurônios) para obtenção de uma mesma taxa de erros.

MLP	RBF	RNN
(5,10,1)	($M_1 = 512$)	($N = 3$)

Tabela 3.9: Configurações dos equalizadores utilizados para modulação 4-QAM, $M = 5$ e $\tau_d = 3$.

Os equalizadores considerados foram comparados com os ótimos segundo o critério de Bayes e o critério da máxima verossimilhança [23] em termos de curvas de taxa de erro de símbolo (SER). Inicialmente, não foi considerada realimentação de decisões (Figura 3.13). Na Figura 3.15, por sua vez tal realimentação foi levada em conta.

Observando a Figura 3.13-a nota-se que o equalizador linear transversal (LTE) e a rede recorrente apresentaram os melhores resultados para o canal $\mathcal{H}_{1c}$, sendo que estes estão próximos dos resultados dos equalizadores ótimos segundo Bayes e o critério da máxima verossimilhança. Este canal, embora seja de fase não-mínima, é fácil de equalizar, pois uma estrutura linear de baixa complexidade consegue ter um bom desempenho.

Considerando o canal não-linear $\mathcal{H}_{2c}$ (Figura 3.13-b), observa-se que o LTE apresenta o pior desempenho. Os equalizadores não-lineares apresentam resultados que estão longe do ótimo segundo Bayes. Dentre eles, a rede MLP apresenta o melhor desempenho, bem diferente do que acontece na ausência de não-linearidades.

Finalmente para o canal $\mathcal{H}_{3c}$ (Figura 3.13-c), que possui dois zeros próximos da circunferência unitária, a situação do caso real se repete. Em canais desse tipo, há estados bem próximos o que dificulta a equalização. Para tratar esse problema, é necessário utilizar estruturas que possuam realimentação, como é o caso da RNN e do equalizador ótimo segundo o critério da máxima verossimilhança. É interessante notar ainda que quando se utiliza o equalizador ótimo segundo o critério de Bayes, mesmo não havendo realimentação, obtém-se um bom resultado, diferente do que fora observado no caso real. No entanto, o canal $\mathcal{H}_{3c}$ não possui estados coincidentes como o canal $\mathcal{H}_3$ o que proporciona um funcionamento razoável para o equalizador bayesiano.

Um outro aspecto interessante de ser analisado é a sensibilidade das estruturas com relação ao atraso, que pode ser visto na Figura 3.14. Para a modulação 4-QAM, SNR= 12,5 dB e os canais considerados, o desempenho das redes e do LTE são bem mais dependentes do atraso que no caso binário (2-PAM). Novamente, é possível observar nos exemplos considerados que um atraso de 3 amostras proporciona o melhor desempenho em quase todas as estruturas independente dos canais $\mathcal{H}_{1c}$, $\mathcal{H}_{2c}$ e $\mathcal{H}_{3c}$.

Figura 3.13: Curvas de taxa de erro de símbolo para $M = 5$, $\tau_d = 3$, 4-QAM, configurações dos equalizadores da Tabela 3.9 e canais a) $\mathcal{H}_{1c}$; b) $\mathcal{H}_{2c}$; c) $\mathcal{H}_{3c}$.

Figura 3.14: Taxa de erro de símbolo em função do atraso para 4-QAM e SNR=12,5 dB.

Considerando a realimentação de decisões, esperava-se uma melhora no desempenho de todas as estruturas como aconteceu no caso real. No entanto, isso não foi observado sempre.

Para o canal $\mathcal{H}_{1c}$ (Figura 3.15-a), sendo ele fácil de equalizar, a realimentação de decisões erradas pode ser a causadora da piora na taxa de erros.

Para o canal $\mathcal{H}_{2c}$ (Figura 3.15-b), a realimentação de decisões já proporciona melhoras nos resultados das estruturas, exceto para a rede RNN. Como já fora mencionado anteriormente, DF na RNN não proporcionou mudanças significativas no desempenho desta rede para modulação 2-PAM. No caso de modulação 4-QAM porém, observou-se uma piora para todos os canais considerados. Isso talvez seja reflexo da estrutura de realimentação de decisões $M_f = 3$ e $M_b = 2$ e do atraso de $\tau_d = 3$ amostras

considerados.

No caso do canal $\mathcal{H}_{3c}$, é interessante observar que a realimentação de decisões proporcionou uma melhora significativa nos desempenhos das redes MLP e RBF, do LTE e também do equalizador ótimo segundo o critério de Bayes. Como no caso real, quando há zeros próximos ou sobre a circunferência unitária a utilização de estruturas com realimentação é essencial na equalização. Um fato curioso é que fazendo a realimentação de decisões no LTE, obtém-se para este canal um desempenho melhor que o ótimo segundo o critério da máxima verossimilhança e bem próximo do ótimo segundo Bayes, o que mostra que para certos canais, a utilização de uma estrutura mais simples como o LTE acrescido de DF consegue obter resultados excelentes.

Fazendo novamente uma análise da sensibilidade dessas estruturas para uma relação sinal-ruído de 12,5 dB com relação ao atraso, obtém-se as curvas da Figura 3.16. É interessante observar que o desempenho quando se usa DF é muito mais dependente do atraso que na ausência de realimentação de decisões. Um outro aspecto a ser observado é que, com exceção da RNN, um atraso de 2 amostras parece ser o melhor para as demais estruturas utilizadas.

Figura 3.15: Curvas de taxa de erro de símbolo para $M_f = 3$, $M_b = 2$, $\tau_d = 3$, 4-QAM, configurações dos equalizadores da Tabela 3.9, utilização de realimentação de decisões e canais a) $\mathcal{H}_{1c}$; b) $\mathcal{H}_{2c}$; c) $\mathcal{H}_{3c}$.

Figura 3.16: Taxa de erro de símbolo em função do atraso para 4-QAM e SNR= 12,5 dB com utilização de DF.

Capítulo 4

Equalizadores Híbridos

Como vimos ao longo da dissertação, o LTE está longe de ser uma solução ótima para o problema de equalização. No entanto, quando se utiliza realimentação de decisões, consegue-se em muitas situações obter resultados satisfatórios. A utilização das decisões passadas faz com que o LTE se torne uma estrutura não-linear conhecida como DFE e que ainda é computacionalmente bem simples.

O problema é que nem sempre se consegue um bom desempenho com o DFE. Para canais não-lineares e para os que possuem zeros sobre a circunferência unitária é interessante utilizar uma estrutura não-linear mais complexa como uma rede neural.

Diante disso, muitos autores propuseram estruturas híbridas constituídas da junção de uma estrutura linear como o LTE e de estruturas não-lineares como as redes neurais. Surgiram também variações destas quando realimentação de decisões é utilizada.

Lee e Pearson em [44] propuseram um equalizador híbrido que segue o esquema da Figura 4.1. Como pode ser observado nesta figura, a rede RBF é usada somente quando o erro do LTE é maior que um certo limiar (γ). Dessa forma, segundo [44], a parcela não-linear dessa estrutura só é usada em regiões do espaço de estados em que a separação linear não consegue obter um desempenho razoável que é determinado pelo limiar utilizado.

Embora os resultados das simulações apresentados em [44] evidenciem o melhor

Figura 4.1: Diagrama do equalizador híbrido proposto em [44].

funcionamento do equalizador híbrido proposto em relação ao linear, seu esforço computacional é muito grande. Isso porque nas situações em que o erro do LTE é maior que o limiar, a complexidade computacional da estrutura híbrida é a soma das complexidades do LTE e da rede RBF. Além disso, como vimos, deve-se ainda usar um algoritmo para atualização dos centros e variâncias a fim de utilizar uma RBF com menos neurônios.

Nesse tipo de equalizador híbrido, a estrutura não-linear serve para dar o ajuste fino na equalização feita pelo LTE. No entanto, para canais não-lineares, a parcela não-linear do equalizador deve ter uma participação maior pois como vimos para estes tipos de canais o desempenho do LTE não é satisfatório.

Dessa forma, alguns autores propuseram a utilização do LTE em série com uma rede neural. Em [45], foram apresentadas algumas estruturas híbridas utilizadas na equalização de canais de satélite. Essas estruturas consistem basicamente do LTE em série com uma rede RBF ou MLP utilizando ou não realimentação de decisões. Os resultados das simulações mostrados em [45] evidenciam que a utilização de uma estrutura híbrida garante um bom desempenho na equalização de canais não-lineares.

Estes equalizadores híbridos, porém, possuem complexidades computacionais elevadas impostas pelo uso das redes RBF e MLP. Por outro lado, os resultados das

simulações apresentados no Capítulo 3 mostraram um bom funcionamento da rede neural recorrente, tanto do ponto de vista de complexidade computacional do algoritmo de treinamento utilizado (RTRL), como de desempenho. É interessante observar que a realimentação de decisões nessa rede deve ser feita de forma cautelosa. No caso real (2-PAM), a realimentação de decisões não proporcionou melhoras significativas no desempenho dessa rede. No caso complexo (4-QAM) porém, a realimentação fez com que os resultados da mesma se tornassem um pouco piores. O que é interessante de se observar também é que no caso complexo o DFE apresentou um desempenho próximo do ótimo em algumas situações em que a RNN não funcionou tão bem.

Diante dessas observações, resolvemos propor um equalizador híbrido formado a partir da RNN e do LTE. Este equalizador é apresentado na seção seguinte.

4.1 Equalizador híbrido proposto

O equalizador híbrido proposto é formado a partir do LTE que constitui a parcela linear e da RNN que é a parte não-linear. A disposição dessas estruturas pode ser vista na Figura 4.2. Deve-se notar que elas estão em série e que a entrada do LTE é formada pelo vetor de saída do canal afetado de ruído $\mathbf{u}(n)$ e pelo vetor de saída da RNN.

O treinamento desse equalizador pode ser feito utilizando o algoritmo RTRL para a adaptação dos parâmetros da RNN e o LMS para atualização dos pesos do LTE. O único problema a ser considerado é a propagação do erro para a adaptação da RNN. Para isso, deve-se observar a Figura 4.3. Vamos supor inicialmente que não haja erro na RNN e que a saída do LTE é o próprio sinal desejado (Figura 4.3-a). Supondo que haja erro na RNN, obtém-se na saída do LTE o sinal desejado $d(n)$ menos o sinal de erro $e(n)$ (Figura 4.3-b). Subtraindo os esquemas das figuras 4.3-a e 4.3-b, obtém-se a Figura 4.3-c e o sinal de erro pode ser descrito pela seguinte equação

Figura 4.2: Diagrama do equalizador híbrido proposto.

$$e(n) = \sum_{k=M_f}^{M_f+M_b-1} w_k(n) e_{\text{RNN}}(n - k - M_f), \quad (4.1)$$

sendo M_f o número de entradas que constituem a saída do canal afetada de ruído e M_b o número de entradas que correspondem à saída da RNN.

A partir desta equação, pode-se calcular o erro da RNN como

$$e_{\text{RNN}}(n) = \frac{1}{w_{M_f}(n)} \left[e(n) - \sum_{k=M_f+1}^{M_f+M_b-1} w_k(n) e_{\text{RNN}}(n - k - M_f) \right]. \quad (4.2)$$

Observando a equação anterior, nota-se que há um problema de divisão por zero, quando $w_{M_f}(n)$ for nulo. Dessa forma, a fim de evitar este problema e diminuir a complexidade computacional do algoritmo de treinamento da estrutura híbrida, considerou-se $e_{\text{RNN}}(n) = e(n)$, sendo $w_{M_f}(n) = 1$. A partir de simulações, verificou-se que a simplificação considerada não gera mudanças significativas no desempenho do equalizador. Deve-se notar que tal simplificação impõe que o peso $w_{M_f}(n)$ seja sempre igual a 1, tanto para o caso real como para o complexo.

Figura 4.3: Propagação do erro para adaptação da RNN.

Uma variação do esquema da Figura 4.2 pode ser visto na Figura 4.4. Nesta figura, considera-se a realimentação de decisões na RNN.

Figura 4.4: Diagrama do equalizador híbrido proposto considerando DF na RNN.

Resultados de simulações evidenciam que o esquema do equalizador híbrido da Figura 4.4 em que se faz realimentação de decisões na RNN é melhor que o esquema da Figura 4.2. Novamente, a propagação do erro para a RNN é feita da forma simplificada: $e_{\text{RNN}}(n) = e(n)$, mantendo sempre $w_{M_f}(n) = 1$. O número de entradas referentes à realimentação de decisões na RNN é designado por M_r .

Como foi dito anteriormente, o treinamento desse equalizador híbrido pode ser feito com o algoritmo RTRL para o ajuste dos parâmetros da RNN e o algoritmo LMS para a adaptação dos coeficientes do LTE. A complexidade do algoritmo de treinamento do equalizador híbrido como um todo corresponde às somas das complexidades computacionais de cada algoritmo considerado separadamente desde que se use a propagação de erro para a RNN da forma simplificada. Como na prática se deseja utilizar este

equalizador em tempo real, é desejável que além de ter uma baixa complexidade computacional, ele tenha também uma alta velocidade de convergência. Para que esta última condição seja mantida, os passos de adaptação do LMS e do RTRL devem ser tomados da mesma ordem de grandeza (0,01, por exemplo). Há situações, no entanto, em que é conveniente aumentar o passo de aprendizagem do RTRL tornando-o cerca de 10 vezes maior que o passo do LMS.

Na seção seguinte são mostrados os resultados das simulações considerando o equalizador híbrido da Figura 4.4.

4.2 Resultados experimentais com o equalizador híbrido

4.2.1 Canais usados nas simulações

Nas simulações com o equalizador híbrido foram usados os canais da Tabela 4.1. Dentre eles, há um canal de microondas obtido a partir do processamento de medidas de campo de sinais recebidos [46] e cujos dados podem ser obtidos na referência [47]. Para simulações com a modulação 2-PAM, foi utilizada a parte real do canal em questão. As figuras 4.5 e 4.6 se referem a este canal.

Figura 4.5: a) Coeficientes do canal $\mathcal{H}_6$, parte real dos coeficientes do canal $\mathcal{H}_{6c}$, b) Parte imaginária dos coeficientes do canal $\mathcal{H}_{6c}$, c) Zeros do canal $\mathcal{H}_6$, d) Zeros do canal $\mathcal{H}_{6c}$.

$\mathcal{H}_4$ (Linear com zeros na circunferência unitária [18]) $\mathcal{H}_4(z) = 1 - 2z^{-1} + z^{-2}$ Zeros: $\{1;1\}$
$\mathcal{H}_5$ (Não-Linear) Parte não-linear: $x(n) = x_l(n) + 0,07x_l^2(n) - 0,03x_l^3(n)$ Parte linear: $\mathcal{H}_{5l}(z) = \mathcal{H}_4(z)$
$\mathcal{H}_6$ (Parte real de um canal de microondas [46], [47]) Modelo FIR de 300 coeficientes com zeros indicados na Figura 4.5 e resposta em frequência na Figura 4.6.
$\mathcal{H}_7$ (Não-linear) Parte não-linear: $x(n) = x_l(n) + 0,1x_l^2(n) + 0,05x_l^3(n)$ Parte linear: $\mathcal{H}_{7l}(z) = \mathcal{H}_6(z)$
$\mathcal{H}_{4c}$ (Linear com zeros próximos da circunferência unitária [17]) $\mathcal{H}_{4c}(z) = (0,4313 + j0,4311) + (0,3663 - j1,0996)z^{-1} + (-0,5282 + j0,0756)z^{-2}$ Zeros: $\{(0,4996 + j1,0000); (0,3503 + j0,7000)\}$ Módulo dos zeros: $\{1,1179; 0,7827\}$
$\mathcal{H}_{5c}$ (Não-Linear) Parte não-linear: $x(n) = x_l(n) + 0,1x_l^2(n) + 0,05x_l^3(n)$ Parte linear: $\mathcal{H}_{5cl}(z) = \mathcal{H}_{4c}(z)$
$\mathcal{H}_{6c}$ (Canal de microondas [46], [47]) Modelo FIR de 300 coeficientes com zeros indicados na Figura 4.5 e resposta em frequência na Figura 4.6.
$\mathcal{H}_{7c}$ (Não-linear) Parte não-linear: $x(n) = x_l(n) + 0,05x_l^2(n) + 0,01x_l^3(n)$ Parte linear: $\mathcal{H}_{7cl}(z) = \mathcal{H}_{6c}(z)$

Tabela 4.1: Modelos dos canais de comunicação usados nas simulações com o equalizador híbrido.

Figura 4.6: a) e b) Resposta em frequência do canal $\mathcal{H}_{6c}$, c) e d) Resposta em frequência do canal $\mathcal{H}_6$.

4.2.2 Resultados das simulações com 2-PAM

A fim de verificar o funcionamento do equalizador híbrido da Figura 4.4 no caso real, foram obtidas curvas de taxa de erro de símbolo para a modulação 2-PAM. Na obtenção destas curvas, mantiveram-se os parâmetros dos equalizadores fixos após a convergência e utilizou-se então uma sequência de 10^6 ou 10^7 símbolos a fim de que o número mínimo de erros fosse sempre superior a 30, independente da relação sinal-ruído. Para efeito de comparação, considerou-se também a RNN e o DFE separadamente.

As configurações das estruturas utilizadas estão mostradas na Tabela 4.2. Considerando esta tabela, a complexidade computacional do equalizador híbrido após o treinamento é da mesma ordem de grandeza da complexidade da RNN_2 e do DFE_1 .

Por isso estas estruturas foram consideradas. A grandeza N representa o número de unidades da rede recorrente. A propagação do erro para a RNN na estrutura híbrida é feita da forma simplificada como explicado anteriormente. A estrutura DFE₁ foi utilizada para canais de 2ª ordem, considerando um atraso de $\tau_d = 3$ amostras. Como a ordem do equalizador é maior neste caso, um atraso maior possivelmente proporcionaria melhores resultados. No entanto, isso não acontece e dessa forma foi utilizado o atraso de $\tau_d = 3$ amostras. Como o desempenho desta estrutura é semelhante ao do DFE₂, ela não mais foi considerada para os canais $\mathcal{H}_6$ e $\mathcal{H}_7$.

Equalizador	N	M_f	M_b	M_r
RNN ₁	2	5	-	-
RNN ₂	3	5	-	-
DFE ₁	-	20	9	-
DFE ₂	-	5	4	-
HIB	2	5	4	3

Tabela 4.2: Configurações dos equalizadores utilizados nas simulações.

Para o canal $\mathcal{H}_4$, obtém-se as curvas de taxa de erro de símbolo mostradas na Figura 4.7-a. Neste caso, o desempenho do equalizador híbrido se aproximou dos desempenhos das estruturas DFE₁ e DFE₂ enquanto as estruturas baseadas somente nas redes recorrentes obtiveram resultados inferiores. Para o canal $\mathcal{H}_5$ (Figura 4.7-b), porém, a situação se inverte, ou seja, o desempenho do híbrido fica mais próximo dos das estruturas RNN₁ e RNN₂, sendo melhor que as mesmas para valores de relação sinal-ruído maiores que 15 dB enquanto as estruturas baseadas somente no DFE apresentam taxas de erro bem maiores.

Observando a Figura 4.7 percebe-se que o equalizador híbrido funciona perto da melhor das estruturas que o compõem. Isso se repete quando se considera a parte real do canal de microondas: $\mathcal{H}_6$ (Figura 4.8-a). No entanto, quando se introduzem não-linearidades neste canal (Figura 4.8-b), o desempenho do equalizador híbrido é bem superior aos das demais estruturas consideradas.

Figura 4.7: Curvas de taxa de erro de símbolo considerando $\tau_d = 3$, 2-PAM, configurações dos equalizadores da Tabela 4.2 e canais a) $\mathcal{H}_4$ e b) $\mathcal{H}_5$.

Figura 4.8: Curvas de taxa de erro de símbolo considerando $\tau_d = 50$, 2-PAM, configurações dos equalizadores da Tabela 4.2 e canais a) $\mathcal{H}_6$ e b) $\mathcal{H}_7$.

4.2.3 Resultados das simulações com 4-QAM

Com o objetivo de verificar se o que foi observado nas simulações com o caso real se repetia para o caso complexo, foram considerados os canais $\mathcal{H}_{4c}$, $\mathcal{H}_{5c}$, $\mathcal{H}_{6c}$ e $\mathcal{H}_{7c}$ da Tabela 4.1 e a modulação 4-QAM. As configurações das estruturas utilizadas são as mesmas da Tabela 4.2.

Para o canal $\mathcal{H}_{4c}$ (Figura 4.9-a), o desempenho do equalizador híbrido se aproximou do desempenho do DFE₂ enquanto as redes recorrentes tiveram uma taxa de erros maior. Considerando não-linearidades neste canal (canal $\mathcal{H}_{5c}$), observa-se na Figura 4.9-b que o desempenho do híbrido ficou bem próximo do da RNN₂ enquanto as demais estruturas tiveram desempenhos inferiores. É interessante observar que as não-linearidades consideradas tornaram o canal extremamente difícil de equalizar e que o desempenho da estrutura RNN₂ foi sempre superior ao da RNN₁ e por isso descartamos esta última para as simulações com o canal de microondas.

Para o canal $\mathcal{H}_{6c}$, observa-se na Figura 4.10-a que os desempenhos do equalizador híbrido e do DFE₂ ficam bem próximos e são bem melhores que os resultados da rede recorrente. Incluindo não-linearidades neste canal (canal $\mathcal{H}_{7c}$) o híbrido passa a ter o melhor desempenho (Figura 4.10-b).

Figura 4.9: Curvas de taxa de erro de símbolo considerando $\tau_d = 3$, 4-QAM, configurações dos equalizadores da Tabela 4.2 e canais a) $\mathcal{H}_{4c}$ e b) $\mathcal{H}_{5c}$.

Figura 4.10: Curvas de taxa de erro de símbolo considerando $\tau_d = 50$, 4-QAM, configurações dos equalizadores da Tabela 4.2 e canais a) $\mathcal{H}_{6c}$ e b) $\mathcal{H}_{7c}$.

4.3 Conclusões a respeito do equalizador híbrido

Os resultados das simulações mostraram que na maioria das situações o equalizador híbrido funciona de forma semelhante à melhor das estruturas que o compõem. Em canais sem não-linearidades ele funcionou próximo do DFE e em canais não-lineares próximo da RNN. No entanto, quando se considerou um canal de microondas com não-linearidades, ele apresentou um desempenho superior.

Há também vantagens com relação à complexidade computacional. O híbrido formado a partir de um LTE e de uma RNN com 2 unidades possui uma complexidade equivalente à de uma RNN com 3 unidades. É como se o LTE substituísse uma unidade da RNN com a vantagem de proporcionar melhores resultados em algumas situações.

Ainda é necessário fazer testes com relação à sensibilidade ao atraso e também ao funcionamento *on line* da estrutura híbrida. Porém, os resultados obtidos até o momento mostram que a utilização desta estrutura pode ser vantajosa na equalização de canais de comunicação.

Capítulo 5

Conclusões

Sistemas não-lineares têm sido utilizados no problema de equalização durante décadas. No início dos anos 70, Forney demonstrou que o equalizador ótimo segundo o critério da máxima verossimilhança era uma estrutura não-linear. O equalizador bayesiano, por sua vez, segundo o critério de MAP é também um equalizador ótimo não-linear. Estes equalizadores ótimos, porém, possuem uma elevada complexidade computacional o que constitui um entrave ao seu uso na prática.

O DFE surgiu como uma solução não-linear simples. Formado basicamente pelo LTE com realimentação de decisões, este equalizador consegue bons desempenhos em canais em que apenas a utilização do LTE é insatisfatória. Por outro lado, há canais em que o DFE também não consegue bons resultados. Isso pode ser observado principalmente em canais que possuem zeros próximos ou sobre a circunferência unitária e também canais de satélite devido às não-linearidades dos amplificadores.

Diante disso, muitos autores começaram a estudar a utilização de outros sistemas não-lineares aplicados ao problema de equalização, como é o caso das redes neurais. Dessa forma, considerando a equalização como um problema de classificação, analisou-se inicialmente neste trabalho, o uso de três redes MLP, RBF e RNN.

Nesta dissertação foi feito um estudo comparativo destas três redes e de seus algoritmos de treinamento. Com relação aos algoritmos de treinamento pôde-se concluir o

seguinte:

- No treinamento de redes MLP, o algoritmo BP com *momentum* apresenta uma forte dependência das condições iniciais, uma velocidade de convergência lenta e complexidade computacional baixa. Os algoritmos baseados no método dos mínimos quadrados recursivo (RLS-I e RLS-II) por sua vez, possuem velocidade de convergência mais alta. Entretanto, a complexidade computacional deste tipo de algoritmo inviabiliza a sua utilização na prática. O algoritmo Acelerador tem um desempenho intermediário entre estes algoritmos e por isso seu uso se torna atraente, pelo menos para sinais binários. Entretanto, no domínio complexo ele possui muito mais operações que o conhecido CBP;
- No treinamento de redes do tipo RBF é importante utilizar uma técnica de análise de agrupamento a fim de adaptar os centros e variâncias das funções de base radial. Com a utilização dessa técnica é possível diminuir o número de neurônios, tornando a rede RBF com uma complexidade comparável às complexidades das outras redes. Considerando a adaptação dos pesos, o uso do algoritmo Acelerador apresentou vantagens quanto a velocidade de convergência e complexidade computacional. Deve-se porém, usar algum critério para a escolha dos parâmetros deste algoritmo, pois uma escolha incorreta pode levá-lo a situações de convergência lenta;
- Com relação às redes recorrentes, foi utilizado somente o algoritmo RTRL. Maiores detalhes na comparação de algoritmos de treinamento desta rede podem ser encontrados em [19].

Com relação ao estudo comparativo das redes quanto ao desempenho, pôde-se concluir o seguinte:

- Para a modulação 2-PAM, a RNN apresenta os melhores resultados, tanto do ponto de vista de complexidade computacional de seu algoritmo de treinamento

utilizado (RTRL) quanto de desempenho em termos de taxa de erro de símbolo. Para a modulação 4-QAM porém, seu desempenho para alguns canais é pior que o das redes MLP, RBF ou pior até mesmo que o DFE;

- A realimentação de decisões no caso de transmissão binária proporcionou melhoras significativas no desempenho de todas as redes, exceto no da RNN. Esta rede já faz realimentação de estados internos e a realimentação de decisões parece não acrescentar nenhuma informação nova. No caso complexo, porém, o uso de DF provocou uma deterioração no desempenho para algumas redes. Isso pode estar relacionado com a realimentação de erros e também com o atraso utilizado. Como vimos, a realimentação de decisões faz com que as estruturas se tornem mais sensíveis ao atraso. Uma vantagem aparente é que estruturas com DF tendem a ser menos complexas (com menos neurônios);
- O treinamento da rede recorrente é bem mais simples do que das redes MLP e RBF. Com relação à primeira, em algumas situações há uma certa dificuldade em se determinar sua configuração. Com relação à segunda, há uma certa dependência das condições iniciais dos centros e variâncias, mesmo com o uso de um algoritmo de adaptação.

Diante desses resultados e com o intuito de se obter uma estrutura que unisse características da RNN e do DFE foi proposto um equalizador híbrido. A partir dos resultados das simulações com este equalizador, pôde-se concluir que:

- Na maioria das situações o equalizador híbrido funciona de forma semelhante à melhor das estruturas que o compõem. No entanto, quando se considerou um canal de microondas com não-linearidades, ele apresentou um desempenho superior;
- Há também vantagens com relação à complexidade computacional. O equalizador híbrido formado a partir de um LTE e uma RNN com 2 unidades possui uma

complexidade equivalente à de uma RNN com 3 unidades. É como se o LTE substituísse uma unidade da RNN com a vantagem de proporcionar melhores resultados em algumas situações.

De modo geral, em situações em que uma estrutura não-linear simples, como o DFE, não funciona bem, se faz necessário utilizar uma estrutura não-linear mais complexa. Nessas situações, uma rede neural apresenta um desempenho mais interessante. Um equalizador híbrido, como o proposto, pode ser uma solução para se obter uma estrutura não-linear menos complexa com um algoritmo de treinamento relativamente simples que faz com que a estrutura híbrida preserve as boas características das que a compõem.

Nesta dissertação, a propagação de erros em estruturas que utilizam realimentação de decisões não foi abordada. Tal investigação extrapola o âmbito deste trabalho.

Referências Bibliográficas

- [1] HAYKIN, S. *Adaptive Filter Theory*. 3.ed. New Jersey, Prentice Hall, 1996.
- [2] MIRANDA, M. D.; GERKEN, M.; SILVA, M. T. M. Efficient implementation of error-feedback LSL algorithm. [Letter]. *Electronics Letters*, v.35, n.16, p.1308-1309, Aug. 1999.
- [3] LING, F.; MONOLAKIS, D.; PROAKIS, J. G. Numerically Robust Least-Squares Lattice-Ladder algorithms with direct updating of the reflection coefficients. *IEEE Transactions on Acoutics, Speech and Signal Processing*, v.ASSP-34, n.4, p.837-845, Aug. 1986.
- [4] PAIT, F. M. A tuner that accelerates parameters. *Systems and Control Letters*, v.35, p. 65-68, 1998.
- [5] JOJOA, P. E.; GERKEN, M.; PAIT, F. Um algoritmo acelerador de parâmetros. In: SIMPÓSIO BRASILEIRO DE TELECOMUNICAÇÕES, 17., Vila Velha, 1999. *Anais*. Vila Velha, Sociedade Brasileira de Telecomunicações / Universidade Federal do Espírito Santo, 1999. p.197-202.
- [6] GERKEN, M.; PAIT, F.; JOJOA, P. E. An Adaptive Filtering Algorithm with Parameter Acceleration. In: PROCEEDINGS OF THE IEEE ICASSP'2000, Istanbul, Turquia, 5 a 9 de julho de 2000.
- [7] GOMEZ P. E. J; GERKEN, M.; PAIT, F. The accelerating adaptive filtering algorithm. Submitted to IFAC Workshop on Adaptation and Learning in Control and Signal Processing, Aug. 29-31, 2001, Como, Itália.

- [8] HAYKIN, S. *Neural Networks*. 2.ed. New Jersey, Prentice Hall, 1999.
- [9] LEUNG, H.; HAYKIN, S. The complex backpropagation algorithm. *IEEE Transaction on Signal Processing*, v.39, n.9, Sept. 1991.
- [10] BENVENUTO, N.; PIAZZA, F. On the complex backpropagation algorithm. *IEEE Transactions on Signal Processing*, v.40, p.967-969, Apr. 1992.
- [11] BILSKI, J.; RUTKOWSKI, L. A fast training algorithm for neural networks. *IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing*, v.45, n.6, p.749-753, June 1998.
- [12] AZIMI-SADJADI, M. R.; LIOU, R. Fast learning process of Multilayer neural networks using recursive least squares method. *IEEE Transactions on Signal Processing*, v.40, n.2, Feb. 1992.
- [13] DINIZ, P. S. R. *Nonlinear adaptive filtering*. Espoo, Helsinki University of Technology / Laboratory of Telecommunications Technology, 1998. (Report)
- [14] MULGREW, B. Applying radial basis function. *IEEE Signal Processing Magazine*, v.13, p.50-65, Mar. 1996.
- [15] CHEN, S.; MULGREW, B.; GRANT, P. M. A clustering technique for digital communications channel equalization using radial basis function networks. *IEEE Transactions on Neural Networks*, v.4, n.4, p.570-579, July 1993.
- [16] CHEN, S.; MCLAUGHLIN, S.; MULGREW, B. Complex-valued radial basis function network, Part I: Network architecture and learning algorithms. *Signal Processing*, v.35, p.19-31, 1994.
- [17] CHEN, S.; MCLAUGHLIN, S.; MULGREW, B. Complex-valued radial basis function network, Part II: Application to digital communications channel equalisation. *Signal Processing*, v.36, p.175-188, 1994.

- [18] KECHRIOTIS, G.; ZERVAS, E.; MANOLAKOS, E. S. Using recurrent neural networks for adaptive communication channel equalization. *IEEE Transactions on Neural Networks*, v.5, p.267-278, Mar. 1994.
- [19] PEARLMUTTER, B. A. Gradient Calculations for Dynamic Recurrent Neural Networks: a Survey. *IEEE Transactions on Neural Networks*, v. 6, p. 1212-1228, Sep, 1995.
- [20] KECHRIOTS, G.; MANOLAKOS, E. S. Training fully recurrent neural networks with complex weights. *IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing*, v.41, n.3, Mar. 1994.
- [21] PROAKIS, J. G. *Digital Communications*, McGraw-Hill, 3.ed., 1996.
- [22] GIBSON, G. J.; COWAN, C. F. N.; GRANT, P. M. On the decision regions of multilayer perceptrons. *Proceedings of IEEE*, v.78, p.1590-1599, 1990.
- [23] FORNEY JR., G. D. Maximum-likelihood sequence estimation of digital sequences in the presence of intersymbol interference. *IEEE Transactions on Information Theory*, v.IT-18, n.3, p.363-378, May 1972.
- [24] FORNEY, G. D. The Viterbi Algorithm. *Proceedings of the IEEE*, v. 61, n. 3, Mar. 1973.
- [25] UNGERBOECK, G. Adaptive maximum-likelihood receiver for carrier-modulated data-transmission systems. *IEEE Transactions on Communications*, v. com-22, n. 5, p. 1128-1137, May 1974.
- [26] MAGEE JR., F. R.; PROAKIS, J. G. Adaptive maximum-likelihood sequence estimation for digital signaling in the presence of intersymbol interference. *IEEE Transactions on Information Theory*, p. 120-124, Jan. 1973.

- [27] VITERBI, A. J. Error bounds for convolutional codes and asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*. p. 260-269, Apr. 1967.
- [28] OMURA, J. K. On the Viterbi decoding algorithm. *IEEE Transactions on Information Theory*, p. 177-179, Jan. 1969.
- [29] FORNEY JR., G. D. Convolutional Codes II. Maximum-likelihood decoding, *Information and Control*, p. 222-266, July 1974.
- [30] FIGUEIRAS-VIDAL, A. R. *Digital Signal Processing in Telecommunications*. Great Britain, Spring Verlag, 1996.
- [31] QUERESHI, S. H. Adaptive equalization. *Proceedings of the IEEE*, v.73, n.9, p.1349-1387, Sep. 1985.
- [32] HONIG, M. L.; MESSERSCHMIT, D. G.; *Adaptive Filters*. Kluwer Academic Publishers.
- [33] CHEN, S.; GIBSON, G. J.; COWAN, C. F. N.; GRANT, P. M. Adaptive equalization of finite non-linear channels using multilayer perceptrons. *Signal Processing*, v.20, p.107-119, 1990.
- [34] DUDA, R. O.; HART, P. E. *Pattern Classification and Scene Analysis*, New York, Wiley, 1973.
- [35] MONTALVÃO FILHO, J. R.; DORIZZI, B.; MOTA, J. C. Some theoretical limits of efficiency of linear and nonlinear equalizers. *Revista da Sociedade Brasileira de Telecomunicações*, v. 14, n.2, Dez. 1999.
- [36] MONTALVÃO FILHO, J. R.; MOTA, J. C. M.; DORIZZI, B.; CAVALCANTI, C. C. Reducing bayes equalizer complexity: a new approach for clusters determination. In: *Proceedings of SBT/IEEE International Telecommunications Symposium*, v.2, p.428-433, São Paulo, 9 a 13 de agosto de 1998.

- [37] IBNKAHLA, M. Applications of neural networks to digital communications - a survey. *Signal Processing*, v. 80, p.1185-1215, 2000.
- [38] DESTRO FILHO, J. B. *Egalisation aveugle de canaux de communication à l'aide d'algorithmes de Bussgang de réseaux neuronaux*. Nice, 1998. 261p. Thèse de Docteur en Sciences - Ecole Doctorale "Sciences Pour l'Ingénieur", Université de Nice-Sofhia Antipolis.
- [39] GIBSON, G. J.; SIU, S.; COWAN, C. F. N. The application of nonlinear structures to the reconstruction of binary signals. *IEEE Transactions on Signal Processing*, v. 39, p.1877-1884, Aug.1991.
- [40] CHA, I.; KASSAN, S. A. Channel equalization using adaptive complex radial basis function networks. *IEEE Journal on Selected Areas in Communications*, v. 13, n. 1, p. 122-131, Jan. 1995.
- [41] PARISI, R.; CLAUDIO, D. D.; ORLANDI, G.; RAO, B. D. Fast adaptive equalization by recurrent neural networks. *IEEE Transactions on Signal Processing*, v.45, n.11, p.2731-2739, Nov. 1997.
- [42] CHEN, S.; GIBSON, G. J.; COWAN, C. F. N.; GRANT, P. M. Reconstruction of binary signals using an adaptive radial-basis-function equalizer. *Signal Processing*, v.22, p.77-93, 1991.
- [43] CHANG, C.; SIU, S.; WEI, C. Decision feedback equalization using complex backpropagation algorithm. In: *IEEE International Symposium on Circuits and Systems*, Hong Kong, June 9-12, 1997. *Anais*. p.589-592.
- [44] LEE, W.; PEARSON, J. A hybrid linear/nonlinear approach to channel equalization properties. In: HANSON, S.; COWAN, J. D.; GILES, C. L., eds. *Advances in Neural Information Processes Systems 5*, San Mateo, Morgan Kauffmann, 1993, p.674-681.

- [45] BOUCHIRED, S.; ROVIRAS, D.; CASTANI, F. Equalisation of satellite mobile channels with neural network techniques. *Space Communications*, v. 15, p. 209-220, 1998/1999.
- [46] TREICHLER, J. R.; FIJALKOW, I.; JOHNSON Jr., C. R.; Fractionally spaced equalizers. *IEEE Signal Processing Magazine*, p.65-81, May 1996.
- [47] SIGNAL PROCESSING SOCIETY and NATIONAL SCIENCE FOUNDATION. The Signal Processing Information Base (SPIB). Arquivo de dados (chan1.mat) disponível na Internet: <http://spib.rice.edu/spib/microwave.html> [14 fev. 2001].
- [48] MARPLE JR., S. L. *Digital Spectral Analysis with applications*, 1.ed. New Jersey, Prentice Hall, 1987.
- [49] LIU, H.; ZOLTOWSKI, M. D. Blind equalization in antenna array CDMA systems. *IEEE Transactions on Signal Processing*, v. 45, p. 161-172, Jan, 1997.
- [50] HECHT-NIELSEN, R. *Neurocomputing*. New York, Addison Wesley Publ. Co., 1990.
- [51] SILVA, M. T. M.; GERKEN, M. Um algoritmo acelerador para treinamento de redes neurais do tipo MLP. In: *Simpósio Brasileiro de Telecomunicações*, 18., Gramado, 2000. Anais. Gramado, Sociedade Brasileira de Telecomunicações / PUC-RS, 2000.
- [52] SILVA, M. T. M.; GERKEN, M; MIRANDA, M. D. Estruturas não-lineares aplicadas à recuperação de sinais binários. In: *Simpósio Brasileiro de Telecomunicações*, 18., Gramado, 2000. Anais. Gramado, Sociedade Brasileira de Telecomunicações / PUC-RS, 2000.
- [53] GITLIN, R. D.; WEINSTEIN, S. B. Fractionally-spaced equalization: an improved digital transversal equalizer. *The Beel System Technical Journal*, p. 275-296, Feb. 1981.

Apêndice A

Lema de inversão de matrizes

Seja:

- $\mathbf{A}$ uma matriz não-singular de dimensões $N \times N$,
- $\mathbf{C}$ uma matriz não-singular de dimensões $M \times M$,
- $\mathbf{B}$ uma matriz de dimensões $N \times M$ e
- $\mathbf{D}$ uma matriz de dimensões $M \times N$,

então a inversa da matriz aumentada $\mathbf{A} + \mathbf{BCD}$ é dada por

$$(\mathbf{A} + \mathbf{BCD})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}(\mathbf{DA}^{-1}\mathbf{B} + \mathbf{C}^{-1})^{-1}\mathbf{DA}^{-1}, \quad (\text{A.1})$$

assumindo que a inversa de $\mathbf{DA}^{-1}\mathbf{B} + \mathbf{C}^{-1}$ exista. Esta igualdade é conhecida como lema de inversão da matriz aumentada [48].

Há ainda duas outras formas de interesse. Uma delas é quando a matriz $\mathbf{C}$ é um escalar k (1×1), $\mathbf{B}$ é um vetor coluna $\mathbf{v}$ ($N \times 1$) e $\mathbf{D}$ é um vetor linha conjugado $\mathbf{w}^H$ ($1 \times N$), então

$$(\mathbf{A} + k\mathbf{vw}^H)^{-1} = \mathbf{A}^{-1} - \frac{k(\mathbf{A}^{-1}\mathbf{v})(\mathbf{w}^H\mathbf{A}^{-1})}{1 + k\mathbf{w}^H\mathbf{A}^{-1}\mathbf{v}} \quad (\text{A.2})$$

é a inversa da matriz formada somando-se a matriz $k\mathbf{vw}^H$ à matriz $\mathbf{A}$.

A outra forma de interesse é quando $\mathbf{A} = \mathbf{E}^{-1}$, $\mathbf{D} = \mathbf{B}^H$, $\mathbf{C} = \mathbf{F}^{-1}$. Assim, a inversa da matriz $\mathbf{E}^{-1} + \mathbf{B}\mathbf{F}^{-1}\mathbf{B}^H$ é dada por

$$(\mathbf{E}^{-1} + \mathbf{B}\mathbf{F}^{-1}\mathbf{B}^H)^{-1} = \mathbf{E} - \mathbf{E}\mathbf{B}(\mathbf{B}^H\mathbf{E}\mathbf{B} + \mathbf{F})^{-1}\mathbf{B}^H\mathbf{E}. \quad (\text{A.3})$$

Essa última igualdade é chamada na referência [1] de lema de inversão de matrizes. Nesta referência as matrizes $\mathbf{E}$ e $\mathbf{F}$ são consideradas positivas definidas, garantindo dessa forma a existência da inversa da matriz $\mathbf{B}^H\mathbf{E}\mathbf{B} + \mathbf{F}$. Essa consideração é muito restritiva e se aplica bem quando se deseja inverter matrizes de autocorrelação. Para um caso mais genérico, é necessário que $\mathbf{E}$ e $\mathbf{F}$ tenham inversa e sejam tais que a inversa de $\mathbf{B}^H\mathbf{E}\mathbf{B} + \mathbf{F}$ exista.

Apêndice B

Algoritmo RLS-I

Em implementações recursivas do método dos mínimos quadrados, introduz-se uma medida de desempenho

$$Q(n) = \sum_{m=1}^n \lambda^{n-m} \sum_{k=1}^{N_L} [e_k^{(L)}(m)]^2, \quad (\text{B.1})$$

que pode ser escrita de outra forma

$$Q(n) = \sum_{m=1}^n \lambda^{n-m} \sum_{k=1}^{N_L} [d_k^{(L)}(m) - \varphi(\mathbf{y}^{(L-1)T}(m) \mathbf{w}_k^{(L)}(n))]^2, \quad (\text{B.2})$$

sendo λ o fator de esquecimento, $0 \ll \lambda < 1$. Os sinais que aparecem nas equações anteriores estão definidos na Seção 2.3.2.

O índice de desempenho (B.2) é então minimizado tomando-se sua derivada parcial em relação a $\mathbf{w}_i^{(l)}(n)$ e igualando a zero:

$$\begin{aligned} \frac{\partial Q(n)}{\partial \mathbf{w}_i^{(l)}(n)} &= 2 \sum_{m=1}^n \lambda^{n-m} \sum_{k=1}^{N_L} \frac{\partial e_k^{(L)}(m)}{\partial \mathbf{w}_i^{(l)}(n)} e_k^{(L)}(m) = \\ &= -2 \sum_{m=1}^n \lambda^{n-m} \sum_{k=1}^{N_L} \frac{\partial y_k^{(L)}}{\partial \mathbf{w}_i^{(l)}} e_k^{(L)}(m) = 0. \end{aligned} \quad (\text{B.3})$$

É importante lembrar que a derivada do escalar $y_k^{(L)}$ em relação ao vetor $\mathbf{w}_i^{(l)}$ gera como resultado um outro vetor de mesma dimensão do anterior. Cada posição desse novo

vetor contém a derivada do escalar $y_k^{(L)}$ em relação a cada elemento do vetor $\mathbf{w}_i^{(l)}$.

$$\begin{aligned}
\sum_{m=1}^n \lambda^{n-m} \sum_{k=1}^{N_L} \frac{\partial y_k^{(L)}(m)}{\partial \mathbf{w}_i^{(l)}(n)} e_k^{(L)}(m) &= \\
&= \sum_{m=1}^n \lambda^{n-m} \sum_{k=1}^{N_L} \frac{\partial y_k^{(L)}(m)}{\partial v_k^{(L)}(m)} \sum_{p=1}^{N_{L-1}} \frac{\partial v_k^{(L)}(m)}{\partial y_p^{(L-1)}(m)} \frac{\partial y_p^{(L-1)}(m)}{\partial \mathbf{w}_i^{(m)}} e_k^{(L)}(m) = \\
&= \sum_{m=1}^n \lambda^{n-m} \sum_{p=1}^{N_{L-1}} \frac{\partial y_p^{(L-1)}(m)}{\partial \mathbf{w}_i^{(l)}(n)} \sum_{k=1}^{N_L} \frac{\partial y_k^{(L)}(m)}{\partial v_k^{(L)}(m)} \mathbf{w}_{kp}^{(L)}(n) e_k^{(L)}(m) = \\
&= \sum_{m=1}^n \lambda^{n-m} \sum_{p=1}^{N_{L-1}} \frac{\partial y_p^{(L-1)}(m)}{\partial \mathbf{w}_i^{(l)}(n)} e_p^{(L-1)}(m) = 0.
\end{aligned} \tag{B.4}$$

Esta equação pode então ser generalizada para qualquer camada l da seguinte forma

$$\sum_{m=1}^n \lambda^{n-m} \sum_{q=1}^{N_l} \frac{\partial y_q^{(l)}(m)}{\partial \mathbf{w}_i^{(l)}(n)} e_q^{(l)}(m) = 0, \tag{B.5}$$

sendo

$$e_q^{(l)}(m) = \sum_{k=1}^{N_{k+1}} \frac{\partial y_k^{(l+1)}(m)}{\partial v_k^{(l+1)}(m)} \mathbf{w}_{kq}^{(l+1)}(n) e_k^{(l+1)}(m), \tag{B.6}$$

o erro propagado regressivamente.

$$\begin{aligned}
\sum_{m=1}^n \lambda^{n-m} \sum_{q=1}^{N_k} \frac{\partial y_q^{(l)}(m)}{\partial \mathbf{w}_i^{(l)}(n)} e_q^{(l)}(m) &= \\
&= \sum_{m=1}^n \lambda^{n-m} \sum_{q=1}^{N_k} \frac{\partial y_q^{(l)}(m)}{\partial v_q^{(l)}(m)} \frac{\partial v_q^{(l)}(m)}{\partial \mathbf{w}_i^{(l)}(n)} e_q^{(l)}(m)
\end{aligned} \tag{B.7}$$

A segunda somatória da equação (B.7) só é diferente de zero para $q = i$, pois a derivada parcial é em relação a $\mathbf{w}_i^{(l)}$, ou seja

$$\begin{aligned}
\sum_{m=1}^n \lambda^{n-m} \frac{\partial y_i^{(l)}(m)}{\partial v_i^{(l)}(m)} \mathbf{y}^{(l-1)T}(m) e_i^{(l)}(m) &= \\
&= \sum_{m=1}^n \lambda^{n-m} \frac{\partial y_i^{(l)}(m)}{\partial v_i^{(l)}(m)} \mathbf{y}^{(l-1)T}(m) [d_i^{(l)}(m) - y_i^{(l)}(m)] = 0.
\end{aligned} \tag{B.8}$$

Definindo

$$\varphi(b_i^{(l)}(m)) = d_i^{(l)}(m), \tag{B.9}$$

lembrando que

$$\varphi(v_i^{(l)}(m)) = y_i^{(l)}(m), \quad (\text{B.10})$$

com

$$v_i^{(l)}(m) = \mathbf{y}^{(l-1)T}(m) \mathbf{w}_i^{(l)}(n) \quad (\text{B.11})$$

e assumindo a aproximação

$$\varphi(b_i^{(l)}(m)) \approx \varphi(v_i^{(l)}(m)) + \varphi'(v_i^{(l)}(m))(b_i^{(l)}(m) - v_i^{(l)}(m)), \quad (\text{B.12})$$

chega-se nas equações (B.13) e (B.14), fazendo as devidas substituições na Equação (B.8):

$$\sum_{m=1}^n \lambda^{n-m} \varphi'(v_i^{(l)}(m)) \mathbf{y}^{(l-1)T}(m) \cdot [\varphi(v_i^{(l)}(m)) + \varphi'(v_i^{(l)}(m))(b_i^{(l)}(m) - v_i^{(l)}(m)) - \varphi(v_i^{(l)}(m))] = 0 \quad (\text{B.13})$$

$$\sum_{m=1}^n \lambda^{n-m} [\varphi'(v_i^{(l)}(m))]^2 [b_i^{(l)}(m) - \mathbf{y}^{(l-1)T}(m) \mathbf{w}_i^{(l)}(n)] \mathbf{y}^{(l-1)T}(m) = 0. \quad (\text{B.14})$$

Escrevendo esta última equação na forma vetorial, obtém-se

$$\mathbf{R}_i^{(l)}(n) \mathbf{w}_i^{(l)}(n) = \mathbf{r}_i^{(l)}(n), \quad (\text{B.15})$$

sendo

$$\mathbf{R}_i^{(l)}(n) = \sum_{m=1}^n \lambda^{n-m} [\varphi'(v_i^{(l)}(m))]^2 \mathbf{y}^{(l-1)}(m) \mathbf{y}^{(l-1)T}(m) \quad (\text{B.16})$$

e

$$\mathbf{r}_i^{(l)}(n) = \sum_{m=1}^n \lambda^{n-m} [\varphi'(v_i^{(l)}(m))]^2 b_i^{(l)}(m) \mathbf{y}^{(l-1)T}(m). \quad (\text{B.17})$$

Estas equações podem ser rescritas de outra forma

$$\mathbf{R}_i^{(l)}(n) = \lambda \mathbf{R}_i^{(l)}(n-1) + [\varphi'(v_i^{(l)}(n))]^2 \mathbf{y}^{(l-1)}(n) \mathbf{y}^{(l-1)T}(n) \quad (\text{B.18})$$

e

$$\mathbf{r}_i^{(l)}(n) = \lambda \mathbf{r}_i^{(l)}(n-1) + [\varphi'(v_i^{(l)}(n))]^2 \mathbf{y}^{(l-1)}(n) b_i^{(l)}(n). \quad (\text{B.19})$$

Para obter a matriz $\mathbf{P}_i^{(l)}(n) = [\mathbf{R}_i^{(l)}(n)]^{-1}$, aplica-se o lema de inversão de matrizes apresentado no Apêndice A, chegando-se no seguinte resultado:

$$\mathbf{P}_i^{(l)}(n) = \lambda^{-1} [\mathbf{I} - \varphi'(v_i^{(l)}(n)) \mathbf{g}_i^{(l)}(n) \mathbf{y}^{(l-1)T}(n)] \mathbf{P}_i^{(l)}(n-1), \quad (\text{B.20})$$

sendo

$$\mathbf{g}_i^{(l)}(n) = \frac{\varphi'(v_i^{(l)}(n)) \mathbf{P}_i^{(l)}(n-1) \mathbf{y}^{(l-1)}(n)}{\lambda + [\varphi'(v_i^{(l)}(n))]^2 \mathbf{y}^{(l-1)T}(n) \mathbf{P}_i^{(l)}(n-1) \mathbf{y}^{(l-1)}(n)}. \quad (\text{B.21})$$

A partir de (B.21) e (B.20) chega-se ainda em

$$\mathbf{g}_i^{(l)}(n) = \varphi'(v_i^{(l)}(n)) \mathbf{P}_i^{(l)}(n) \mathbf{y}^{(l-1)}(n). \quad (\text{B.22})$$

Com (B.19) obtém-se a seguinte expressão para a solução de (B.15):

$$\begin{aligned} \mathbf{w}_i^{(l)}(n) &= (\mathbf{R}_i^{(l)}(n))^{-1} \mathbf{r}_i^{(l)}(n) = \\ &= \lambda \mathbf{P}_i^{(l)}(n) \mathbf{r}_i^{(l)}(n-1) + [\varphi'(v_i^{(l)}(n))]^2 \mathbf{P}_i^{(l)}(n) \mathbf{y}^{(l-1)}(n) b_i^{(l)}(n). \end{aligned} \quad (\text{B.23})$$

Com (B.20) e (B.22) esta expressão resulta em

$$\begin{aligned} \mathbf{w}_i^{(l)}(n) &= [\mathbf{I} - \varphi'(v_i^{(l)}(n)) \mathbf{g}_i^{(l)}(n) \mathbf{y}^{(l-1)T}(n)] \mathbf{w}_i^{(l)}(n-1) + \\ &\quad + [\varphi'(v_i^{(l)}(n))]^2 \mathbf{P}_i^{(l)}(n) \mathbf{y}^{(l-1)}(n) b_i^{(l)}(n) = \\ &= \mathbf{w}_i^{(l)}(n-1) + \mathbf{g}_i^{(l)}(n) \varphi'(v_i^{(l)}(n)) [b_i^{(l)} - \mathbf{y}^{(l-1)T}(n) \mathbf{w}_i^{(l)}(n-1)]. \end{aligned} \quad (\text{B.24})$$

Como $e_i^{(l)}(n) = d_i^{(l)}(n) - y_i^{(l)}(n) = \varphi(b_i^{(l)}(n)) - \varphi(v_i^{(l)}(n))$, das equações (B.12) e (B.24) chega-se finalmente em

$$\mathbf{w}_i^{(l)}(n) = \mathbf{w}_i^{(l)}(n-1) + \mathbf{g}_i^{(l)}(n) e_i^{(l)}(n). \quad (\text{B.25})$$

Para obter uma versão do algoritmo que garanta a simetria da matriz de autocorrelação inversa, vamos definir a seguinte variável

$$\alpha_i^{(l)}(n) = \lambda + \tilde{\mathbf{y}}_i^{(l-1)T}(n) \mathbf{g}_{\mathbf{m}_i}^{(l)}(n), \quad (\text{B.26})$$

sendo

$$\tilde{\mathbf{y}}_i^{(l-1)} = \varphi'(v_i^{(l-1)}(n))\mathbf{y}_i^{(l-1)}(n) \quad (\text{B.27})$$

e $\mathbf{g}_m$ o ganho de Kalman modificado, definido como

$$\mathbf{g}_{m_i}^{(l)}(n) = \mathbf{P}_i^{(l)}(n-1)\tilde{\mathbf{y}}_i^{(l-1)}(n), \quad (\text{B.28})$$

ou seja

$$\mathbf{g}_{m_i}^{(l)}(n) = \alpha_i^{(l)}(n)\mathbf{g}_i^{(l)}(n). \quad (\text{B.29})$$

Partindo da Equação (B.20) e utilizando as definições feitas anteriormente, obtém-se:

$$\mathbf{P}_i^{(l)}(n) = \lambda^{-1} \left[\mathbf{P}_i^{(l)}(n-1) - \frac{\mathbf{g}_{m_i}^{(l)}(n)\mathbf{g}_{m_i}^{(l)T}(n)}{\alpha_i^{(l)}(n)} \right] \quad (\text{B.30})$$

e

$$\mathbf{w}_i^{(l)}(n) = \mathbf{w}_i^{(l)}(n-1) + \mathbf{g}_{m_i}^{(l)}(n) \frac{e_i^{(l)}(n)}{\alpha_i^{(l)}(n)}. \quad (\text{B.31})$$

As equações (B.6), (B.27), (B.28), (B.30) e (B.31) definem o algoritmo RLS-I que está apresentado na Tabela 2.7. Deve-se notar que para facilitar a notação, o ganho de Kalman modificado foi representado nesta tabela por $\mathbf{g}$ ao invés de $\mathbf{g}_m$.

Os resultados de simulações evidenciam que algoritmos do tipo RLS que garantem a simetria da matriz de autocorrelação inversa apresentam maior estabilidade numérica. Por isso foi feita esta modificação no algoritmo proposto originalmente em [11].

Apêndice C

Algoritmo Acelerador Real

C.1 O algoritmo Acelerador real de tempo contínuo modificado

Como pode ser visto em [4], [5] e [6], o algoritmo acelerador de tempo contínuo é descrito pelas seguintes equações:

$$\dot{\mathbf{w}}(t) = \mathbf{q}(t) \quad (\text{C.1})$$

$$\begin{aligned} \dot{\mathbf{q}}(t) = & -\mathbf{M}_1 \mathbf{u}(t) e(t) + \\ & -2\mathbf{M}_1 (\mathbf{M}_2 + \mathbf{u}(t) \mathbf{u}^T(t) \mathbf{M}_1 \mathbf{M}_3) \mathbf{q}(t) \end{aligned} \quad (\text{C.2})$$

$$e(t) = \mathbf{u}^T(t) \mathbf{w}(t) - d(t). \quad (\text{C.3})$$

Nestas equações, $\mathbf{u}(t)$ corresponde ao sinal de entrada (vetor regressor), $\mathbf{w}(t)$ é o vetor de parâmetros, $d(t)$ o sinal desejado, $e(t)$ o erro de predição e $\mathbf{q}(t)$ a derivada do vetor de parâmetros. Todos os vetores tem mesma dimensão N_0 .

As matrizes $\mathbf{M}_1$, $\mathbf{M}_2$ e $\mathbf{M}_3$ têm dimensão $N_0 \times N_0$, são simétricas e positivas definidas. Estas expressões podem ser implementadas com N_0 integradores sendo que na prática as matrizes $\mathbf{M}_i$, $i = 1, 2, 3$, são tomadas proporcionais à matriz identidade, o que diminui consideravelmente a complexidade da implementação [5]. Além disso, conforme [4] e [6], este sistema é estável segundo o critério de Liapunov quando as

matrizes $\mathbf{M}_1$, $\mathbf{M}_2$ e $\mathbf{M}_3$ satisfizerem as seguintes condições:

$$4\mathbf{M}_1\mathbf{M}_3\mathbf{M}_1 \geq \mathbf{M}_2^{-1} \quad (\text{C.4})$$

$$\mathbf{M}_2\mathbf{M}_1\mathbf{M}_3 + \mathbf{M}_3\mathbf{M}_1\mathbf{M}_2 \geq \mathbf{M}_1^{-1}/2. \quad (\text{C.5})$$

Introduzindo uma função não-linear $\varphi(\cdot)$ na saída, a Equação (C.3) resulta em

$$e(t) = \varphi(\mathbf{u}^T(t)\mathbf{w}(t)) - d(t). \quad (\text{C.6})$$

Dessa forma, as equações (C.1), (C.2) e (C.6) descrevem o algoritmo Acelerador de tempo contínuo modificado e serão usadas na seção seguinte para obtenção das equações progressivas do algoritmo de treinamento de uma rede neuronal do tipo MLP.

C.2 O algoritmo Acelerador real para treinamento de redes MLP

C.2.1 Cálculo progressivo

Para obter as equações do cálculo progressivo do algoritmo Acelerador para treinamento de uma rede MLP, basta considerar inicialmente apenas um neurônio de uma camada l , cujo esquema pode ser visto na Figura C.1. Além disso, considera-se ainda a camada l como a de saída de modo que o erro seja calculado através da comparação do sinal desejado com a saída da rede. Para simplificar a notação, os índices da camada e do neurônio serão omitidos nesta dedução, ou seja $w_{i1}^{(k)}(n) = w(n)$.

Discretizando as equações (C.1), (C.2) e (C.6) pelo método de Euler reverso ($f(n) = f(n-1) + Tg(n)$, sendo $\dot{f}(t) = g(t)$) e considerando $T = \mu$, resulta:

$$\mathbf{w}(n) = \mathbf{w}(n-1) + \mu\mathbf{q}(n) \quad (\text{C.7})$$

$$\begin{aligned} \mathbf{q}(n) = & \mathbf{q}(n-1) - \mu\mathbf{M}_1\{\mathbf{u}(n)e(n) + \\ & + 2(\mathbf{M}_2 + \mathbf{u}(n)\mathbf{u}^T(n)\mathbf{M}_1\mathbf{M}_3)\mathbf{q}(n)\} \end{aligned} \quad (\text{C.8})$$

$$e(n) = \varphi(\mathbf{u}^T(n)\mathbf{w}(n)) - d(n). \quad (\text{C.9})$$

Figura C.1: Modelo do k-ésimo neurônio da camada l .

Substituindo (C.7) em (C.9), obtém-se

$$e(n) = \varphi(\mathbf{u}^T(n)\mathbf{w}(n-1) + \mu\mathbf{u}^T(n)\mathbf{q}(n)) - d(n). \quad (\text{C.10})$$

Expandindo $\varphi(v)$, com $v = \mathbf{u}^T(n)\mathbf{w}(n-1) + \mu\mathbf{u}^T(n)\mathbf{q}(n)$, em torno de $v_0 = \mathbf{u}^T(n)\mathbf{w}(n-1)$ por meio da série de Taylor e mantendo-se somente os dois termos de menor ordem da série, obtém-se

$$\varphi(v_0 + \mu\mathbf{u}^T(n)\mathbf{q}(n)) \approx \varphi(v_0) + \varphi'(v_0)(\mu\mathbf{u}^T(n)\mathbf{q}(n)). \quad (\text{C.11})$$

Substituindo (C.11) em (C.10) resulta então

$$e(n) = \varphi(\mathbf{u}^T(n)\mathbf{w}(n-1)) + \varphi'(\mathbf{u}^T(n)\mathbf{w}(n-1))(\mu\mathbf{u}^T(n)\mathbf{q}(n)) - d(n). \quad (\text{C.12})$$

As equações (C.8) e (C.12) não definem um algoritmo computável. Este problema pode ser resolvido introduzindo-se um erro *a priori* $e_a(n)$ definido como

$$e_a(n) = \varphi(\mathbf{u}^T(n)\mathbf{w}(n-1)) - d(n). \quad (\text{C.13})$$

Assim pode-se rescrever a Equação (C.12) como

$$e(n) = e_a(n) + \varphi'(\mathbf{u}^T(n)\mathbf{w}(n-1))\mu\mathbf{u}^T(n)\mathbf{q}(n) \quad (\text{C.14})$$

e substituir este resultado na Equação (C.8), resultando

$$\mathbf{q}(n) = \Gamma^{-1}(n)(\mathbf{q}(n-1) - \mu\mathbf{M}_1\mathbf{u}(n)e_a(n)), \quad (\text{C.15})$$

sendo

$$\begin{aligned}\Gamma(n) = \{ & \mathbf{I} + \mu^2 \varphi'(\mathbf{u}^T(n) \mathbf{w}(n-1)) \mathbf{M}_1 \mathbf{u}(n) \mathbf{u}^T(n) + \\ & + 2\mu \mathbf{M}_1 (\mathbf{M}_2 + \mathbf{u}(n) \mathbf{u}^T(n) \mathbf{M}_1 \mathbf{M}_3) \}.\end{aligned}\quad (\text{C.16})$$

As equações (C.13), (C.16), (C.15), (C.7) e (C.9) constituem um algoritmo computável, cuja complexidade computacional pode ser reduzida considerando matrizes $\mathbf{M}_i$ diagonais, isto é, $\mathbf{M}_i = m_i \mathbf{I}$, $i = 1, 2, 3$. Dessa forma, pode-se rescrever as equações (C.15) e (C.16) como

$$\mathbf{q}(n) = \Gamma^{-1}(n) (\mathbf{q}(n-1) - \mu m_1 e_a(n) \mathbf{u}(n)) \quad (\text{C.17})$$

e

$$\Gamma(n) = \alpha \mathbf{I} + b(n) \mathbf{u}(n) \mathbf{u}^T(n), \quad (\text{C.18})$$

sendo

$$\alpha = 1 + 2\mu m_1 m_2 \quad (\text{C.19})$$

e

$$b(n) = \mu^2 m_1 \varphi'(\mathbf{u}^T(n) \mathbf{w}(n-1)) + 2\mu m_1^2 m_3. \quad (\text{C.20})$$

Para obter a inversa da matriz $\Gamma(n)$ aplicou-se na Equação (C.18) o lema de inversão de matrizes apresentado no Apêndice A, chegando-se no seguinte resultado:

$$\Gamma^{-1}(n) = \frac{1}{\alpha} \left[\mathbf{I} - \frac{b(n)}{\alpha + b(n) \|\mathbf{u}(n)\|^2} \mathbf{u}(n) \mathbf{u}^T(n) \right]. \quad (\text{C.21})$$

Substituindo este resultado na Equação (C.17) obtém-se

$$\mathbf{q}(n) = \frac{1}{\alpha} \mathbf{q}(n-1) - \frac{1}{\alpha} \left[\frac{b(n) \mathbf{u}^T(n) \mathbf{q}(n-1) + \mu m_1 \alpha e_a(n)}{\alpha + b(n) \|\mathbf{u}(n)\|^2} \right] \mathbf{u}(n), \quad (\text{C.22})$$

ou seja

$$\mathbf{q}(n) = \frac{1}{\alpha} \mathbf{q}(n-1) - \gamma(n) \mathbf{u}(n), \quad (\text{C.23})$$

sendo

$$\gamma(n) = \frac{1}{\alpha} \left[\frac{b(n)\mathbf{u}^T(n)\mathbf{q}(n-1) + \mu m_1 \alpha e_a(n)}{\alpha + b(n)\|\mathbf{u}(n)\|^2} \right]. \quad (\text{C.24})$$

Das equações anteriores, observa-se que, quando $\mathbf{M}_i = m_i \mathbf{I}$, $i = 1, 2, 3$, o cálculo da matriz $\Gamma^{-1}(n)$ se reduz ao cálculo de um escalar $\gamma(n)$, chegando-se num conjunto de equações que exigem um menor esforço computacional. As equações aqui obtidas, podem ser estendidas facilmente para um neurônio k da camada l , como pode ser visto na Tabela 2.10 da seção 2.3.4.

C.2.2 Cálculo regressivo

Para obter as equações do cálculo regressivo, deve-se considerar toda a estrutura da rede. Considerou-se uma rede com L camadas, sendo N_l , o número de neurônios da camada l , $w_{ij}^{(l)}$ o peso que liga o neurônio i da camada l ao neurônio j da camada $l-1$, $y_i^{(l)}(n) = \varphi(v_{a,i}^{(l)}(n))$, com $v_{a,i}^{(l)}(n) = \mathbf{y}^{(l-1)T}(n)\mathbf{w}_i^{(l)}(n-1)$, a saída do neurônio i da camada l e por conseguinte a k -ésima entrada da camada $l+1$.

Para se propagar regressivamente o erro da camada de saída de uma MLP, introduz-se uma medida de desempenho

$$\xi(n) = \frac{1}{2} \sum_{j=1}^{N_L} (e_{a,j}^{(L)}(n))^2, \quad (\text{C.25})$$

que deve ser minimizada. Para isso, toma-se sua derivada parcial em relação a $\mathbf{w}_i^{(k)}(n-1)$ e iguala-se a zero, ou seja

$$\begin{aligned} \frac{\partial \xi(n)}{\partial \mathbf{w}_i^{(k)}(n-1)} &= \sum_{j=1}^{N_L} \frac{\partial e_{a,j}^{(L)}(n)}{\partial \mathbf{w}_i^{(k)}(n-1)} e_{a,j}^{(L)}(n) = \\ &= \sum_{j=1}^{N_L} \frac{\partial y_j^{(L)}(n)}{\partial \mathbf{w}_i^{(k)}(n-1)} e_{a,j}^{(L)}(n) = 0. \end{aligned} \quad (\text{C.26})$$

Esta última expressão pode ser rescrita como

$$\begin{aligned} \sum_{j=1}^{N_L} \frac{\partial y_j^{(L)}(n)}{\partial v_{a,j}^{(L)}(n)} \sum_{p=1}^{N_{L-1}} \frac{\partial v_{a,j}^{(L)}(n)}{\partial y_p^{(L-1)}(n)} \frac{\partial y_p^{(L-1)}(n)}{\partial \mathbf{w}_i^{(k)}(n-1)} e_{a,j}^{(L)}(n) = \\ = \sum_{p=1}^{N_{L-1}} \frac{\partial y_p^{(L-1)}(n)}{\partial \mathbf{w}_i^{(k)}(n-1)} \sum_{j=1}^{N_L} \frac{\partial y_j^{(L)}(n)}{\partial v_{a,j}^{(L)}(n)} w_{jp}^{(L)}(n-1) e_{a,j}^{(L)}(n) = 0. \end{aligned} \quad (\text{C.27})$$

A comparação desta última expressão com a Equação (C.26) mostra que os erros *a priori* da camada $L - 1$ podem ser obtidos pela expressão

$$e_{a,p}^{(L-1)}(n) = \sum_{j=1}^{N_L} \frac{\partial y_j^{(L)}(n)}{\partial v_{a,j}^{(L)}(n)} w_{jp}^{(L)}(n-1) e_{a,j}^{(L)}(n). \quad (\text{C.28})$$

Assim pode-se rescrever a equação (C.27) da seguinte forma

$$\sum_{p=1}^{N_{L-1}} \frac{\partial y_p^{(L-1)}(n)}{\partial \mathbf{w}_i^{(k)}(n-1)} e_{a,p}^{(L-1)}(n) = 0. \quad (\text{C.29})$$

Este procedimento (a partir de (C.26)) pode ser repetido para a camada anterior e assim por diante. Desta forma para a camada $l = k$ obtém-se

$$\sum_{q=1}^{N_k} \frac{\partial y_q^{(k)}(n)}{\partial \mathbf{w}_i^{(k)}(n-1)} e_{a,q}^{(k)}(n) = 0, \quad (\text{C.30})$$

com

$$e_{a,q}^{(k)}(n) = \sum_{j=1}^{N_{k+1}} \underbrace{\frac{\partial y_j^{(k+1)}(n)}{\partial v_{a,j}^{(k+1)}(n)}}_{\varphi'(v_{a,j}^{(k+1)}(n))} w_{jq}^{(k+1)}(n-1) e_{a,j}^{(k+1)}(n), \quad (\text{C.31})$$

que é a expressão geral para o erro *a priori* propagado regressivamente.

Essencialmente a diferença entre a propagação do erro do algoritmo de treinamento Acelerador em comparação com os algoritmos BP e RLS-I é que no primeiro se faz propagação do erro *a priori*, enquanto nos demais se faz a propagação do erro *a posteriori*.

Reunindo as equações do cálculo progressivo e regressivo obtidas anteriormente para $\mathbf{M}_i = m_i \mathbf{I}$, $i = 1, 2, 3$, chega-se no algoritmo Acelerador para treinamento de redes do tipo MLP que está apresentado na Tabela 2.10.

C.3 Observações sobre convergência e inicialização

Para garantir a estabilidade do algoritmo Acelerador (sem a não-linearidade $\varphi(\cdot)$) os valores de m_1, m_2 e m_3 devem satisfazer

$$4m_1^2 m_2 m_3 \geq 1. \quad (\text{C.32})$$

Essa condição, como pode ser verificado em [5], resulta da análise de estabilidade, segundo o critério de Liapunov, do algoritmo de tempo contínuo (equações (C.4) e (C.5)). Para o caso do algoritmo Acelerador com a não-linearidade $\varphi(\cdot)$ é necessário também que esta restrição seja satisfeita. Estudos preliminares utilizando linearização indicam que no caso não-linear outras restrições não são necessárias para garantir a convergência. Entretanto, deve-se lembrar que o algoritmo é utilizado no ajuste dos pesos de uma rede em que erros são propagados regressivamente e não somente no ajuste dos pesos de um neurônio. Assim, da mesma forma que para os algoritmos BP e RLS-I, uma escolha criteriosa dos parâmetros e das condições iniciais é muito importante para o bom funcionamento do algoritmo.

Em [5] foi feita uma análise da velocidade de convergência e verificou-se que ela maior quando os parâmetros satisfazem $4m_1^2 m_2 m_3 = 1$ e, dentro de certos limites, é tanto maior quanto maiores forem os valores de m_1 e μ , sendo μ o passo de adaptação. Este comportamento se repete quando se trata do algoritmo para treinamento de redes neurais do tipo MLP. Experimentalmente observou-se que bons resultados são alcançados se os pesos e os níveis de ajuste (*bias*) forem inicializados com números aleatórios de pequenos valores (da ordem de 10^{-3} por exemplo) uniformemente distribuídos. Cabe observar que na formulação utilizada os níveis de ajuste (*bias*) estão incluídos no vetor de pesos. Os vetores $\mathbf{q}_k^{(l)}$ com $k = 1, 2, \dots, N_l$ e $l = 1, 2, \dots, L$ do instante anterior ao inicial deve ser inicializado com vetor nulo: $\mathbf{q}_k^{(l)}(-1) = \mathbf{0}$.

Apêndice D

Algoritmo Acelerador Complexo

D.1 O algoritmo Acelerador complexo de tempo contínuo modificado

O algoritmo Acelerador de tempo contínuo modificado, em sua versão complexa, é descrito pelas seguintes equações:

$$\dot{\mathbf{w}}(t) = \mathbf{q}(t) \quad (\text{D.1})$$

$$\begin{aligned} \dot{\mathbf{q}}(t) = & -\mathbf{M}_1 \mathbf{u}(t) e^*(t) + \\ & -2\mathbf{M}_1(\mathbf{M}_2 + \mathbf{u}(t)\mathbf{u}^H(t)\mathbf{M}_1\mathbf{M}_3)\mathbf{q}(t) \end{aligned} \quad (\text{D.2})$$

$$e(t) = \phi(\mathbf{u}^T(t)\mathbf{w}^*(t)) - d(t), \quad (\text{D.3})$$

sendo que

- $\phi(\cdot)$ é uma função não-linear tal que $\phi(v) = \varphi(v_R) + j\varphi(v_I)$ e $v = v_R + jv_I$ um sinal complexo. Esta função foi feita igual à do algoritmo *Backpropagation* complexo descrito em [10];
- $\mathbf{u}$ é um vetor complexo tal que $\mathbf{u} = \mathbf{u}_R + j\mathbf{u}_I$, sendo $\mathbf{u}_R$ e $\mathbf{u}_I$ as partes real e imaginária do vetor $\mathbf{u}$ respectivamente;
- $\varphi(\cdot)$ é uma função não linear do tipo sigmoidal;

- $\mathbf{u}^H$ é o hermitiano do vetor $\mathbf{u}$;
- $\mathbf{u}^T$ é o transposto do vetor $\mathbf{u}$;
- $\mathbf{u}^*$ é o complexo conjugado do vetor $\mathbf{u}$.

Como na dedução do algoritmo para sinais reais, a modificação feita no algoritmo Acelerador complexo consistiu apenas na introdução de uma função não-linear ($\phi(\cdot)$) adaptada para sinais complexos.

D.2 O algoritmo Acelerador complexo para treinamento de redes MLP

D.2.1 Cálculo progressivo

O cálculo progressivo, como no caso real, foi feito considerando um neurônio na camada de saída, tendo como entrada o vetor $\mathbf{u}$. Por conveniência de notação, os índices dos neurônios e das camadas foram omitidos.

Utilizando-se o método de Euler reverso para discretizar as equações (D.1) à (D.3), chega-se em

$$\mathbf{w}(n) = \mathbf{w}(n-1) + \mu \mathbf{q}(n) \quad (\text{D.4})$$

$$\begin{aligned} \mathbf{q}(n) = & \mathbf{q}(n-1) - \mu \mathbf{M}_1 \{ \mathbf{u}(n) e^*(n) + \\ & + 2(\mathbf{M}_2 + \mathbf{u}(n) \mathbf{u}^H(n) \mathbf{M}_1 \mathbf{M}_3) \mathbf{q}(n) \} \end{aligned} \quad (\text{D.5})$$

$$e(n) = \phi(\mathbf{u}^T(n) \mathbf{w}^*(n)) - d(n). \quad (\text{D.6})$$

Substituindo a Equação (D.4) na (D.6), obtém-se

$$e(n) = \phi(\mathbf{u}^T(n) \mathbf{w}^*(n-1) + \mu \mathbf{u}^T(n) \mathbf{q}^*(n)) - d(n). \quad (\text{D.7})$$

Considerando $v_0 = \mathbf{u}^T(n) \mathbf{w}^*(n-1)$, pode-se escrever

$$\phi(v_0 + \mu \mathbf{u}^T(n) \mathbf{q}^*(n)) = \varphi(v_{0R} + (\mu \mathbf{u}^T(n) \mathbf{q}^*(n))_R) + j\varphi(v_{0I} + (\mu \mathbf{u}^T(n) \mathbf{q}^*(n))_I). \quad (\text{D.8})$$

Fazendo aproximação por série de Taylor de 1ª ordem em torno de v_0 , obtém-se

$$\begin{aligned}\phi(v_0 + \mu \mathbf{u}^T(n) \mathbf{q}^*(n)) &\approx \varphi(v_{0R}) + \varphi'(v_{0R}) \{\mu \mathbf{u}^T(n) \mathbf{q}^*(n)\}_R + \\ &+ j\varphi(v_{0I}) + j\varphi'(v_{0I}) \{\mu \mathbf{u}^T(n) \mathbf{q}^*(n)\}_I.\end{aligned}\quad (\text{D.9})$$

Chamando $v_0 = v_a(n) = \mathbf{u}^T(n) \mathbf{w}^*(n-1)$ e substituindo a aproximação anterior na Equação (D.7), chega-se em

$$\begin{aligned}e(n) &= \varphi(v_{aR}(n)) + \mu \varphi'(v_{aR}(n)) \{\mathbf{u}^T(n) \mathbf{q}^*(n)\}_R + \\ &+ j\varphi(v_{aI}(n)) + j\mu \varphi'(v_{aI}(n)) \{\mathbf{u}^T(n) \mathbf{q}^*(n)\}_I - d(n).\end{aligned}\quad (\text{D.10})$$

Definindo o erro *a priori* $e_a(n) = \phi(v_a(n)) - d(n)$, a equação anterior pode ser rescrita como

$$e(n) = e_a(n) + \mu \varphi'(v_{aR}(n)) \{\mathbf{u}^T(n) \mathbf{q}^*(n)\}_R + j\mu \varphi'(v_{aI}(n)) \{\mathbf{u}^T(n) \mathbf{q}^*(n)\}_I. \quad (\text{D.11})$$

Substituindo esta equação em (D.5), considerando as partes real e imaginária separadamente e igualando a partes reais de ambos os lados da equação resultante, obtém-se

$$\begin{aligned}&\{I + \mu^2 \varphi'(v_{aR}(n)) \mathbf{M}_1 \mathbf{u}_R(n) \mathbf{u}_R^T(n) + \mu^2 \varphi'(v_{aI}(n)) \mathbf{M}_1 \mathbf{u}_I(n) \mathbf{u}_I^T(n) + \\ &+ 2\mu \mathbf{M}_1 (\mathbf{M}_2 + (\mathbf{u}_R(n) \mathbf{u}_R^T(n) + \mathbf{u}_I(n) \mathbf{u}_I^T(n)) \mathbf{M}_1 \mathbf{M}_3) \mathbf{q}_R(n) + \\ &+ \{\mu^2 \varphi'(v_{aR}(n)) \mathbf{M}_1 \mathbf{u}_R(n) \mathbf{u}_I^T(n) - \mu^2 \varphi'(v_{aI}(n)) \mathbf{M}_1 \mathbf{u}_I(n) \mathbf{u}_R^T(n) + \\ &+ 2\mu \mathbf{M}_1 (\mathbf{u}_R(n) \mathbf{u}_I^T(n) - \mathbf{u}_I(n) \mathbf{u}_R^T(n)) \mathbf{M}_1 \mathbf{M}_3\} \mathbf{q}_I(n) = \\ &= \mathbf{q}_R(n-1) - \mu \mathbf{M}_1 (\mathbf{u}_R(n) e_{aR}(n) + \mathbf{u}_I(n) e_{aI}(n)).\end{aligned}\quad (\text{D.12})$$

Igualando agora as partes imaginárias, resulta em

$$\begin{aligned}&\{\mu^2 \varphi'(v_{aR}(n)) \mathbf{M}_1 \mathbf{u}_I(n) \mathbf{u}_R^T(n) - \mu^2 \varphi'(v_{aI}(n)) \mathbf{M}_1 \mathbf{u}_R(n) \mathbf{u}_I^T(n) + \\ &+ 2\mu \mathbf{M}_1 (\mathbf{u}_I(n) \mathbf{u}_R^T(n) - \mathbf{u}_R(n) \mathbf{u}_I^T(n)) \mathbf{M}_1 \mathbf{M}_3\} \mathbf{q}_R(n) + \\ &+ \{I + \mu^2 \varphi'(v_{aR}(n)) \mathbf{M}_1 \mathbf{u}_I(n) \mathbf{u}_I^T(n) + \mu^2 \varphi'(v_{aI}(n)) \mathbf{M}_1 \mathbf{u}_R(n) \mathbf{u}_R^T(n) + \\ &+ 2\mu \mathbf{M}_1 (\mathbf{M}_2 + (\mathbf{u}_R(n) \mathbf{u}_R^T(n) + \mathbf{u}_I(n) \mathbf{u}_I^T(n)) \mathbf{M}_1 \mathbf{M}_3) \mathbf{q}_I(n) = \\ &= \mathbf{q}_I(n-1) - \mu \mathbf{M}_1 (\mathbf{u}_I(n) e_{aR}(n) + \mathbf{u}_R(n) e_{aI}(n)).\end{aligned}\quad (\text{D.13})$$

Rescrevendo as equações (D.12) e (D.13) na forma matricial, obtém-se

$$\begin{pmatrix} \Gamma_{RR}(n) & \Gamma_{RI}(n) \\ \Gamma_{IR}(n) & \Gamma_{II}(n) \end{pmatrix} \begin{pmatrix} \mathbf{q}_R(n) \\ \mathbf{q}_I(n) \end{pmatrix} = \begin{pmatrix} \mathbf{q}_R(n-1) - \mu \mathbf{M}_1(\mathbf{u}_R(n)e_{a_R}(n) + \mathbf{u}_I(n)e_{a_I}(n)) \\ \mathbf{q}_I(n-1) - \mu \mathbf{M}_1(\mathbf{u}_I(n)e_{a_R}(n) + \mathbf{u}_R(n)e_{a_I}(n)) \end{pmatrix}, \quad (\text{D.14})$$

sendo:

$$\begin{aligned} \Gamma_{RR}(n) = & \{I + \mu^2 \varphi'(v_{a_R}(n)) \mathbf{M}_1 \mathbf{u}_R(n) \mathbf{u}_R^T(n) + \mu^2 \varphi'(v_{a_I}(n)) \mathbf{M}_1 \mathbf{u}_I(n) \mathbf{u}_I^T(n) + \\ & + 2\mu \mathbf{M}_1(\mathbf{M}_2 + (\mathbf{u}_R(n) \mathbf{u}_R^T(n) + \mathbf{u}_I(n) \mathbf{u}_I^T(n)) \mathbf{M}_1 \mathbf{M}_3\}, \end{aligned} \quad (\text{D.15})$$

$$\begin{aligned} \Gamma_{RI}(n) = & \{\mu^2 \varphi'(v_{a_R}(n)) \mathbf{M}_1 \mathbf{u}_R(n) \mathbf{u}_I^T(n) - \mu^2 \varphi'(v_{a_I}(n)) \mathbf{M}_1 \mathbf{u}_I(n) \mathbf{u}_R^T(n) + \\ & + 2\mu \mathbf{M}_1(\mathbf{u}_R(n) \mathbf{u}_I^T(n) - \mathbf{u}_I(n) \mathbf{u}_R^T(n)) \mathbf{M}_1 \mathbf{M}_3\}, \end{aligned} \quad (\text{D.16})$$

$$\begin{aligned} \Gamma_{IR}(n) = & \{\mu^2 \varphi'(v_{a_R}(n)) \mathbf{M}_1 \mathbf{u}_I(n) \mathbf{u}_R^T(n) - \mu^2 \varphi'(v_{a_I}(n)) \mathbf{M}_1 \mathbf{u}_R(n) \mathbf{u}_I^T(n) \\ & + 2\mu \mathbf{M}_1(\mathbf{u}_I(n) \mathbf{u}_R^T(n) - \mathbf{u}_R(n) \mathbf{u}_I^T(n)) \mathbf{M}_1 \mathbf{M}_3\}, \end{aligned} \quad (\text{D.17})$$

$$\begin{aligned} \Gamma_{II}(n) = & \{I + \mu^2 \varphi'(v_{a_R}(n)) \mathbf{M}_1 \mathbf{u}_I(n) \mathbf{u}_I^T(n) + \mu^2 \varphi'(v_{a_I}(n)) \mathbf{M}_1 \mathbf{u}_R(n) \mathbf{u}_R^T(n) + \\ & + 2\mu \mathbf{M}_1(\mathbf{M}_2 + (\mathbf{u}_R(n) \mathbf{u}_R^T(n) + \mathbf{u}_I(n) \mathbf{u}_I^T(n)) \mathbf{M}_1 \mathbf{M}_3\}. \end{aligned} \quad (\text{D.18})$$

Observando a Equação (D.14), conclui-se que para se obter $\mathbf{q}(n) = \mathbf{q}_R(n) + j\mathbf{q}_I(n)$, é necessário calcular a inversa da matriz $\Gamma_c(n) = \begin{pmatrix} \Gamma_{RR}(n) & \Gamma_{RI}(n) \\ \Gamma_{IR}(n) & \Gamma_{II}(n) \end{pmatrix}$. Para isso, pode-se aplicar o lema de inversão de matrizes apresentado no Apêndice A. Antes, porém, foram definidas as seguintes variáveis

$$\alpha(n) = \mu \varphi'(v_{a_R}(n)), \quad b(n) = \mu^2 \varphi'(v_{a_I}(n)),$$

$$\mathbf{M}_a = \begin{pmatrix} \mathbf{M}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{M}_1 \end{pmatrix}, \quad \mathbf{M}_b = \begin{pmatrix} \mathbf{M}_2 & \mathbf{0} \\ \mathbf{0} & \mathbf{M}_2 \end{pmatrix}, \quad \mathbf{M}_c = \begin{pmatrix} \mathbf{M}_3 & \mathbf{0} \\ \mathbf{0} & \mathbf{M}_3 \end{pmatrix},$$

$$\mathbf{I}_c = \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{pmatrix}, \quad \mathbf{E} = \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & -\mathbf{I} \end{pmatrix},$$

$$\mathbf{x}_a(n) = \begin{pmatrix} \mathbf{u}_R(n) \\ \mathbf{u}_I(n) \end{pmatrix}, \quad \mathbf{x}_b(n) = \begin{pmatrix} \mathbf{u}_I(n) \\ \mathbf{u}_R(n) \end{pmatrix},$$

sendo

- $\mathbf{q}(n)$ um vetor de dimensão N_0 ;
- $\mathbf{M}_1$, $\mathbf{M}_2$ e $\mathbf{M}_3$ matrizes positivas definidas de dimensão $N_0 \times N_0$;
- $\mathbf{I}$ a matriz identidade de dimensão $N_0 \times N_0$;
- $\mathbf{0}$ a matriz nula de dimensão $N_0 \times N_0$.

Partindo dessas definições pode-se rescrever a matriz $\Gamma_c(n)$ da seguinte forma

$$\begin{aligned} \Gamma_c(n) = & \mathbf{I}_c + 2\mu\mathbf{M}_a\mathbf{M}_b + b(n)\mathbf{M}_a\mathbf{x}_b(n)\mathbf{x}_b^T(n) + 2\mu\mathbf{M}_a\mathbf{E}\mathbf{x}_b(n)\mathbf{x}_b^T(n)\mathbf{E}\mathbf{M}_a\mathbf{M}_c + \\ & + \mu\alpha(n)\mathbf{M}_a\mathbf{x}_a(n)\mathbf{x}_a^T(n) + 2\mu\mathbf{M}_a\mathbf{x}_a(n)\mathbf{x}_a^T(n)\mathbf{M}_a\mathbf{M}_c. \end{aligned} \quad (\text{D.19})$$

Aplicando o lema de inversão de matrizes (Apêndice A) três vezes consecutivas, chega-se no seguinte resultado para a inversa da matriz $\Gamma_c(n)$

$$\Gamma_c(n) = \mu^{-1}(\alpha(n)\mathbf{I}_c + 2\mathbf{M}_a\mathbf{M}_c)^{-1}\mathbf{B}_1^{-1}(n)\mathbf{M}_a^{-1}, \quad (\text{D.20})$$

sendo

$$\mathbf{B}_1^{-1}(n) = \mathbf{B}_2^{-1}(n) - \frac{1}{1 + \mathbf{x}_a^T(n)\mathbf{B}_2^{-1}(n)\mathbf{x}_a(n)}\mathbf{B}_2^{-1}(n)\mathbf{x}_a(n)\mathbf{x}_a^T(n)\mathbf{B}_2^{-1}(n), \quad (\text{D.21})$$

$$\mathbf{B}_2^{-1}(n) = \mu b^{-1}(n)(\alpha(n)\mathbf{I}_c + 2\mathbf{M}_a\mathbf{M}_c)\mathbf{G}_1^{-1}(n), \quad (\text{D.22})$$

$$\mathbf{G}_1^{-1}(n) = \mathbf{G}_2^{-1}(n) - \frac{1}{1 + \mathbf{x}_b^T(n)\mathbf{G}_2^{-1}(n)\mathbf{x}_b(n)}\mathbf{G}_2^{-1}(n)\mathbf{x}_b(n)\mathbf{x}_b^T(n)\mathbf{G}_2^{-1}(n), \quad (\text{D.23})$$

$$\mathbf{G}_2^{-1}(n) = b(n)(2\mu)^{-1}(\mathbf{M}_a\mathbf{M}_c)^{-1}\mathbf{H}_1^{-1}(n) \quad (\text{D.24})$$

$$\mathbf{H}_1^{-1}(n) = \mathbf{H}_2^{-1} - \frac{1}{1 + \mathbf{x}_b^T(n)\mathbf{E}\mathbf{H}_2^{-1}\mathbf{E}\mathbf{x}_b(n)}\mathbf{H}_2^{-1}\mathbf{E}\mathbf{x}_b(n)\mathbf{x}_b^T(n)\mathbf{E}\mathbf{H}_2^{-1}, \quad (\text{D.25})$$

$$\mathbf{H}_2^{-1} = \mathbf{M}_a\mathbf{M}_c((2\mu)^{-1}\mathbf{M}_a^{-1} + \mathbf{M}_b^{-1})^{-1}. \quad (\text{D.26})$$

Como na dedução para o caso real, o cálculo da matriz $\Gamma_c^{-1}(n)$ pode ser simplificado se as matrizes $\mathbf{M}_a$, $\mathbf{M}_b$ e $\mathbf{M}_c$ forem diagonais. Fazendo $\mathbf{M}_a = m_1\mathbf{I}_c$, $\mathbf{M}_b = m_2\mathbf{I}_c$, $\mathbf{M}_c = m_3\mathbf{I}_c$, obtém-se a seguinte equação para $\Gamma_c^{-1}(n)$

$$\Gamma_c^{-1}(n) = \frac{1}{\mu m_1(\alpha(n) + 2m_1m_3)}\mathbf{B}_1^{-1}(n), \quad (\text{D.27})$$

sendo

$$\mathbf{B}_1^{-1}(n) = \mathbf{B}_2^{-1}(n) - \frac{1}{1 + \mathbf{x}_a^T(n)\mathbf{B}_2^{-1}(n)\mathbf{x}_a(n)}\mathbf{B}_2^{-1}(n)\mathbf{x}_a(n)\mathbf{x}_a^T(n)\mathbf{B}_2^{-1}(n), \quad (\text{D.28})$$

$$\mathbf{B}_2^{-1}(n) = \mu(\alpha(n) + 2m_1m_3)\mathbf{G}_1^{-1}(n), \quad (\text{D.29})$$

$$\mathbf{G}_1^{-1}(n) = \mathbf{G}_2^{-1}(n) - \frac{b(n)}{1 + b(n)\mathbf{x}_b^T(n)\mathbf{G}_2^{-1}(n)\mathbf{x}_b(n)}\mathbf{G}_2^{-1}(n)\mathbf{x}_b(n)\mathbf{x}_b^T(n)\mathbf{G}_2^{-1}(n), \quad (\text{D.30})$$

$$\mathbf{G}_2^{-1}(n) = (2\mu m_1m_3)^{-1}\mathbf{H}_1^{-1}(n), \quad (\text{D.31})$$

$$\mathbf{H}_1^{-1}(n) = \mathbf{H}_2^{-1} - \frac{1}{1 + \mathbf{x}_b^T(n)\mathbf{E}\mathbf{H}_2^{-1}\mathbf{E}\mathbf{x}_b(n)}\mathbf{H}_2^{-1}\mathbf{E}\mathbf{x}_b(n)\mathbf{x}_b^T(n)\mathbf{E}\mathbf{H}_2^{-1}, \quad (\text{D.32})$$

$$\mathbf{H}_2^{-1} = \frac{2\mu m_1^2m_3}{1 + 2\mu m_1m_2}\mathbf{I}_c. \quad (\text{D.33})$$

Para obter as equações anteriores, além de considerar as matrizes $\mathbf{M}_a$, $\mathbf{M}_b$ e $\mathbf{M}_c$ diagonais, foi feita uma modificação no cálculo de $\mathbf{B}_2^{-1}(n)$, evitando a divisão por $b(n)$, que pode ser zero quando a derivada da função não-linear ($\varphi'(\cdot)$) for nula. Com essas últimas considerações, além da constatação de que $\mathbf{p}^T\mathbf{v}$ é um escalar, sendo $\mathbf{p}$ e $\mathbf{v}$ vetores coluna de mesma dimensão, consegue-se obter um algoritmo não-matricial mais simples em que o cálculo da matriz $\Gamma_c^{-1}(n)$ se reduz ao cálculo de escalares a cada instante n . Mais detalhes do algoritmo Acelerador complexo não-matricial podem ser vistos na seção D.3.

Com a matriz $\Gamma_c^{-1}(n)$ (ou sua forma não matricial), calcula-se o vetor $\mathbf{q}(n) = \mathbf{q}_R(n) + j\mathbf{q}_I(n)$ pela Equação (D.14). Com $\mathbf{q}(n)$ faz-se a atualização dos pesos pela Equação (D.4). Como no algoritmo para sinais reais, os vetores $\mathbf{q}$ do instante anterior ao inicial devem ser inicializados com o vetor nulo. Os pesos devem ser inicializados com valores complexos aleatórios distribuídos uniformemente num intervalo de pequenos valores.

Deve-se notar que nesta dedução não se utilizou o índice das camadas e nem dos neurônios. Como no caso real, para a dedução do cálculo progressivo do algoritmo Acelerador complexo foi considerado apenas um neurônio pertencente à camada de saída da rede, sendo $\mathbf{u}$ o vetor de entrada desse neurônio. As equações aqui obtidas, podem ser estendidas facilmente para um neurônio pertencente à qualquer camada l

da rede, como pode ser observado comparando tais equações com aquelas expostas na seção 2.3.4.

D.2.2 Cálculo regressivo

Como no caso real, para obter as equações do cálculo regressivo do algoritmo Acelerador complexo, deve-se considerar toda a estrutura da rede. Considerou-se então, uma rede com L camadas, sendo N_l , o número de neurônios da camada l , $w_{km}^{(l)}$ o peso que liga o neurônio k da camada l ao neurônio m da camada $l - 1$, $y_k^{(l)}(n) = \phi(v_{a,k}^{(l)}(n))$, com $v_{a,k}^{(l)}(n) = \mathbf{y}^{(l-1)T}(n)\mathbf{w}_k^{(l)*}(n-1)$, a saída do neurônio k da camada l e por conseguinte a k -ésima entrada da camada $l + 1$.

Com o objetivo de se propagar regressivamente o erro da camada de saída, introduz-se a seguinte medida de desempenho

$$\xi(n) = \sum_{k=1}^{N_L} |e_{a,k}^{(L)}(n)|^2, \quad (\text{D.34})$$

sendo $e_{a,k}^{(L)} = y_k^{(L)}(n) - d(n)$, o erro *a priori* da camada de saída. Introduzindo o gradiente de $e_{a,k}^{(l)2}$ com relação a $y_k^{(l)}$, obtém-se

$$e_{a,k}^{(l)} = -\frac{1}{2} \nabla y_k^{(l)}(e_{a,k}^2) = -\frac{1}{2} \frac{\partial e_{a,k}^{(l)2}}{\partial y_{kR}^{(l)}} - \frac{j}{2} \frac{\partial e_{a,k}^{(l)2}}{\partial y_{kI}^{(l)}}. \quad (\text{D.35})$$

Similarmente, pode-se definir

$$\begin{aligned} \Delta_k^{(l)} &= -\frac{1}{2} \nabla v_{a,k}^{(l)}(e_{a,k}^2) = \delta_{kR}^{(l)} + j\delta_{kI}^{(l)} = -\frac{1}{2} \frac{\partial e_{a,k}^{(l)2}}{\partial v_{a,kR}^{(l)}} - \frac{j}{2} \frac{\partial e_{a,k}^{(l)2}}{\partial v_{a,kI}^{(l)}} = \\ &= -\frac{1}{2} \frac{\partial e_{a,k}^{(l)2}}{\partial y_{kR}^{(l)}} \frac{\partial y_{kR}^{(l)}}{\partial v_{a,kR}^{(l)}} - \frac{j}{2} \frac{\partial e_{a,k}^{(l)2}}{\partial y_{kI}^{(l)}} \frac{\partial y_{kI}^{(l)}}{\partial v_{a,kI}^{(l)}} = e_{a,kR}^{(l)} \varphi'(v_{a,kR}^{(l)}) + j e_{a,kI}^{(l)} \varphi'(v_{a,kI}^{(l)}). \end{aligned} \quad (\text{D.36})$$

O próximo passo é achar uma propagação recursiva para $e_{a,k}^{(l)}$.

Na última camada ($l = L$), obtém-se

$$e_{a,k}^{(L)}(n) = (y_{kR}^{(L)}(n) - d_{kR}(n)) + j(y_{kI}^{(L)}(n) - d_{kI}(n)). \quad (\text{D.37})$$

Para uma camada oculta ($0 < l < L$), pode-se expandir a expressão de $e_{a,k}^{(l)}$ usando a regra da cadeia:

$$e_{a,k}^{(l)} = -\frac{1}{2} \sum_{q=1}^{N_{l+1}} \left\{ \left(\frac{\partial e_{a,q}^2}{\partial v_{a,qR}^{(l+1)}} \frac{\partial v_{a,qR}^{(l+1)}}{\partial y_{kR}^{(l)}} + \frac{\partial e_{a,q}^2}{\partial v_{a,qI}^{(l+1)}} \frac{\partial v_{a,qI}^{(l+1)}}{\partial y_{kR}^{(l)}} \right) + \right. \\ \left. + j \left(\frac{\partial e_{a,q}^2}{\partial v_{a,qR}^{(l+1)}} \frac{\partial v_{a,qR}^{(l+1)}}{\partial y_{kI}^{(l)}} + \frac{\partial e_{a,q}^2}{\partial v_{a,qI}^{(l+1)}} \frac{\partial v_{a,qI}^{(l+1)}}{\partial y_{kI}^{(l)}} \right) \right\}, \quad (D.38)$$

sendo

$$v_{a,kR}^{(l)}(n) = \sum_{m=0}^{N_{l-1}} (w_{kmR}^{(l)}(n-1) y_{mR}^{(l-1)}(n) + w_{kmI}^{(l)}(n-1) y_{mI}^{(l-1)}(n)) \quad (D.39)$$

e

$$v_{a,kI}^{(l)}(n) = \sum_{m=0}^{N_{l-1}} (w_{kmR}^{(l)}(n-1) y_{mI}^{(l-1)}(n) + w_{kmI}^{(l)}(n-1) y_{mR}^{(l-1)}(n)). \quad (D.40)$$

Substituindo as as equações (D.39) e (D.40) em (D.38) e calculando-se as derivadas, obtém-se

$$e_{a,k}^{(l)}(n) = \sum_{q=1}^{N_{l+1}} \{ (\delta_{qR}^{(l+1)}(n) w_{qkR}^{(l+1)}(n-1) - \delta_{qI}^{(l+1)}(n) w_{qkI}^{(l+1)}(n-1)) + \\ + j (\delta_{qR}^{(l+1)}(n) w_{qkI}^{(l+1)}(n-1) + \delta_{qI}^{(l+1)}(n) w_{qkR}^{(l+1)}(n-1)) \}. \quad (D.41)$$

Finalmente, substituindo a Equação (D.36) na equação anterior, obtém-se

$$e_{a,k}^{(l)}(n) = \sum_{q=1}^{N_{l+1}} w_{qn}^{(l+1)}(n-1) \Delta_q^{(l+1)}(n), \quad \text{para } l = L-1, \dots, 1. \quad (D.42)$$

A propagação regressiva do erro no algoritmo Acelerador em sua versão complexa é semelhante à do algoritmo *Backpropagation* complexo. A diferença está apenas na definição das variáveis em que se toma o conjugado ou hermitiano e também na propagação do erro *a priori*.

D.3 O algoritmo Acelerador complexo não-matricial

Como foi dito anteriormente considerando as matrizes $\mathbf{M}_a$, $\mathbf{M}_b$ e $\mathbf{M}_c$ diagonais e constatando-se que $\mathbf{p}^T \mathbf{v}$ é um escalar, sendo $\mathbf{p}$ e $\mathbf{v}$ vetores coluna de mesma dimensão,

consegue-se obter um algoritmo não-matricial mais simples em que o cálculo da matriz $\Gamma_c^{-1}(n)$ se reduz ao cálculo de escalares a cada instante n .

Utilizando as definições de $\mathbf{x}_a(n)$, $\mathbf{x}_b(n)$ e $\mathbf{I}$ da seção anterior, a matriz $\Gamma_c^{-1}(n)$ pode ser rescrita como

$$\Gamma_c^{-1}(n) = \begin{pmatrix} \Gamma_{11}(n) & \Gamma_{12}(n) \\ \Gamma_{21}(n) & \Gamma_{22}(n) \end{pmatrix}, \quad (\text{D.43})$$

sendo

$$\Gamma_{11} = a_0 \mathbf{I} + a_1 \mathbf{u}_R \mathbf{u}_R^T + (a_2 + a_3) \mathbf{u}_R \mathbf{u}_I^T + (a_4 + a_7) \mathbf{u}_I \mathbf{u}_R^T + (a_5 + a_6 + a_8 + a_9) \mathbf{u}_I \mathbf{u}_I^T, \quad (\text{D.44})$$

$$\Gamma_{12} = a_1 \mathbf{u}_R \mathbf{u}_I^T + (-a_2 + a_3) \mathbf{u}_R \mathbf{u}_R^T + (a_4 + a_7) \mathbf{u}_I \mathbf{u}_I^T + (-a_5 + a_6 - a_8 + a_9) \mathbf{u}_I \mathbf{u}_R^T, \quad (\text{D.45})$$

$$\Gamma_{21} = a_1 \mathbf{u}_I \mathbf{u}_R^T + (a_2 + a_3) \mathbf{u}_I \mathbf{u}_I^T + (-a_4 + a_7) \mathbf{u}_R \mathbf{u}_R^T + (-a_5 - a_6 + a_8 + a_9) \mathbf{u}_R \mathbf{u}_I^T, \quad (\text{D.46})$$

$$\Gamma_{22} = a_0 \mathbf{I} + a_1 \mathbf{u}_I \mathbf{u}_I^T + (-a_2 + a_3) \mathbf{u}_I \mathbf{u}_R^T + (-a_4 + a_7) \mathbf{u}_R \mathbf{u}_I^T + (a_5 - a_6 - a_8 + a_9) \mathbf{u}_R \mathbf{u}_R^T. \quad (\text{D.47})$$

Nas equações (D.44) a (D.47), as variáveis a_i para $i = 0, \dots, 9$ são escalares que variam com o tempo, os vetores $\mathbf{u}_R$ e $\mathbf{u}_I$ são respectivamente as partes real e imaginária do vetor de entrada do neurônio e também variam a cada instante n . Dessa forma a Equação (D.14) pode ser rescrita como

$$\begin{pmatrix} \Gamma_{11}(n) & \Gamma_{12}(n) \\ \Gamma_{21}(n) & \Gamma_{22}(n) \end{pmatrix} \begin{pmatrix} \mathbf{q}_R(n) \\ \mathbf{q}_I(n) \end{pmatrix} = \begin{pmatrix} \mathbf{q}_R(n-1) - \mu \mathbf{M}_1(\mathbf{u}_R(n)e_{a_R}(n) + \mathbf{u}_I(n)e_{a_I}(n)) \\ \mathbf{q}_I(n-1) - \mu \mathbf{M}_1(\mathbf{u}_I(n)e_{a_R}(n) + \mathbf{u}_R(n)e_{a_I}(n)) \end{pmatrix}, \quad (\text{D.48})$$

Levando-se em conta novamente que o produto $\mathbf{p}^T \mathbf{v}$ é um escalar e observando a equação anterior, conclui-se que é possível obter os vetores $\mathbf{q}_R(n)$ e $\mathbf{q}_I(n)$ sem a necessidade do cálculo da matriz $\Gamma_c^{-1}(n)$ diretamente. Neste caso, é necessário calcular escalares a cada instante de tempo n . A dedução das expressões desses escalares é

longa e tediosa e não será mostrada aqui. A seguir é apresentado um programa em MatLab que calcula o vetor $\mathbf{q}(n) = \mathbf{q}_R(n) + j\mathbf{q}_I(n)$. Neste programa os escalares a_i para $i = 0, \dots, 9$ e d_i , $i = 1, \dots, 7$ dependem dos vetores $\mathbf{u}(n)$ e/ou $\mathbf{q}(n-1)$ e devem ser calculados a cada iteração.

```
function q=calculaq(m1,m2,m3,mu,Xr,Xi,qr11,qi11,ear,eai,dvr,dvi)
```

```
% q=calculaq(m1,m2,m3,mu,Xr,Xi,qr11,qi11,ear,eai,dvr,dvi)
```

```
% Calcula o vetor q, através da forma simplificada do algoritmo
```

```
% acelerador complexo
```

```
% Parâmetros de entrada:
```

```
% m1,m2,m3,mu: parâmetros do algoritmo acelerador
```

```
% Xr:           parte real do vetor de entrada do neurônio
```

```
% Xi:           parte imag do vetor de entrada do neurônio
```

```
% qr11:         parte real do vetor q do instante anterior
```

```
% qi11:         parte imag do vetor q do instante anterior
```

```
% ear:          parte real do erro a priori do neurônio
```

```
% eai:          parte imaginária do erro a priori do neurônio
```

```
% dvr:          parte real da derivada da função não-linear
```

```
%              calculada na saída do neurônio
```

```
% dvi:          parte imag da derivada da função não-linear
```

```
%              calculada na saída do neurônio
```

```
% MTMS, maio de 2000
```

```
M=length(Xr);
```

```

E=diag([ones(1,M) -ones(1,M)]);
ya=[Xr;Xi];
yb=[Xi;Xr];
qr11=qr11';
qi11=qi11';

a=mu*dvr;
b=mu^2*dvi;

kh2=2*mu*m1^2*m3/(1+2*mu*m1*m2);
kg2=1/(2*mu*m1*m3);
kb2=mu*(a+2*m1*m3);
kc=1/(m1*kb2);

bh1=kh2^2/(1+kh2*yb'*yb);

c1=yb'*E*yb;
c2=ya'*E*yb;
c3=yb'*E*ya;
c4=ya'*yb;

bg1=b/(1+b*kg2*kh2*yb'*yb-b*kg2*bh1*c1^2);
bb1=1/(1+kb2*kg2*kh2*ya'*ya-kb2*kg2*bh1*...
(1+kg2*bg1*bh1*c1^2)*c2*c3+kb2*kg2^2*kh2*bg1*...
bh1*c1*c4*(c2+c3)-kb2*kg2^2*kh2^2*bg1*c4^2);

k1=(1+kg2*bg1*bh1*c1^2);

```

```

a0=kc*kb2*kg2*kh2;
a1=-kc*bb1*(kb2*kg2*kh2)^2;
a2=-kc*bb1*(-kb2^2*kg2^2*kh2*bh1*k1*c2+...
    kb2^2*kg2^3*kh2^2*bg1*bh1*c1*c4);
a3=-kc*bb1*(kb2^2*kg2^3*kh2^2*bg1*bh1*...
    c1*c2-kb2^2*kg2^3*kh2^3*bg1*c4);
a4=-kc*bb1*(-kb2^2*kg2^2*kh2*bh1*k1*c3+...
    kb2^2*kg2^3*kh2^2*bg1*bh1*c1*c4);
a5=kc*(-kb2*kg2*bh1*k1-bb1*(kb2^2*kg2^2*...
    bh1^2*k1^2*c3*c2-kb2^2*kg2^3*kh2*bg1*...
    bh1^2*k1*c3*c1*c4-kb2^2*kg2^3*kh2*bg1*bh1^2*...
    c1*c4*k1*c2+(kb2*kg2^2*kh2*bg1*bh1*c1)^2*c4^2));
a6=kc*(kb2*kg2^2*kh2*bg1*bh1*c1-bb1*(-kb2^2*kg2^3*...
    kh2*bg1*bh1^2*k1*c3*c1*c2+kb2^2*kg2^3*kh2^2*bg1*...
    bh1*k1*c3*c4+(kb2*kg2^2*kh2*bg1*bh1*c1)^2*c4*c2-...
    kb2^2*kg2^4*kh2^3*bg1^2*bh1*c1*c4^2));
a7=-kc*bb1*(kb2^2*kg2^3*kh2^2*bg1*bh1*c1*c3-...
    kb2^2*kg2^3*kh2^3*bg1*c4);
a8=kc*(kb2*kg2^2*kh2*bg1*bh1*c1-bb1*(-kb2^2*...
    kg2^3*kh2*bg1*bh1^2*c1*c3*k1*c2+(kb2*kg2^2*...
    kh2*bg1*bh1*c1)^2*c3*c4+kb2^2*kg2^3*kh2^2*bg1*...
    bh1*k1*c4*c2-kb2^2*kg2^4*kh2^3*bg1^2*bh1*c1*c4^2));
a9=kc*(-kb2*kg2^2*kh2^2*bg1-bb1*((kb2*kg2^2*kh2*bg1*...
    bh1*c1)^2*c3*c2-kb2^2*kg2^4*kh2^3*bg1^2*bh1*c1*c3*c4-...
    kb2^2*kg2^4*kh2^3*bg1^2*bh1*c1*c2*c4+(kb2*kg2^2*kh2^2*...
    bg1)^2*c4^2));

d1=Xr'*qr11;

```

```

d2=Xi'*qr11;
d3=Xr'*Xr;
d4=Xi'*Xi;
d5=Xr'*Xi;
d6=Xr'*qi11;
d7=Xi'*qi11;

gra=a0*qr11;
qrb=(a1*(d1+d7)+(a2+a3)*d2+(-a2+a3)*d6-mu*m1*(ear*...
      (a0+a1*(d3+d4)+2*a3*d5)+eai*((a2+a3)*d4-(-a2+a3)*d3)))*Xr;
qrc=((a4+a7)*(d1+d7)+(a5+a6+a8+a9)*d2+(-a5+a6-a8+a9)*d6-...
      mu*m1*(ear*(2*(a6+a9)*d5+(a4+a7)*(d3+d4))+eai*(a0+...
      (a5+a6+a8+a9)*d4-(-a5+a6-a8+a9)*d3)))*Xi;
qr=gra+qrb+qrc;

qia=a0*qi11;
qib=(((-a4+a7)*(d1+d7)+(a5-a6-a8+a9)*d6+(-a5-a6+a8+a9)*d2-...
      mu*m1*(ear*(2*(-a6+a9)*d5+(-a4+a7)*(d3+d4))+eai*...
      (-a0+(-a5-a6+a8+a9)*d4-(a5-a6-a8+a9)*d3)))*Xr;
qic=(a1*(d1+d7)+(a2+a3)*d2+(-a2+a3)*d6-mu*m1*(ear*...
      (a0+a1*(d3+d4)+2*a3*d5)+eai*((a2+a3)*d4-(-a2+a3)*d3)))*Xi;
qi=qia+qib+qic;

j=sqrt(-1);
q=qr+j*qi;
q=q.';

```

Este programa não é eficiente porque a quantidade de operações por iteração no cálculo dos escalares ainda pode ser minimizada.

Apêndice E

Algoritmo RTRL

E.1 O algoritmo RTRL real

A dedução deste algoritmo foi baseada no esquema da rede recorrente da Figura 2.8. Neste caso $d_k(n)$ é a resposta desejada do neurônio k no instante de tempo n e $\mathcal{N}_s$ é o conjunto de neurônios de saída da rede. Assim, o sinal de erro é definido como

$$e_k(n) = \begin{cases} d_k(n) - y_k(n) & j \in \mathcal{N}_s \\ 0 & j \notin \mathcal{N}_s, \end{cases} \quad (\text{E.1})$$

sendo

$$y_k(n+1) = \varphi(v_k(n+1)), \quad (\text{E.2})$$

$$v_k(n+1) = \sum_{m \in \mathcal{N}_e \cup \mathcal{N}_o} w_{km}(n) u_m(n), \quad (\text{E.3})$$

$$u_m(n) = \begin{cases} u(n-m+N+1) & m \in \mathcal{N}_e \\ y_m(n) & m \in \mathcal{N}_o, \end{cases} \quad (\text{E.4})$$

$\mathcal{N}_e$ o conjunto formado pelas unidades de entrada

e

$\mathcal{N}_o$ o conjunto formado pelas demais unidades.

A soma instantânea do erro quadrático no tempo n é definida como

$$\xi(n) = \frac{1}{2} \sum_{k \in \mathcal{N}_s} e_k^2(n). \quad (\text{E.5})$$

O objetivo do algoritmo é minimizar a seguinte função custo

$$\xi_{total} = \sum_n \xi(n). \quad (\text{E.6})$$

Um meio de minimizar esta função é utilizar o método do gradiente determinístico

$$\nabla_{\mathbf{W}} \xi_{total} = \frac{\partial \xi_{total}}{\partial \mathbf{W}} = \sum_n \frac{\partial \xi(n)}{\partial \mathbf{W}} = \sum_n \nabla_{\mathbf{W}} \xi(n), \quad (\text{E.7})$$

sendo $\nabla_{\mathbf{W}}$ é o gradiente de $\xi(n)$ com relação à matriz de pesos $\mathbf{W}$.

Com o objetivo de se obter um algoritmo a fim de treinar a rede recorrente em tempo real, deve-se usar uma estimativa instantânea do gradiente $\nabla_{\mathbf{W}}$, que resulte numa aproximação do método do gradiente determinístico. Para o caso de um peso particular $w_{km}(n)$, podemos definir sua mudança incremental $\Delta w_{km}(n)$, feita no tempo n , como

$$\Delta w_{km}(n) = -\mu \frac{\partial \xi(n)}{\partial w_{km}(n)}, \quad (\text{E.8})$$

sendo μ a taxa de aprendizagem.

Utilizando as equações (E.1) e (E.5), obtém-se

$$\frac{\partial \xi(n)}{\partial w_{km}(n)} = \sum_{p \in \mathcal{N}_s} e_p(n) \frac{\partial e_p(n)}{\partial w_{km}(n)} = - \sum_{p \in \mathcal{N}_s} e_p(n) \frac{\partial y_p(n)}{\partial w_{km}(n)}. \quad (\text{E.9})$$

Utilizando as equações (E.2), (E.3) e a regra da cadeia obtém-se

$$\frac{\partial y_p(n+1)}{\partial w_{km}(n)} = \frac{\partial y_p(n+1)}{\partial v_p(n+1)} \frac{\partial v_p(n+1)}{\partial w_{km}(n)} = \varphi'(v_p(n+1)) \frac{\partial v_p(n+1)}{\partial w_{km}(n)} \quad (\text{E.10})$$

e

$$\frac{\partial v_p(n+1)}{\partial w_{km}(n)} = \sum_{q \in \mathcal{N}_e \cup \mathcal{N}_o} \frac{\partial (w_{pq}(n) u_q(n))}{\partial w_{km}(n)}. \quad (\text{E.11})$$

Deve-se notar que

$$\frac{\partial w_{pq}(n)}{\partial w_{km}(n)} = \begin{cases} 1 & \text{para } p = k, q = m \\ 0 & \text{caso contrário.} \end{cases} \quad (\text{E.12})$$

Utilizando este resultado, pode-se rescrever a Equação (E.11) como

$$\frac{\partial v_p(n+1)}{\partial w_{km}(n)} = \sum_{q \in \mathcal{N}_e \cup \mathcal{N}_o} w_{pq}(n) \frac{\partial u_q(n)}{\partial w_{km}(n)} + \delta_{kp} u_m(n), \quad (\text{E.13})$$

sendo δ_{kp} o delta de kronecker, igual a 1 quando $k = p$ e 0 caso contrário. Além disso, é importante notar também que

$$\frac{\partial u_q(n)}{\partial w_{km}(n)} = \begin{cases} 0, & \text{para } q \in \mathcal{N}_e \\ \frac{\partial y_q(n)}{\partial w_{km}(n)}, & \text{para } q \in \mathcal{N}_o. \end{cases} \quad (\text{E.14})$$

Combinando as equações (E.10), (E.13) e (E.14) obtém-se

$$\frac{\partial y_p(n+1)}{\partial w_{km}(n)} = \varphi'(v_p(n+1)) \left[\sum_{q \in \mathcal{N}_o} w_{pq}(n) \frac{\partial y_q(n)}{\partial w_{km}(n)} + \delta_{kp} u_m(n) \right]. \quad (\text{E.15})$$

Assumindo que o estado inicial da rede no tempo $n = 0$ não depende dos pesos, pode-se escrever

$$\frac{\partial y_q(0)}{\partial w_{km}(0)} = 0. \quad (\text{E.16})$$

As equações (E.15) e (E.16) valem para todo $m \in \mathcal{N}_o$ e $l \in \mathcal{N}_e \cup \mathcal{N}_o$.

Definindo

$$\Pi_{km}^p(n) = \frac{\partial y_p(n)}{\partial w_{km}(n)} \quad \text{para todo } p \in \mathcal{N}_o, k \in \mathcal{N}_o, m \in \mathcal{N}_e \cup \mathcal{N}_o, \quad (\text{E.17})$$

podemos rescrever a Equação (E.15) como

$$\Pi_{km}^p(n+1) = \varphi'(v_p(n+1)) \left[\sum_{q \in \mathcal{N}_o} w_{pq}(n) \Pi_{km}^q(n) + \delta_{kp} u_m(n) \right] \quad (\text{E.18})$$

com condições iniciais $\Pi_{km}^q(0) = 0$. Esta é a equação que governa a dinâmica do sistema. O termo Π_{km}^p é conhecido como a sensibilidade de y_p em relação à w_{km} .

Utilizando a definição da Equação (E.17) e a Equação (E.9), pode-se rescrever a Equação (E.8) como

$$\Delta w_{km}(n) = \mu \sum_{p \in \mathcal{N}_s} e_p(n) \Pi_{km}^p(n). \quad (\text{E.19})$$

Finalmente, os pesos devem ser atualizados segundo a equação

$$w_{km}(n+1) = w_{km}(n) + \Delta w_{km}(n). \quad (\text{E.20})$$

E.2 O algoritmo CRTRL

No caso complexo, ainda valem as equações (E.1) a (E.4). Deve-se lembrar no entanto que as variáveis são complexas e que a função de ativação considerada deve ser $\phi(\cdot)$, dada pela Equação (2.54).

Como no caso real, a dedução do algoritmo CRTRL se baseia na minimização de uma função custo definida por

$$\xi(n) = \frac{1}{2} \sum_{k \in \mathcal{N}_s} |e_k(n)|^2, \quad (\text{E.21})$$

sendo $e_k(n)$ o erro complexo da unidade k no instante n

$$e_k(n) = d_k(n) - y_k(n) = e_{kR}(n) + j e_{kI}(n). \quad (\text{E.22})$$

Introduz-se inicialmente os termos de sensibilidade P_{RR} , P_{RI} , P_{IR} , P_{II} definidos por

$$\begin{bmatrix} P_{RRpq}^k & P_{RIpq}^k \\ P_{IRpq}^k & P_{IIpq}^k \end{bmatrix}_n = \begin{bmatrix} \frac{\partial y_{kR}}{\partial w_{pqR}} & \frac{\partial y_{kR}}{\partial w_{pqI}} \\ \frac{\partial y_{kI}}{\partial w_{pqR}} & \frac{\partial y_{kI}}{\partial w_{pqI}} \end{bmatrix}_n, \quad (\text{E.23})$$

sendo que o subscrito n indica a dependência das variáveis com o índice de tempo. Sendo $\xi(n)$ uma função não-constante de valor real, sua derivada complexa não é definida e seu gradiente com relação a w_{pq} pode ser calculado como

$$\nabla_{w_{pq}} \xi(n) = \frac{\partial \xi(n)}{\partial w_{pqR}} + j \frac{\partial \xi(n)}{\partial w_{pqI}}. \quad (\text{E.24})$$

Derivando a Equação (E.21) em relação a w_{pqR} e w_{pqI} , obtém-se

$$\frac{\partial \xi(n)}{\partial w_{pqR}} = - \sum_{k=1}^N e_{kR}(n) \frac{\partial y_{kR}(n)}{\partial w_{pqR}} - \sum_{k=1}^N e_{kI}(n) \frac{\partial y_{kI}(n)}{\partial w_{pqR}} \quad (\text{E.25})$$

e

$$\frac{\partial \xi(n)}{\partial w_{pqI}} = - \sum_{k=1}^N e_{kR}(n) \frac{\partial y_{kR}(n)}{\partial w_{pqI}} - \sum_{k=1}^N e_{kI}(n) \frac{\partial y_{kI}(n)}{\partial w_{pqI}}. \quad (\text{E.26})$$

Devemos agora encontrar uma recursão para o cálculo dos termos P_{RR} , P_{RI} , P_{IR} e P_{II} . Considerando a parcela real da Equação (E.2) e derivando em relação a $w_{pqR}(n)$,

obtem-se

$$\frac{\partial y_{k_R}(n+1)}{\partial w_{pqR}(n)} = \varphi'(v_{k_R}(n+1)) \frac{\partial v_{k_R}(n+1)}{\partial w_{pqR}(n)}. \quad (\text{E.27})$$

Substituindo a Equação (E.4) na Equação (E.3) e expressando o resultado em termos das partes real e imaginária dos pesos w_{km} , e dos sinais y_m e u_m , obtém-se

$$\begin{aligned} v_{k_R}(n+1) = & \sum_{l=1}^N (w_{km_R} y_{m_R}(n) - w_{km_I} y_{m_I}(n)) + \\ & + \sum_{l=N+1}^{N+M} (w_{km_R} u_{m_R}(n) - w_{kl_I} u_{m_I}(n)). \end{aligned} \quad (\text{E.28})$$

Derivando esta equação com relação a w_{km_R} e usando a Equação (E.23) obtém-se

$$P_{RRpq}^k(n+1) = \varphi'(v_{k_R}(n+1)) \left(\sum_{m=1}^N (w_{km_R} P_{RRpq}^m(n)) + \delta_{pk} u_{q_R}(n) \right). \quad (\text{E.29})$$

De forma similar, equações recursivas para P_{RI} , P_{IR} e P_{II} podem ser obtidas. Agrupando estas equações, chega-se na seguinte equação matricial

$$\begin{aligned} \begin{bmatrix} P_{RRpq}^k & P_{RIpq}^k \\ P_{IRpq}^k & P_{IIpq}^k \end{bmatrix}_{n+1} = & \begin{bmatrix} \varphi'(v_{k_R}) & 0 \\ 0 & \varphi'(v_{k_I}) \end{bmatrix}_{n+1} \times \\ \times \left(\sum_{l=1}^N \begin{bmatrix} w_{kl_R} & -w_{kl_I} \\ w_{kl_I} & w_{kl_R} \end{bmatrix}_n \begin{bmatrix} P_{RRij}^l & P_{RIpq}^l \\ P_{IRpq}^l & P_{IIpq}^l \end{bmatrix}_n + \delta_{pk} \begin{bmatrix} u_{q_R} & -u_{q_I} \\ u_{q_I} & u_{q_R} \end{bmatrix}_n \right) \end{aligned} \quad (\text{E.30})$$

Finalmente, a atualização dos pesos deve ser feita utilizando-se a seguinte equação

$$\begin{aligned} w_{pq}(n+1) = & w_{pq}(n) + \mu \nabla w_{pq} Q(n) = \\ = & w_{pq}(n) + \mu \sum_{k=1}^N \left([e_{k_R} \ e_{k_I}]_n \begin{bmatrix} P_{RRpq}^k & P_{RIpq}^k \\ P_{IRpq}^k & P_{IIpq}^k \end{bmatrix}_n \begin{bmatrix} 1 \\ j \end{bmatrix} \right). \end{aligned} \quad (\text{E.31})$$

Como no caso real, os parâmetros P das equações recursivas devem ser inicializados com zero.