UNIVERSIDADE FEDERAL DE CAMPINA GRANDE

CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA

Carlos Eduardo Mendes Alves Pinto

EMULAÇÃO DE FILTROS DIGITAIS ATRAVÉS DE REDES NEURAIS ARTIFICIAIS

Campina Grande

Junho 2006

Carlos Eduardo Mendes Alves Pinto

EMULAÇÃO DE FILTROS DIGITAIS ATRAVÉS DE REDES NEURAIS ARTIFICIAIS

Trabalho de conclusão de curso de graduação em engenharia elétrica da Universidade Federal de Campina Grande

Orientador: Prof. Dr. Hiran de Melo

Campina Grande

Junho 2006

Biblioteca Setorial do CDSA. Fevereiro de 2021.

Sumé - PB

Carlos Eduardo Mendes Alves Pinto

EMULAÇÃO DE FILTROS DIGITAIS ATRAVÉS DE REDES NEURAIS ARTIFICIAIS

Trabalho de conclusão de curso de graduação em engenharia elétrica da Universidade Federal de Campina Grande

Aprovado em 18 de junho de 2006

BANCA EXAMINADORA

Prof.: Hiran de Melo – Doutor

Universidade Federal de Campina Grande – UFCG

Prof.: Raimundo Freire – Doutor

Universidade Federal de Campina Grande – UFCG

AGRADECIMENTOS

Gostaria de agradecer ao Prof. Hiran por me aceitar como seu orientando, por toda disposição e ajuda na realização deste trabalho. À minha família que sempre me apoiou e incentivou. À Karina, pela ajuda e incentivo excepcionais. E aos amigos "Alminha", "Leroy" e "Zé Lelé" que me receberam em sua casa e me deram todo o suporte para que eu pudesse concluir este trabalho.

RESUMO

Esse trabalho tem por objetivo o estudo da viabilidade da construção de emuladores de filtros digitais a partir de redes neurais artificiais, utilizando-se o software MATLAB® São apresentados, além de uma revisão bibliográfica de filtros digitais e redes neurais artificiais, uma análise da implementação de dois emuladores neurais, um para um filtro digital passa-baixas e outro para um filtro digital passa-altas.

SUMÁRIO

LISTA DE FIGURAS

1 INTRODUÇÃO

Em circuitos elétricos tanto digitais quanto analógicos é inegável a necessidade da utilização de filtros, pois sempre se precisam separar os sinais desejáveis dos indesejáveis. Mas nem sempre é possível obter os dados necessários para o design do filtro. Uma solução para isso é a utilização de um sistema que tenha a capacidade de aprender e emular o funcionamento do filtro.

Pelas suas próprias características, as Redes Neurais Artificiais apresentam como principal aplicabilidade problemas envolvendo o reconhecimento e a classificação de padrões. A partir daí surgiu a idéia de emular filtros digitais a partir de redes neurais.

A ferramenta utilizada para isso foi o MATLAB®, pois ele apresenta toolbox tanto de processamento digital de sinais quanto de redes neurais. De maneira simples é possível, através deles, criar e testar várias configurações de modo a achar a que apresenta melhores resultados.

O trabalho encontra-se dividido da seguinte maneira: no capítulo 2 apresenta-se uma introdução teórica sobre filtros digitais, mostrando-se alguns conceitos e características básicas; no capítulo 3 é feita uma abordagem sobre redes neurais artificiais, sua história, características e o algoritmo utilizado: o Backpropagation; nos capítulos 4 e 5 trata-se dos principais comandos do MATLAB® utilizados para a implementação do filtro digital de Butterworth e da rede neural artificial; nos capítulos 6 e 7 apresenta-se as implementações de dois emuladores neurais de filtros digitais, um passa-baixas e um passa-altas; no capítulo 8 faz-se uma análise dos resultados obtidos.

2 FILTROS DIGITAIS

A função do filtro em um circuito eletrônico é remover partes indesejadas do sinal, como um ruído, ou extrair uma parte específica do sinal, como uma determinada faixa de freqüência.

Existem basicamente dois tipos de filtros: os analógicos e os digitais. Os filtros analógicos são feitos de circuitos eletrônicos analógicos como resistores, capacitores e amplificadores operacionais e podem ser ativos ou passivos. Os filtros digitais usam um processador, que pode ser um computador ou um PDS (Processador Digital de Sinais), para executar o processamento.

Um filtro digital pode filtrar sinais analógicos da seguinte forma: através da entrada de um sinal analógico, um conversor analógico-digital (ADC) realiza a amostragem do sinal para que ele possa ser lido pelo processador digital de sinais (DSP). Este então fica encarregado de realizar os cálculos necessários para fazer a filtragem. Estes valores, que representam numericamente o sinal filtrado podem ser convertidos, caso seja necessário, através de um conversor digital-analógico (DAC) para obter-se de volta o sinal analógico (INTRODUCTION TO DIGITAL FILTERS, 2006), como pode ser visto na figura 1.

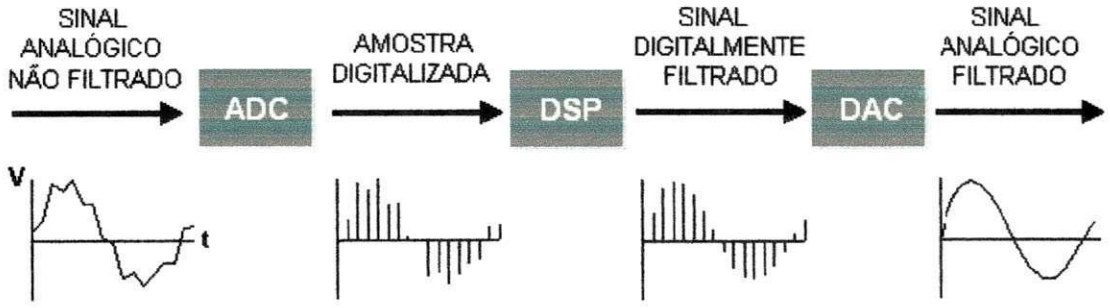

Figura 1: Esquema de Configuração Básica de um Filtro Digital

Nesse trabalho é focada a implementação de filtros digitais utilizando-se o software MATLAB®, especificamente dois: um passa-baixas e um passa-altas, digitais, IIR, usando-se a aproximação de Butterworth. A seguir alguns conceitos básicos sobre eles e suas implementações serão apresentados.

2.1 ESPECIFICAÇÕES PARA O PROJETO DE UM FILTRO

As especificações suficientes e necessárias para o projeto de um filtro são:

- O tipo de filtro de acordo com o tipo de sinal que se deseja filtrar: Passa-baixas, passa-altas,passa-faixa, rejeita-faixa ou passa-tudo;
- Freqüências críticas de corte inferior Ω_P e superior Ω_S
- Ondulação máxima permitida da banda de passagem δ_1 e atenuação mínima δ_2;
- Freqüência de amostragem utilizada: F, (que é igual a $2\times\Omega$);
- Ordem do filtro: N

Na figura 2 são exemplificadas essas especificações para um filtro passa-baixas (OPPENHEIM; SCHAFER; BUCK, 1999):

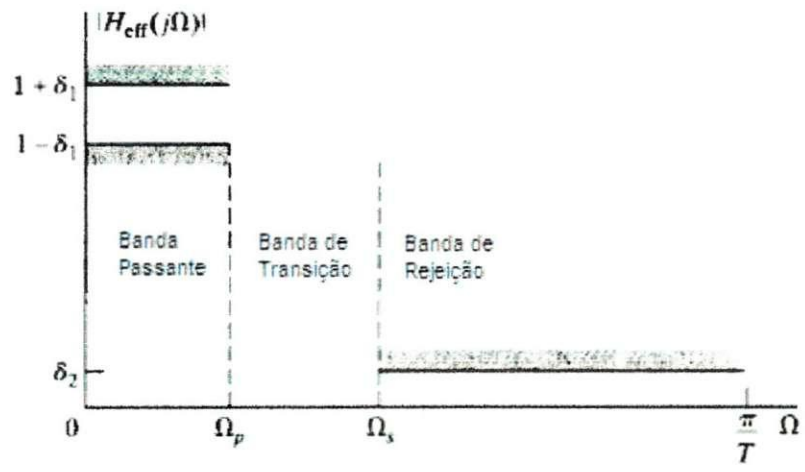

Figura 2: Resposta em Freqüência Idealizada para um Filtro Passa-Baixas

2.2 TIPOS DE FILTROS

Como foi citado anteriormente, deve-se escolher o tipo de filtro de acordo com o a freqüência que se deseja filtrar. Nesse trabalho foram escolhidos os filtros passa-altas e passa-baixas.

2.2.1 Filtro Passa-Baixas

O filtro passa-baixas permite a passagem de sinais de baixas freqüências, atenuando sinais acima da freqüência de corte do filtro. A forma de atenuação do filtro passa-baixas pode ser observada na figura 3 (NETO, 2006).

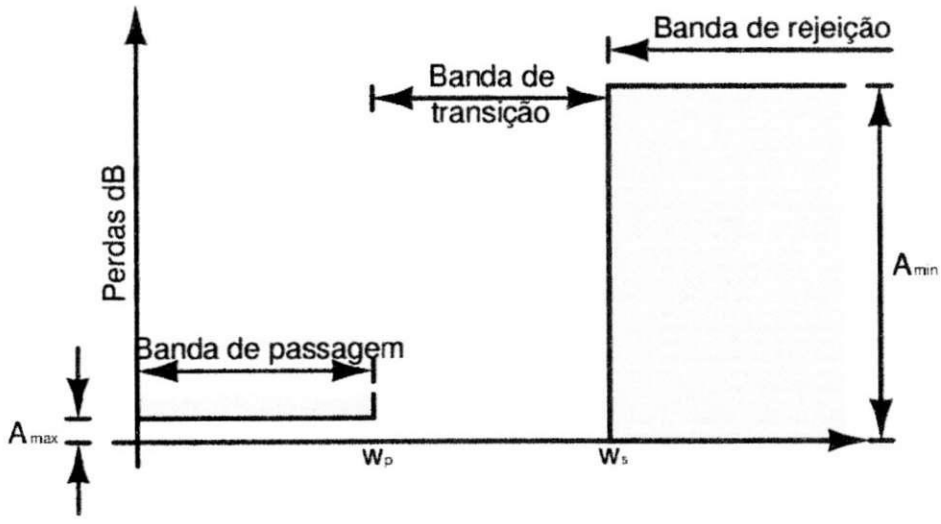

Figura 3: Atenuação do Filtro Passa-Baixas

2.2.2 Filtro Passa-Altas

Permite a passagem de sinais de altas freqüências, bloqueando sinais abaixo da freqüência de corte do filtro. A forma de atenuação do filtro passa-altas pode ser observada na figura 4 (NETO, 2006).

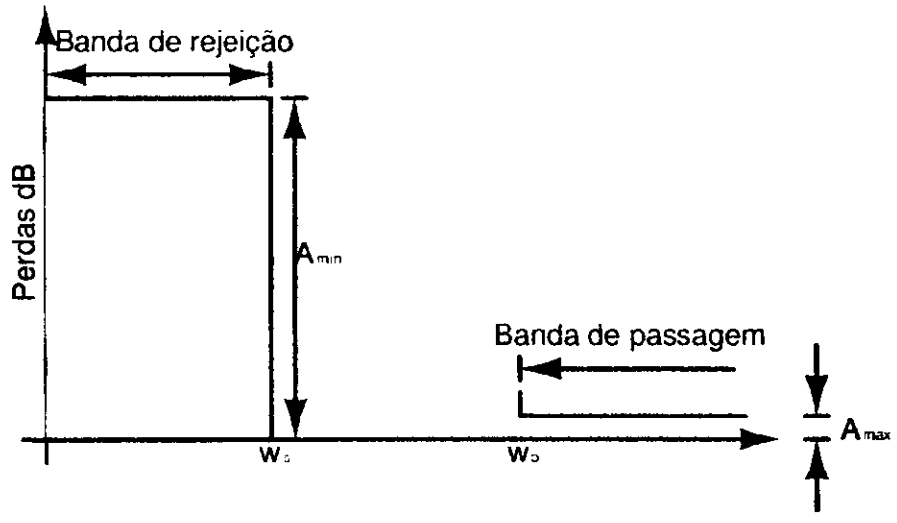

Figura 4: Atenuação do Filtro Passa-Baixas

2.3 APROXIMAÇÃO DE BUTTERWORTH

As principais aproximações de filtro são (SIGNAL, 2006):

- Filtros de Butterworth
- Filtros Elípticos
- Filtros de Chebyshev

Os filtros de Butterworth são definidos de forma que a magnitude da resposta em freqüência seja maximamente plana, na banda de passagem e na banda de rejeição.

Os filtros elípticos possuem a queda mais acentuada de todos, porém apresentam ondulações em toda a largura de banda.

Os filtros de Chebyshev possuem uma queda mais acentuada do que o filtro Butterworth, porém menos acentuada do que o filtro elíptico. Quando a magnitude da sua resposta em freqüência apresenta ondulação na banda de passagem e a banda de rejeição plana eles são do tipo 1. Quando a banda de passagem é plana e apresentam ondulação na banda de rejeição eles são do tipo 2.

Na figura 5 é mostrada a resposta em freqüência do filtro Butterworth junto com outros tipos de filtros obtidos com o mesmo número de coeficientes:

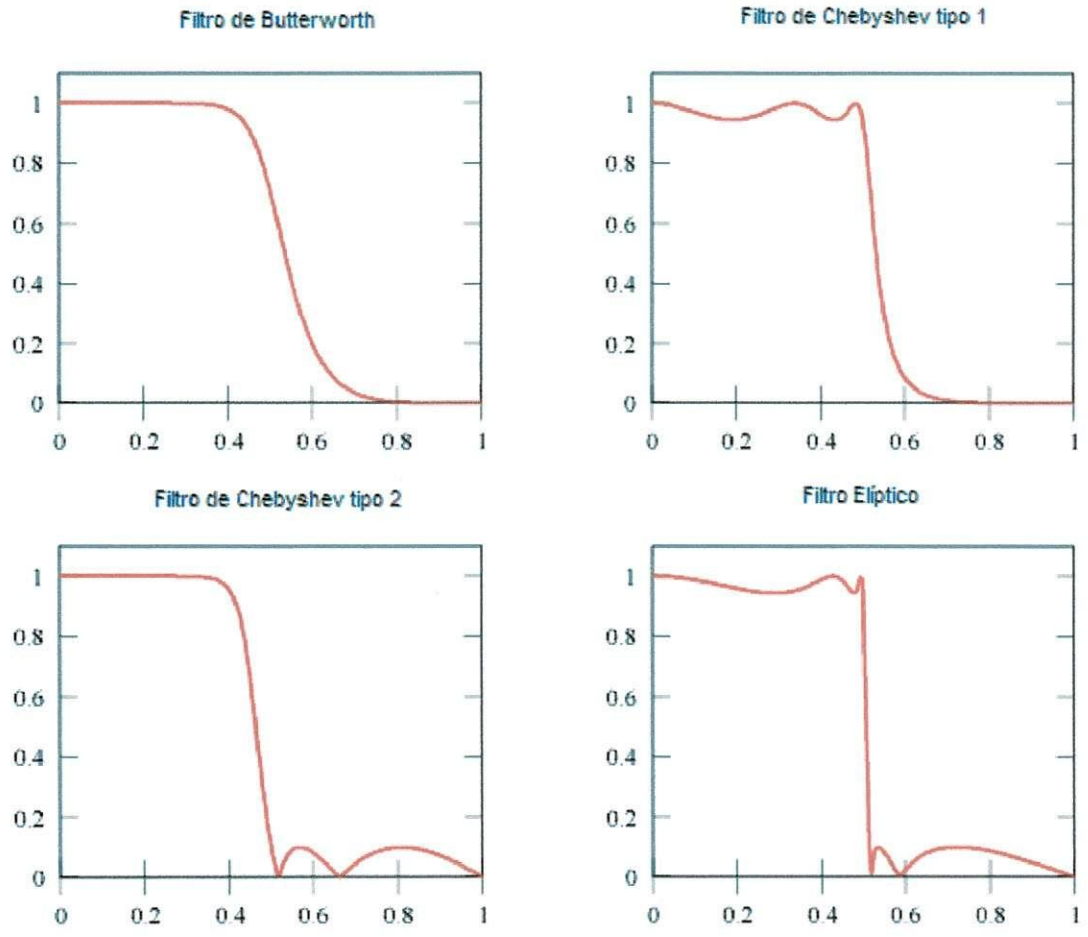

Figura 5: Exemplos de Filtros IIR

A magnitude da resposta em freqüência de um filtro passa-baixas de Butterworth de ordem N pode ser definida matematicamente como (OPPENHEIM; SCHAFER; BUCK, 1999):

$$|H_c(j\Omega)| = \frac{1}{\sqrt{1+(j\Omega / j\Omega_c)^{2N}}}$$

(onde N é a ordem do filtro, Ω é a frequência angular do sinal em radianos por segundo e Ω_c é a frequência de corte). A função está plotada na figura 6.

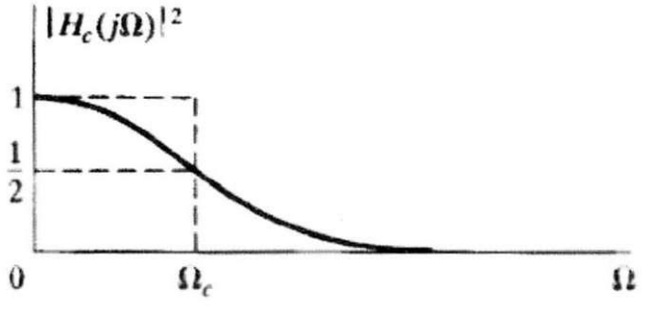

Figura 6 Gráfico da Magnitude para um filtro de Butterworth

A medida que a ordem *N* do filtro aumenta a banda de transição diminui, ou seja, a transição da banda passante para a banda de rejeição ocorre mais rapidamente e de maneira mais íngreme. Mas mantendo o mesmo formato da curva como pode ser visto na figura 7:

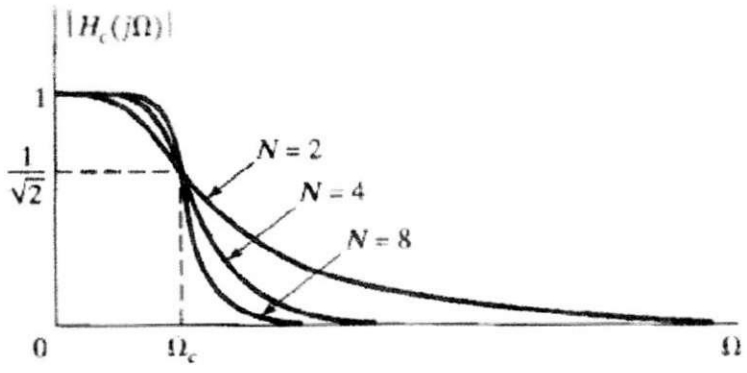

Figura 7: Gráfico da Dependência da Magnitude na Ordem N

A partir da função da magnitude pode-se obter os pólos do filtro. Fazendo-se $j\Omega = s$ tem-se:

$$H_c(s)H_c(-s) = \frac{1}{1+(s/j\Omega_c)^{2N}}$$

Os pólos são as raízes do denominador

$$1+(s/j\Omega_c)^{2N} = 0$$

Isto é:

$$s_k = (-1)^{1/2N}(j\Omega_c) = \Omega_c e^{(j\pi/2N)(2k+N-1)}, \qquad k = 0,1,\ldots,2N-1$$

Com isso tem-se *2N* pólos igualmente espaçados pelo ângulo, no círculo de raio Ω_c, no plano *s*. Na figura 8 é exposto um gráfico do plano *s* para um filtro de Butterworth de ordem 3.

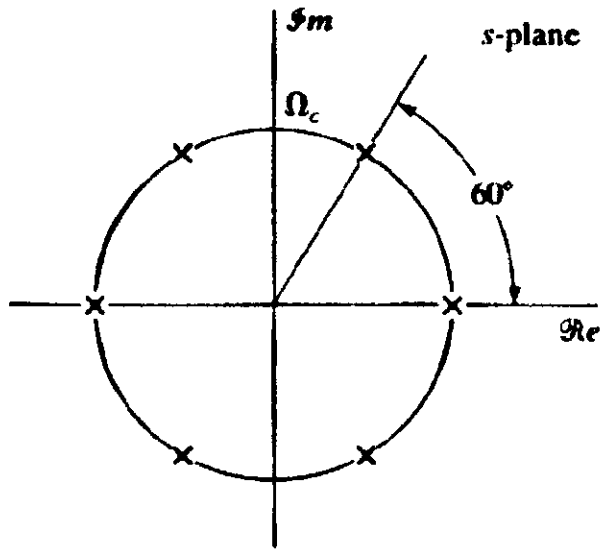

Figura 8: Gráfico da localização dos pólos no plano s para um filtro de Butterworth de ordem 3

2.4 FILTROS DE RESPOSTA INFINITA AO IMPULSO (IIR)

Filtros digitais podem ser implementados de duas maneiras, por convolução (também chamados de Resposta Finita ao Impulso, FIR) ou por recursão (também chamados de Resposta Infinita ao Impulso, IIR).

A resposta de um filtro IIR é função dos sinais de entrada presentes e passados, e dos sinais de saída passados. A equação diferencial a seguir representa um filtro IIR, pois é uma função dos elementos de excitação e resposta.

$$y_n = \sum_{l=0}^{L} A_l . x_{n-1} - \sum_{k=1}^{K} B_k . y_{n-k}$$

A dependência das saídas passadas (recursividade) faz com que a duração da resposta seja infinita, mesmo quando cessaram os sinais de entrada. Devido a

esse fato, ou seja, os sinais de saída já calculados fazerem parte no cálculo dos sinais de saída ainda por calcular, estes filtros também são chamados de filtros recursivos.

A função de transferência do filtro digital IIR pode ser obtida, então, utilizando-se o operador de atraso z^{-1}, onde:

$$x_{n-i} = x.z^{-i}$$
$$y_{n-k} = y.z^{-k}$$

Isso faz com que a função de transferência possa ser escrita da forma:

$$H(z) = \frac{\sum_{i=0}^{M} A_i.z^{-i}}{1 + \sum_{i=0}^{N} B_i.z^{-i}}$$

2.5 PROJETO DO FILTRO DIGITAL IIR

A aproximação para o projeto de filtros digitais IIR não é conceitualmente diferente do problema para projeto de filtros analógicos. A abordagem para o projeto de filtros analógicos envolve uma aproximação analítica das especificações do filtro por uma função de transferência, a partir da qual se projeta uma rede analógica que implemente esta função.

Uma função de transferência realizável é uma das características de uma rede linear estável e causal. Estas características podem ser obtidas fazendo com que a função de transferência seja uma função racional de *s* com coeficientes reais, que os pólos do filtro analógico estejam na metade esquerda do plano *s* e o grau do numerador seja igual ou menor que o grau do polinômio denominador.

O problema do projeto de filtros digitais requer a determinação dos coeficientes da equação diferencial para preencher as características desejadas para o filtro, como resposta em freqüência. Como já existem abordagens clássicas

para o projeto de filtros analógicos, foram desenvolvidas aproximações que mapeiam os pólos e zeros analógicos do plano s para o plano z, de forma a alcançar as características desejadas do filtro digital.

A abordagem tradicional para o projeto de filtros IIR envolve a transformação de um filtro analógico em um filtro digital, com as mesmas especificações (OPPENHEIM; SCHAFER; BUCK, 1999). Esta é uma abordagem razoável, visto que:

- A tecnologia de projeto de filtros analógicos está bastante avançada e, desde que resultados úteis possam ser alcançados, é vantajoso utilizar procedimentos já desenvolvidos para filtros analógicos.
- Muitos métodos de projetos analógicos têm formulas de projeto relativamente simples. Desta maneira, métodos de projetos de filtros digitais baseados nestas fórmulas são também simples de implementar.
- Em muitas aplicações é de interesse utilizar filtros digitais para simular o desempenho de um filtro analógico.

2.6 TRANSFORMAÇÃO BILINEAR

Escolhidos o tipo do filtro e a aproximação, para dar continuidade ao projeto do filtro digital, deve-se escolher o método de transformação do domínio analógico (plano s) para o digital (plano z). Existem várias técnicas para essa transformação, mas a mais usada é a transformação bilinear por poder converter qualquer tipo de filtro (passa-baixas, passa-altas, passa-faixa e rejeita-faixa) (WINDER, 2002, p.397). Além disso, o MATLAB® utiliza a transformação bilinear em suas funções de design de filtros digitais no passo de discretização do filtro (SIGNAL, 2006,)

A transformação bilinear mapeia o plano *s* no plano *z* da seguinte forma:

$$s = k\left(\frac{1-z^{-1}}{1+z^{-1}}\right)$$

(onde *k* é uma constante de distorção) e o mapeamento do eixo $j\Omega$ do domínio contínuo para o círculo unitário do domínio discreto é feito da forma:

$$\omega = 2\tan^{-1}\left(\frac{\Omega}{k}\right)$$

3 REDES NEURAIS ARTIFICIAIS

O ser humano é dotado de circuitos neurais cerebrais complexos que consistem de variadas conexões entre neurônios (sinapses) interagindo entre si fazendo com que surja o comportamento inteligente. Com isso, veio a idéia de modelar computacionalmente estas conexões neurais a fim de obter-se comportamentos inteligentes nas máquinas. Surgiram assim as redes neurais artificiais (RNAs), que são inspiradas na própria natureza das redes de neurônios e sinapses biológicas.

Outro ponto que incentiva a utilização das redes neurais artificiais é que existem classes de problemas que a complexidade algorítmica é muito grande, o que torna inviável a resolução algorítmica convencional.

Além disso, as RNAs são dispositivos que permitem tolerância a ruídos e falhas, isso vem da capacidade de generalização das redes, tornando-as capaz de suportar ruídos e imperfeições no ambiente, ainda mantendo um resultado desejado.

Apesar do pouco conhecimento do cérebro humano e da dificuldade de modelar o que já sabemos, as pesquisas em redes neurais tem se mostrado bastante promissoras em diversas áreas, como engenharia, computação e neurociências. Sendo também de grande utilidade para problemas como reconhecimento de padrões, agrupamento, previsão de séries temporais, etc.

3.1 HISTÓRIA DAS REDES NEURAIS

A história das Redes Neurais Artificiais remonta do ano de 1943 quando McCulloch e Pitts conceberam o primeiro modelo de neurônio artificial. Ele se baseava em uma espécie de modelagem matemática do neurônio biológico (McCulloch & Pitts, 1943). O surgimento do neurônio de McCulloch e Pitts foi

considerado como marco final da chamada *época antiga* das Redes Neurais, caracterizada pelas conquistas da psicologia e neurofisiologia.

A partir daí surge a *Época Romântica* da abordagem conexionista da Inteligência Artificial, esta época foi marcada pelo trabalho de Donald Hebb que sugeriu um modo de proporcionar capacidade de aprendizado às redes neurais artificiais (HEBB,1949). Além disso, cresceu a intenção em buscar inspiração na própria natureza para fazer emergir comportamentos inteligentes. Nesta fase as pesquisas eram limitadas pela baixa tecnologia computacional existente.

Foi na época romântica que o correu o encontro do *Darthmouth College* que foi um encontro conjunto para estudar a Inteligência Artificial (IA) (BARRETO, 1997). Os anos que se seguiram ao encontro em Darthmouth foram de muito otimismo para os pesquisadores de IA. Daí surgiu a idéia de que qualquer problema poderia ser resolvido com inteligência artificial, esta idéia ganhou ainda mais força com o *Perceptron* de Frank Rosenblatt. O *Perceptron* era uma rede neural de duas camadas de neurônios capazes de aprender de acordo com a lei de *Hebb*.

Em 1969, Minsky & Papert em seu livro *Perceptrons (MINSKY & PAPERT,1969)* provaram que a rede proposta anteriormente por Rosenblatt não era capaz de distinguir padrões linearmente separáveis como o problema do OU-Exclusivo.

O livro *Perceptrons* fez as pesquisas em redes neurais artificiais perderem um pouco o fôlego, pois se gerou uma onda de pessimismo na comunidade acadêmica, esta fase ficou conhecida como *Época das Trevas* e durou até 1981. Entretanto, as pesquisas em inteligência artificial continuaram com o intuito de resolver problemas em domínios restritos, contribuindo bastante para o desenvolvimento de sistemas especialistas.

A *Epoca de Renascimento* (1981-1987) caracterizou-se pelo renascimento da inteligência artificial conexionista e serviu como preparação para a *Época Contemporânea,* que se caracteriza pelo crescente aumento na utilização de RNAs. Isso se deve em parte a junção de algumas inovações como o algoritmo de

treinamento *Backpropagation*, proposto por Paul Werbos em 1974, mas que só foi divulgado e popularizado com o trabalho de Rumelhart, Hinton e Williams do grupo PDP ("Parallel Distributed Processing") do MIT em 1986. Além da utilização de várias camadas de neurônios, podem-se considerar importantes também os estudos de John Hopfield que salientava as propriedades associativas de uma classe de redes neurais que apresentava fluxo de dados multidirecional e comportamento dinâmico (ROISENBERG, 1998).

3.2 O NEURÔNIO BIOLÓGICO

Esta seção tem o intuito de apenas contextualizar a inspiração biológica a qual as redes neurais artificiais são baseadas, dando assim uma maior familiaridade com os termos usados.

Um neurônio biológico é composto por um corpo celular ou *soma*, um axônio tubular e várias ramificações arbóreas conhecidas como dendritos. Os dendritos formam uma malha de filamentos ao redor do neurônio. O axônio é um tubo longo e fino que ao final se divide em ramos que terminam em pequenos bulbos que ficam bem próximos dos dendritos dos outros neurônios. O pequeno espaço entre o fim do bulbo e o dendrito é conhecido como sinapse É através da sinapse que as informações se propagam. O número de sinapses recebidas por cada neurônio varia de 100 a 100.000 (NICHOLLS, 2000).

Nas figuras 9 e 10 são mostradas ilustrações de um neurônio artificial e de uma sinapse, respectivamente.

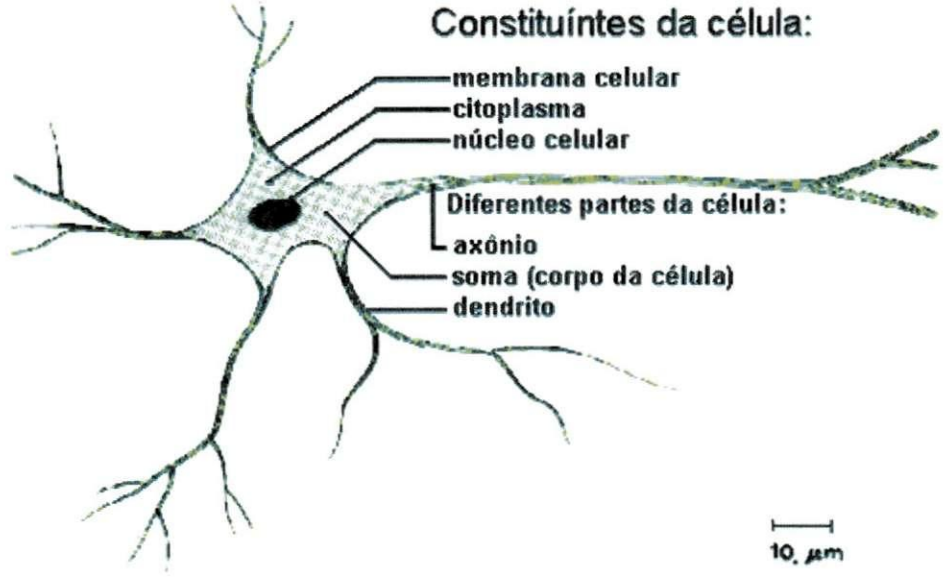

Figura 9: Neurônio Biológico

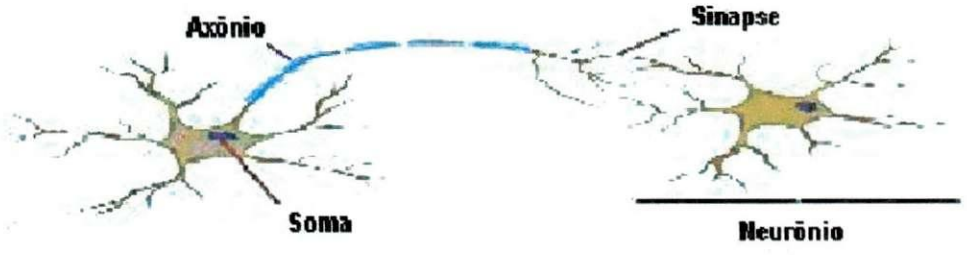

Figura 10: Representação da Sinapse

A célula nervosa tem um potencial de repouso devido aos íons Na+ e K – estarem em concentrações diferentes dentro e fora da célula, de modo que qualquer perturbação na membrana do neurônio provoca uma série de alterações

durante um curto período de tempo.

A alteração na concentração dos íons Na+ e K – gera um trem de pulso que se expande localmente nas proximidades dos dendritos. Dependendo da intensidade do estímulo, este trem de pulso pode exceder um certo limiar no corpo celular e gerar um sinal com amplitude constante ao longo do axônio. Na fronteira, no momento do disparo do neurônio, é gerado um potencial de ação que impulsiona o fluxo do sinal gerado pelo corpo celular para outras células.

O pulso elétrico gerado pelo potencial de ação libera neurotransmissores que são substâncias químicas contidas nos bulbos do axônio, estes neurotransmissores são repassados para os dendritos do neurônio seguinte. Assim, quando o conjunto de neurotransmissores que chegam aos dendritos de um determinado neurônio atinge certo limiar, eles disparam de novo um potencial de ação que vai repetir todo o processo novamente. Convém ressaltar que as sinapses podem ser excitatórias, facilitando o fluxo dos sinais elétricos gerados pelo potencial de ação, como podem também ser inibitórias que tem como característica dificultar a passagem desta corrente.

3.3 O NEURÔNIO ARTIFICIAL

Uma rede neural artificial (RNA) é composta por várias unidades de processamento que são conectadas por canais de comunicação e estão associadas a determinado peso. O comportamento inteligente de uma RNA vem das interações entre as unidades de processamento da rede.

A operação de uma unidade de processamento, proposta por McCullock e Pitts (McCulloch & Pitts, 1943), pode ser resumida da seguinte maneira:

- Sinais são atribuídos às entradas;
- Cada sinal é multiplicado por um número, ou peso, que indica a sua influência na saída da unidade;

- É feita a soma ponderada dos sinais que produz um nível de atividade;
- Se este nível de atividade exceder um certo limite (threshold) a unidade produz uma determinada resposta de saída, de acordo com a função de limiar do neurônio.

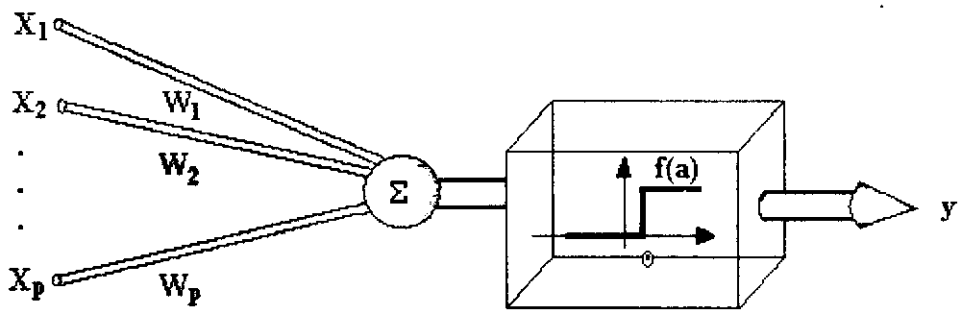

Figura 11: Neurônio Artificial

Para p sinais de entrada X_1, X_2,..., X_p e pesos w_1, w_2,..., w_p e limitador t; com sinais assumindo valores booleanos (0 ou 1) e pesos valores reais.

Neste modelo, por exemplo, o nível de atividade a é dado por:

$$a = \sum_{i=1}^{p} w_i X_i$$

A saída y é dada por

$y = 1$, se $a \geq t$ ou

$y = 0$, se $a < t$

3.4 TOPOLOGIA DAS REDES NEURAIS ARTIFICIAIS

Para a maioria dos problemas práticos um único neurônio não é suficiente. Com isso torna-se necessário utilizarem-se neurônios interconectados tornando a

decisão de como interconectar os neurônios uma das mais importantes decisões em um projeto de uma rede neural artificial.

Na interligação dos neurônios é comum o uso de camadas intermediarias (ou ocultas) que permitem as RNAs implementarem superfícies de decisão mais complexas. Estas camadas permitem que seus elementos se organizem de tal forma que cada elemento aprenda a reconhecer características diferentes do espaço de entrada, assim, o algoritmo de treinamento deve decidir que características devem ser extraídas do conjunto de treinamento.

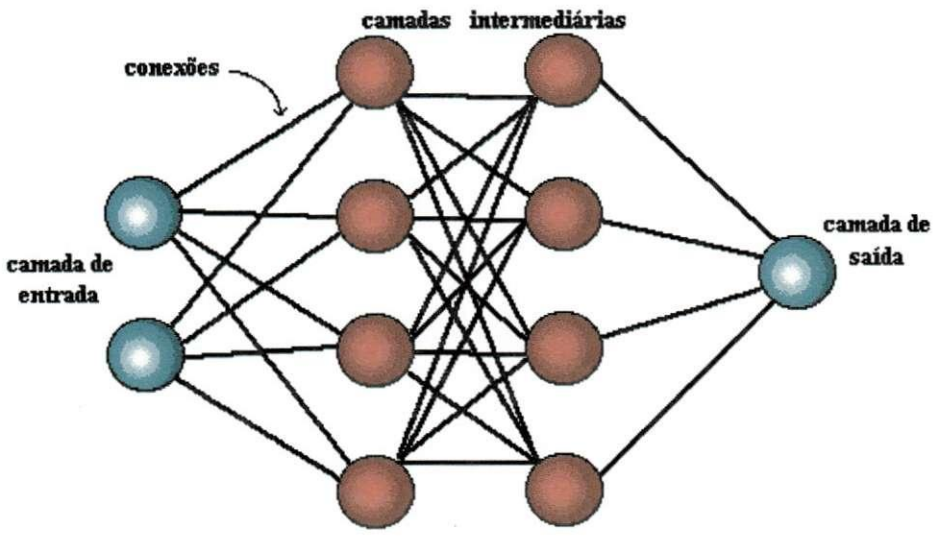

Figura 12: Organização em Camadas

Usualmente as camadas são classificadas em três grupos:

- **Camada de Entrada**: onde os padrões são apresentados à rede;
- **Camadas Intermediárias ou Escondidas**: onde é feita a maior parte do processamento, através das conexões ponderadas; podem ser consideradas como extratoras de características;
- **Camada de Saída**: onde o resultado final é concluído e apresentado.

As redes neurais artificiais podem ser diretas ou recorrentes, sendo que a principal diferença entre elas é que nas redes diretas os neurônios não recebem realimentação em suas entradas. Atualmente as redes neurais diretas são as mais utilizadas principalmente pelo advento da popularização do algoritmo de treinamento *backpropagation*. Este tipo de rede pode ser considerado um aproximador universal de funções, sendo que seu nível de precisão depende principalmente do número de neurônios e da escolha eficiente do conjunto de exemplos (MATHWORKS, 2006).

Nas redes neurais recorrentes existe pelo menos um ciclo de retroalimentação, onde algum neurônio fornece o seu sinal de saída para a entrada de outro neurônio. Este tipo de rede se comporta melhor com problemas dinâmicos e que necessitam de dinâmica da rede neural (HAGAN, 1996).

3.5 ALGORITMO *BACKPROPAGATION*

O algoritmo *backpropagation* provê um aprendizado supervisionado, isto é, ele procura achar iterativamente a mínima diferença entre as saídas desejadas e as saídas obtidas pela rede neural, segundo um erro mínimo. Desta forma, ajustando os pesos entre as camadas através da retropropagação do erro encontrado em cada iteração (HAYKIN, 1998).

Este algoritmo utiliza camadas intermediarias para superar o problema do aprendizado da classificação de padrões não-linearmente com isso ele pode implementar superfícies de decisão mais complexas. A característica principal da camada escondida é que seus elementos se organizam de tal forma que cada elemento aprende a reconhecer características diferentes do espaço de entrada, assim, o algoritmo de treinamento deve decidir que características devem ser extraídas do conjunto de treinamento.

A rede aprende um conjunto pré-definido de pares de exemplos de entrada e saída em ciclos de propagação e adaptação. Depois que um padrão de entrada

é aplicado como um estímulo aos elementos da primeira camada da rede, ele é propagado por cada uma das outras camadas até que a saída é gerada. Este padrão de saída é então comparado com a saída desejada e um sinal de erro é calculado para cada elemento de saída.

O sinal de erro é então retropropagado da camada de saída para cada elemento da camada intermediária anterior que contribui diretamente para a formação da saída. Cada elemento da camada intermediária recebe apenas uma porção do sinal de erro total, proporcional apenas à contribuição relativa de cada elemento na formação da saída original. Este processo se repete, camada por camada, até que cada elemento da rede receba um sinal de erro que descreva sua contribuição relativa para o erro total. Baseado no sinal de erro recebido, os pesos das conexões são então atualizados para cada elemento de modo a fazer a rede convergir para um estado que permita a codificação de todos os padrões do conjunto de treinamento. Esse algoritmo é mostrado a seguir:

1 - Inicializar os pesos e coeficientes de limiar com valores pequenos e randômicos.

2 - Apresentar o vetor de entrada (padrão) e a saída desejada.

3 - Calcular a saída:

$$S(t) = \sum_{i=1} p_i(t) e_i(t) - \theta$$

(onde p_i é um elemento do vetor de pesos, e_i é um elemento do vetor de entradas, S a saída do elemento processador, t o número da amostra de entrada e θ o coeficiente de limiar).

4 - Aplicar a função sigmoid:

$$Sq(t) = sig(S(t))$$

5 - Calcular o erro da última camada:

$$\varepsilon S(t) = S(t)(1 - S(t))(d(t) - S(t))$$

(onde S é a saída linear e d a saída desejada) e atualizar os pesos:

$$P(t+1) = P(t) + \alpha \varepsilon S(t) En(t)$$

(onde P é o vetor de pesos, α é o coeficiente de aprendizado e En o vetor resultante da saída da camada anterior).

6 - Calcular os erros da(s) camada(s) intermediária(s):

$$\varepsilon_i = En(t)(1 - En(t)) \sum_k \varepsilon_k p_{ik}(t)$$

(onde En é o vetor resultante da saída da camada anterior até esta camada intermediária; k é o número de nodos conectados a seguir do atual; ε_k é o erro do nodo k, p_{ik} é o peso correspondente à conexão do nodo i atual com o nodo k) e ajustar os pesos:

$$P(t+1) = P(t) + \alpha \varepsilon_i(t) En(t) + \mu\big(P(t-1)\big)$$

(onde μ é um coeficiente de aceleração de convergência denominado *momentum*).

7 - Voltar ao passo 2 até que atinja um valor próximo ao da saída desejada.

4 IMPLEMENTAÇÃO DE FILTROS DIGITAIS NO MATLAB®

O principio de projeto dos filtros digitais no MATLAB® é baseado na conversão do filtro analógico clássico para o seu equivalente digital. O filtro escolhido para o trabalho foi o filtro digital de Butterworth. A seguir serão mostradas os comandos utilizados no seu projeto e implementação.

4.1 SELEÇÃO DA ORDEM DO FILTRO

Inicialmente deve-se escolher a ordem do filtro que se deseja implementar. Para isso utiliza-se a função BUTTORD:

```
[N,Wn] = BUTTORD(Wp,Ws,Rp,Rs)
```

onde:

`Wp` = Freqüência da banda passante normalizada;

`Ws` = Freqüência da banda de rejeição normalizada;

`Rp` = Atenuação máxima;

`Rs` = Atenuação mínima.

essa função retoma:

`N` = A menor ordem do filtro digital de Butterworth para os parâmetros dados acima;

`Wn` = A freqüência natural de Butterworth (ou a freqüência de 3 dB). Ela é utilizada na função BUTTER para obter-se as especificações desejadas.

4.2 DESIGN DO FILTRO DIGITAL DE BUTTERWORTH

Com a ordem do filtro definida pode-se obtê-lo com a função BUTTER:

```
[B,A] = BUTTER(N,Wn)
```

(onde N e Wn foram definidos anteriormente).

Essa função cria um filtro digital de Butterworth de N-ésima ordem e retorna os coeficientes da função de transferência do filtro. Os coeficientes são listados na ordem decrescente de z.

`B` = Numerador da função de transferência do filtro;

`A` = Denominador da função de transferência do filtro.

4.3 FILTRAGEM DO SINAL

Com o filtro projetado utiliza-se a função FILTER para filtrar o sinal desejado:

```
Y = FILTER(B,A,X)
```

essa função retorna o vetor Y que é o vetor X filtrado pelo filtro digital que apresenta a função de transferência com os coeficientes dados por B e A. Do tipo:

```
a(1)*y(n) = b(1)*x(n) + b(2)*x(n-1) + ... + b(nb+1)*x(n-nb)
                - a(2)*y(n-1) - ... - a(na+1)*y(n-na)
```

5 IMPLEMENTAÇÃO DE REDES NEURAIS NO MATLAB®

5.1 CRIAÇÃO DA REDE NEURAL

Para criar uma rede neural artificial multicamadas *retroalimentada* a partir do algoritmo *backpropagation* utiliza-se a função NEWFF:

```
NET = NEWFF(PR,[S1 S2...SNi],{TF1 TF2...TFN},BTF,BLF,PF)
```

onde:

`PR` = Matriz Rx2 dos valores min e max para a entrada dos R elementos;

`Si` = Tamanho da ni-ésima camada, para Ni camadas;

`TFi` = Função de transferência de cada camada, default = 'tansig';

`BTF` = Função que será usada pra realizar a backpropagation, default = 'trainlm';

`BLF` = Pesos/bias da função de aprendizagem do backpropagation, default = 'learngdm';

`PF` = Função de desempenho, default = 'mse'.

5.2 FUNÇÕES DE TRANSFERÊNCIA DO NEURÔNIO

Os Neurônios podem apresentar diversas funções de transferência. As mais comuns são a log-sigmóide, a tangente-sigmóide e a linear. Nas figuras 13, 14 e 15 são mostrados seus gráficos seguidos da simbologia utilizada no MATLAB® (SIGNAL, 2006):

- **Logsig**

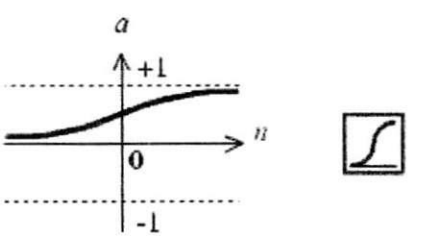

Figura 13: Função de Transferência Log-Sigmóide

- **Tansig**

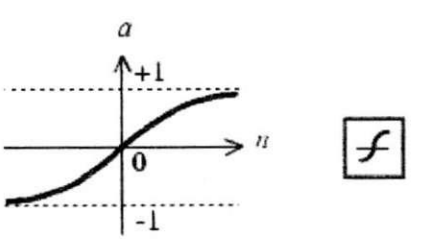

Figura 14: Função de Transferência Tangente-Sigmóide

- **Purelin**

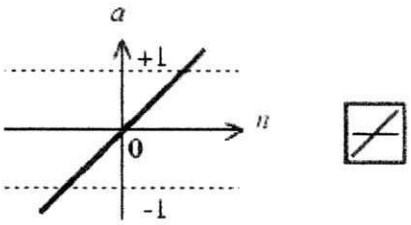

Figura 15: Função de Transferência Linear

5.3 TREINAMENTO

A função utilizada para treinar a rede neural é a função TRAIN. Os principais parâmetros utilizados são:

```
net = TRAIN(NET,P,T)
```

onde,

NET = Rede neural artificial;

P = Dados de entrada da rede neural;

T = Alvo da rede neural.

essa função retorna:

net = Rede neural treinada.

A rede neural é treinada, então, de acordo com a função `NET.trainFcn` e com os parâmetros `NET.trainParam`.

5.4 SIMULAÇÃO

Para simular a rede neural utiliza-se a função SIM.

```
A = SIM(net,p)
```

onde:

`net` = Rede neural criada;

`p` = Matriz de entradas.

essa função retorna uma matriz de saídas da rede.

6 PROJETO DOS FILTROS DIGITAIS

Nesse capítulo serão utilizadas as técnicas abordadas no capítulo 4 para implementar dois tipos de filtros digitais de Butterworth: um passa-baixas e um passa-altas.

Para ambos foram escolhidos os seguintes parâmetros:

- Freqüência de amostragem: $f_a = 128Hz$
- Freqüência de Nyquist $\left(f_n = \frac{f_a}{2} \right)$: $f_n = 64Hz$
- Atenuação na banda passante: $R_p = 3dB$
- Atenuação na banda de rejeição: $R_s = 20dB$

As rotinas do MATLAB® que implementam os resultados das seções 6.1 e 6.2 encontram-se nos Apêndices B e C, respectivamente.

6.1 RESULTADOS DO PROJETO DO FILTRO DIGITAL PASSA-BAIXAS

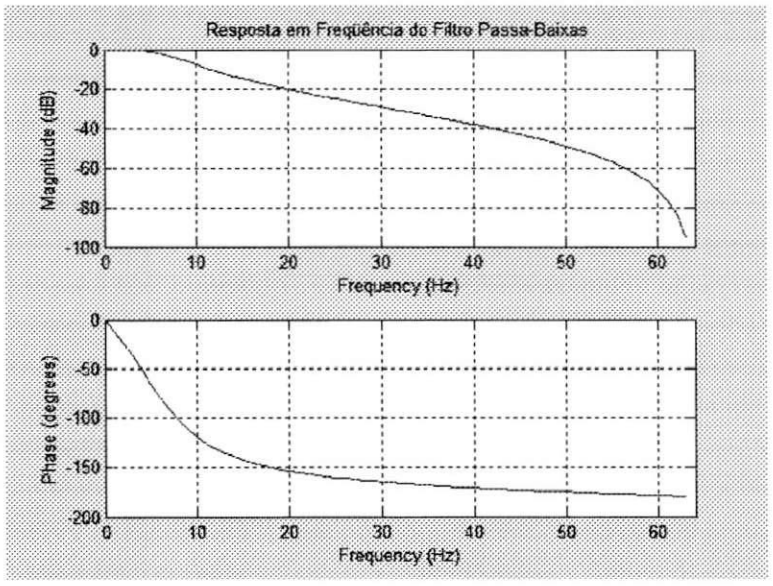

Figura 16: Resposta em Freqüência do Filtro Passa-Baixas

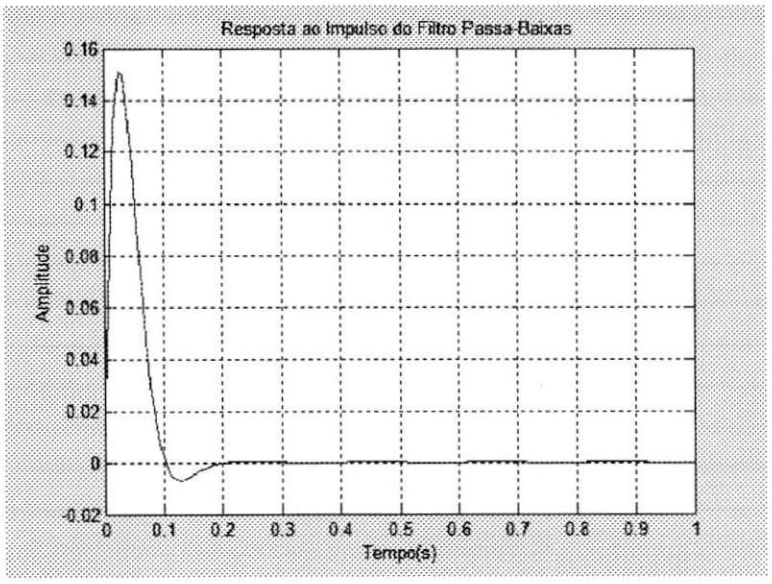

Figura 17 Resposta ao Impulso do Filtro Passa-Baixas

6.2 RESULTADOS DO PROJETO DO FILTRO DIGITAL PASSA-ALTAS

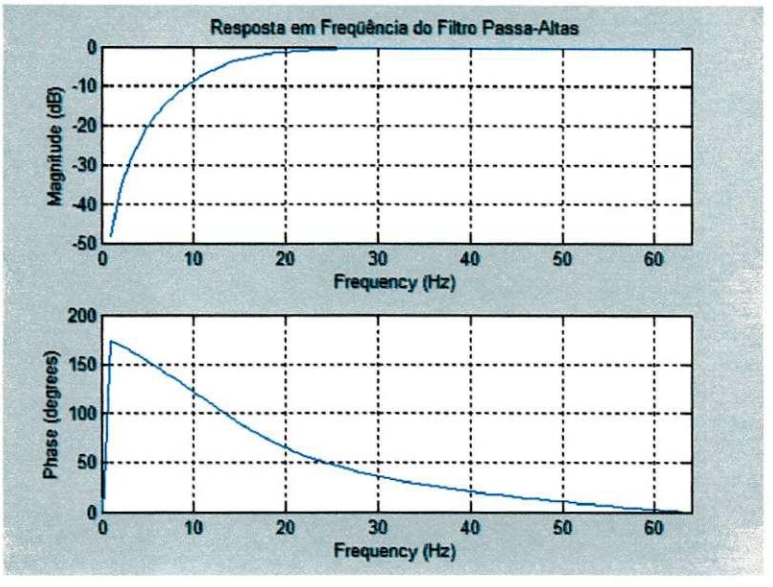

Figura 18: Resposta em Freqüência do Filtro Passa-altas

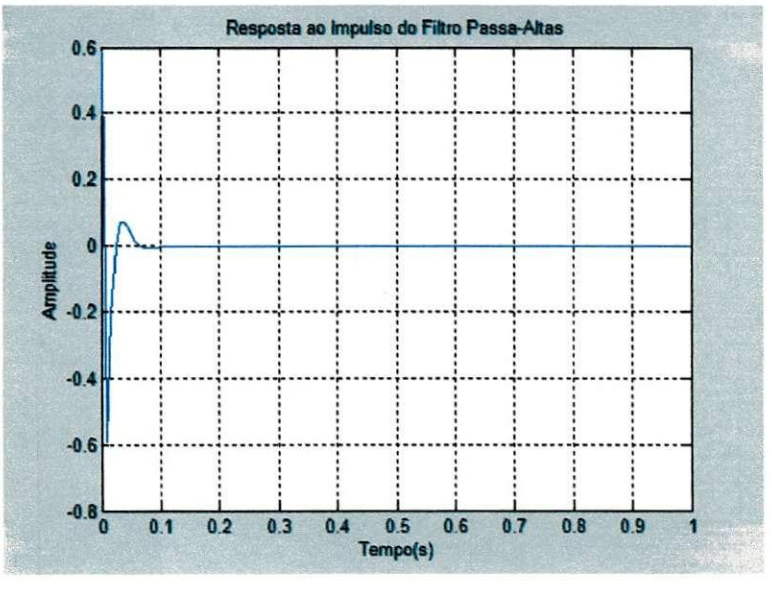

Figura 19: Resposta ao Impulso do Filtro Passa-altas

7 PROJETO DOS EMULADORES NEURAIS

Dados os filtros neurais implementados no capítulo 6, tem-se por objetivo emulá-los utilizando Emuladores Neurais baseados em Redes Neurais de Múltiplas Camadas (RNMC).

O treinamento de uma rede Neural consiste basicamente de quatro passos:

- Montagem dos dados de treinamento;
- Criação da Rede Neural
- Treinamento da Rede Neural
- Simulação da Rede Neural através de novas entradas

O sinal escolhido para treinar os emuladores neurais foi:

```
X = sin(2*pi*t*3)+sin(2*pi*t*50)
```

A partir dele foi composta uma matriz de entradas, juntamente com amostras do sinal depois de passar pelo filtro, caracterizando o emulador como uma malha recursiva. Como alvo foi utilizado o mesmo sinal filtrado pelos filtros digitais mostrados no capítulo 6.

A RNMC é uma rede de duas camadas não linear, pois possui em sua camada intermediária uma função *tangente-sigmoide* e em sua camada de saída uma função *linear*. O algoritmo de treinamento é o `trainlm` (Levenberg-Marquardt). Ela possui a seguinte arquitetura:

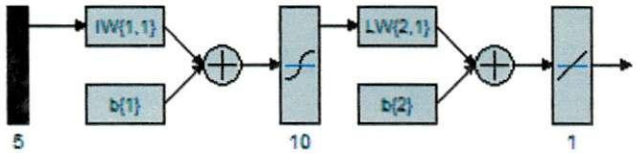

Figura 20: Arquitetura da Rede Neural

Os valores do pesos iniciais são aleatórios e o treinamento da rede Neural cessa ao atingir um desses parâmetros:

```
net.trainParam.epochs=epocas; %Número de épocas. Default:
epocas = 30000

net.trainParam.goal=rms; % Valor Médio Quadrático. Default:
rms = 1e-15
```

As rotinas do MATLAB® que implementam os resultados das seções 7.1 e 7.2 encontram-se nos Apêndices B e C, respectivamente.

7.1 RESULTADOS DO PROJETO DO EMULADOR NEURAL PASSA-BAIXAS

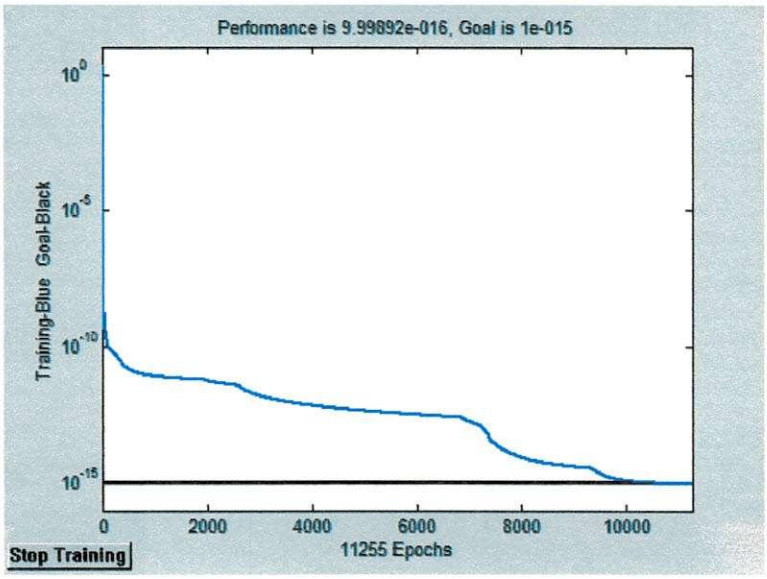

Figura 21: Treinamento do Emulador Neural Passa-Baixas

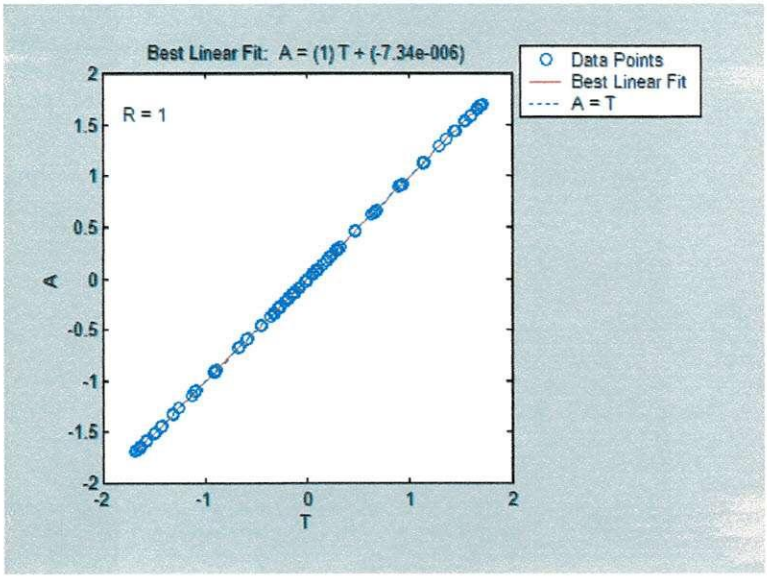

Figura 22: Análise do Desempenho da Generalização do Emulador Neural Passa-Baixas

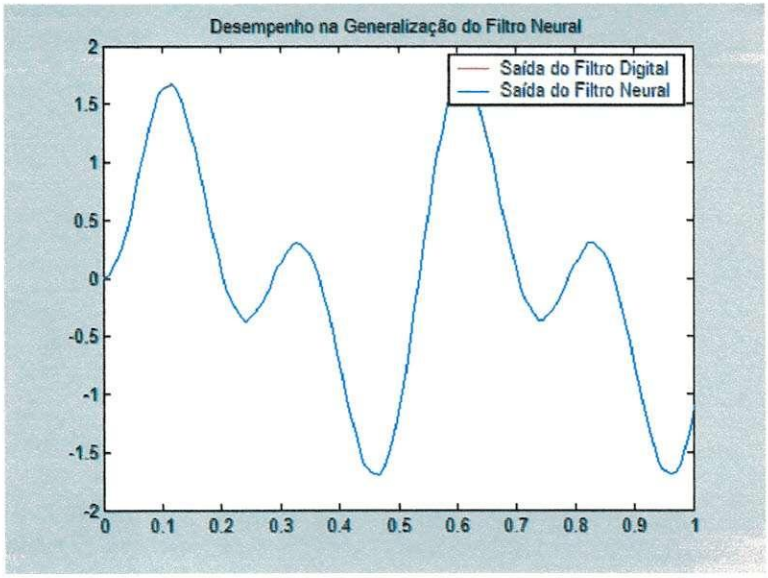

Figura 23: Análise do Desempenho no Domínio do Tempo do Emulador Neural Passa-Baixas

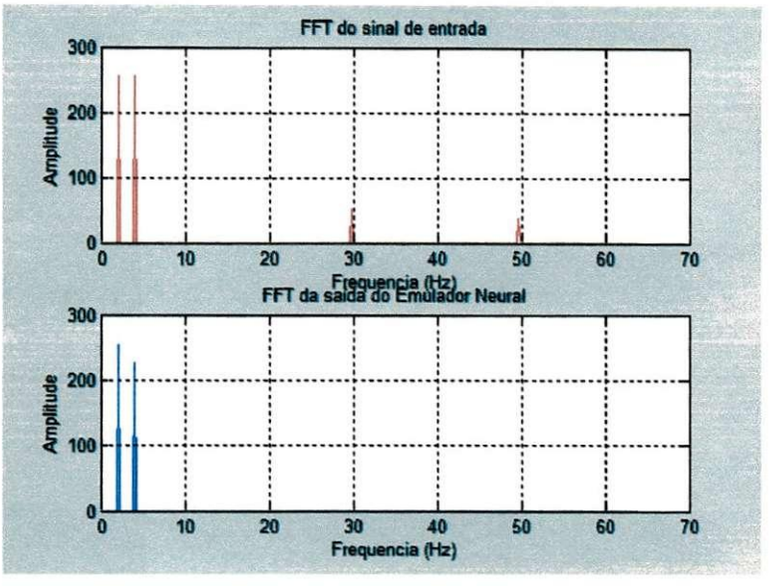

Figura 24: Análise do Desempenho no Domínio da Freqüência do Emulador Neural Passa-Baixas

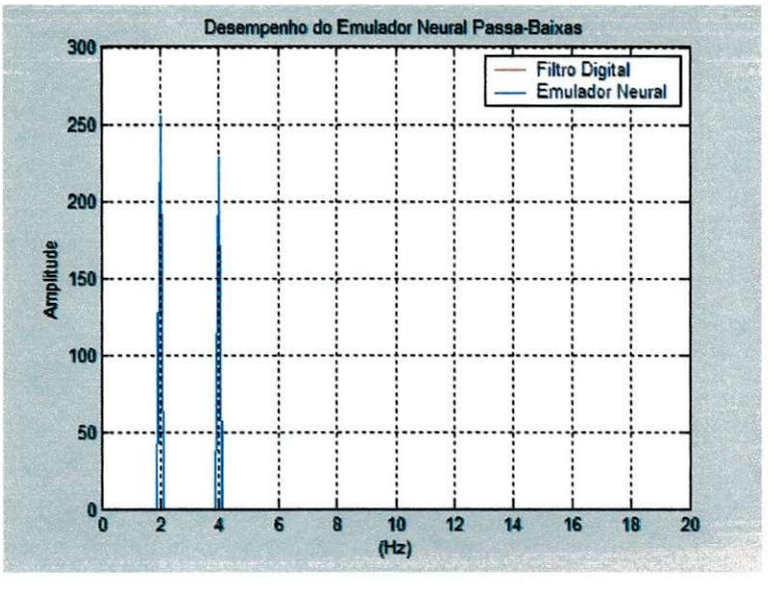

Figura 25: Comparação Entre o Filtro Digital e o Emulador Neural Passa-Baixas

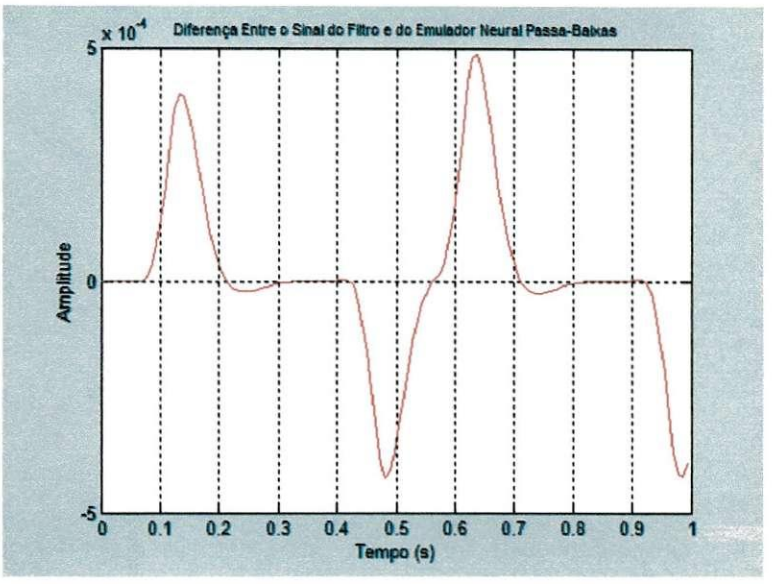

Figura 26: Diferença Entre o Sinal Filtrado Pelo Filtro Digital e Pelo Emulador Neural Passa-Baixas

7.2 RESULTADOS DO PROJETO DO EMULADOR NEURAL PASSA-ALTAS

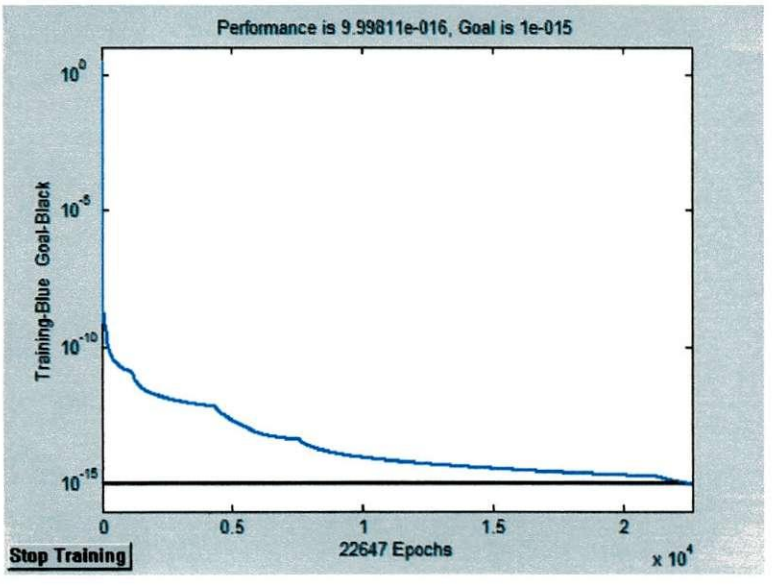

Figura 27: Treinamento do Emulador Neural Passa-Altas

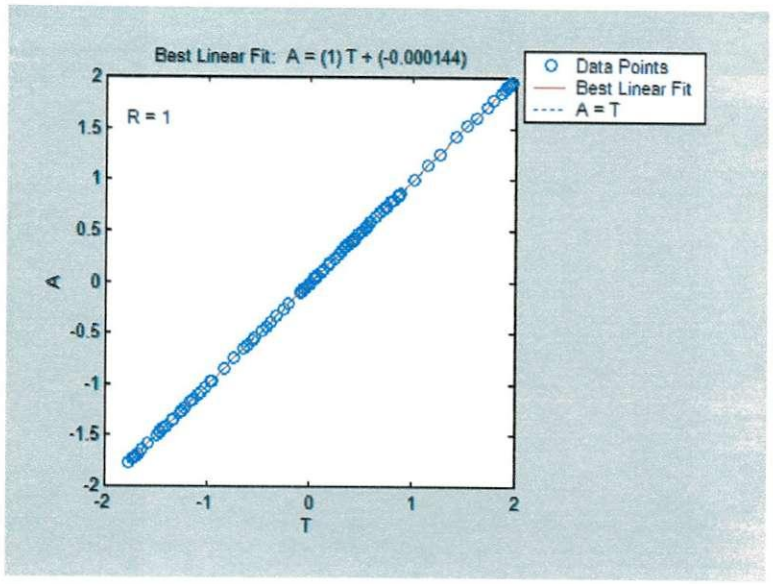

Figura 28: Análise do Desempenho da Generalização do Emulador Neural Passa-Altas

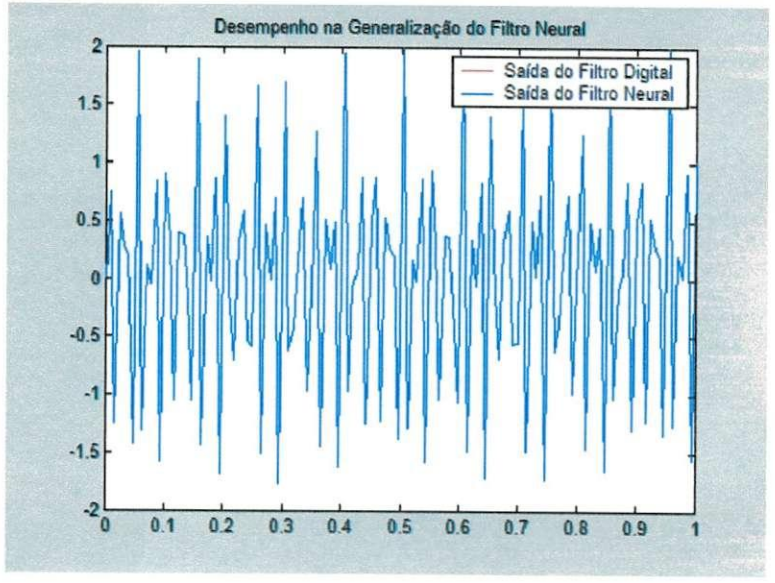

Figura 29: Análise do Desempenho no Domínio do Tempo do Emulador Neural Passa-Altas

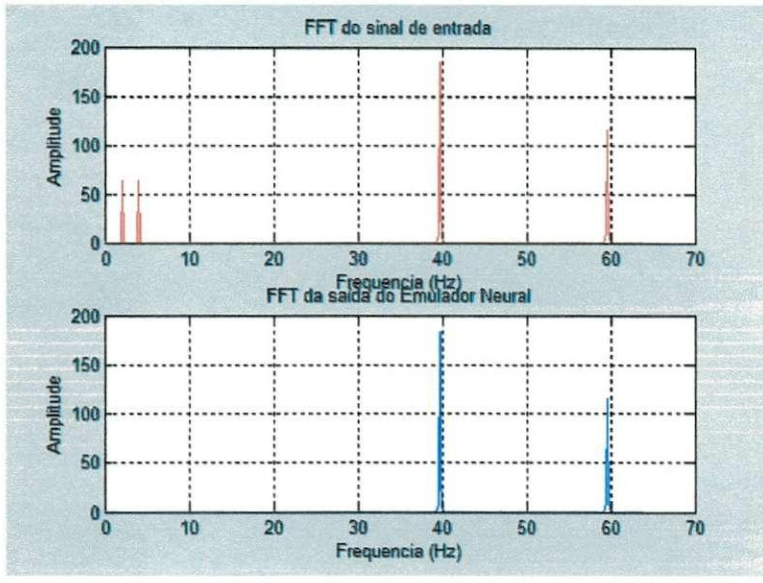

Figura 30: Análise do Desempenho no Domínio da Freqüência do Emulador Neural Passa-Altas

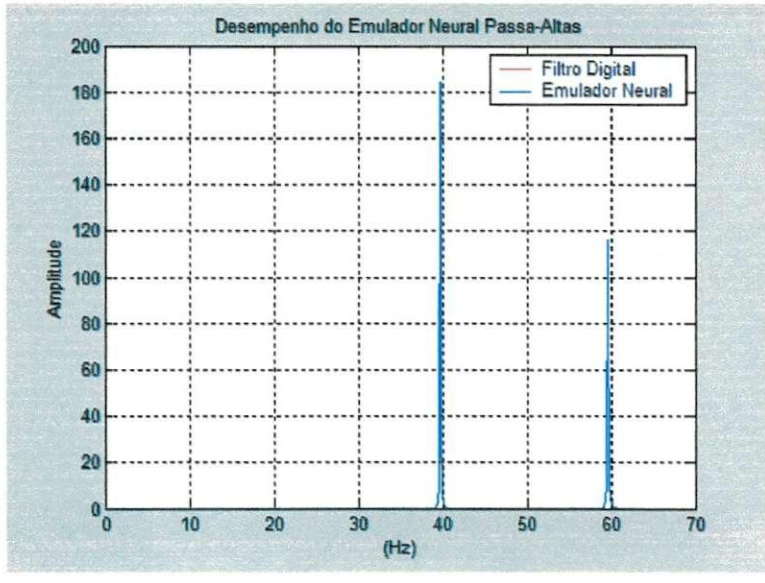

Figura 31: Comparação Entre o Filtro Digital e o Emulador Neural Passa-altas

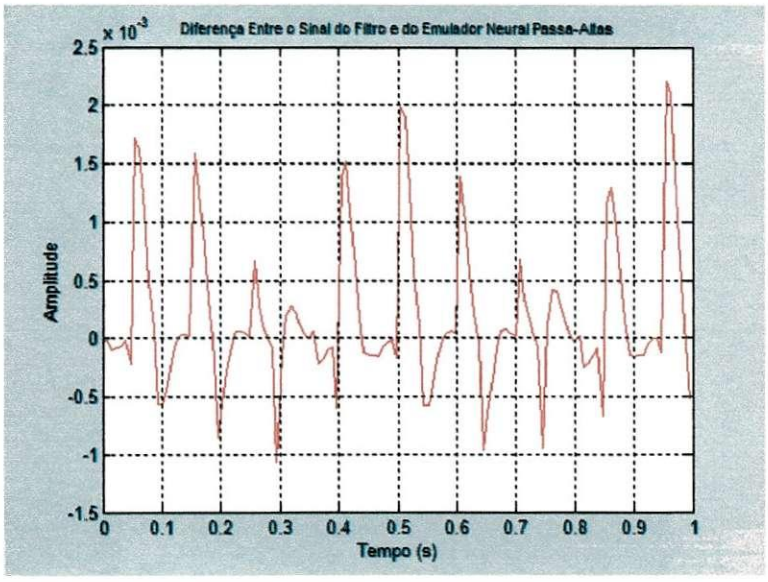

Figura 32: Diferença Entre o Sinal Filtrado Pelo Filtro Digital e Pelo Emulador Neural Passa-altas

8 ANÁLISE DOS RESULTADOS

Como pôde ser observado os emuladores neurais conseguem emular os filtros digitais. Mas o número de épocas necessário para obter-se o resultado desejado varia, pois os valores dos pesos iniciais são aleatórios.

Existem várias configurações de redes neurais possíveis, além de vários algoritmos de treinamento. A rede neural de duas camadas *tansig/purelin* (que apresenta uma função de ativação tangente-sigmoide na camada intermediária e uma função linear na camada de saída) foi escolhida devido à sugestão dada pelo guia do usuário do Toolbox de Redes Neurais do MATLAB® (NEURAL 2006, cap5, p.7): "Essa rede pode ser usada como uma aproximadora geral de funções. Ela pode aproximar praticamente qualquer função com um número finito de descontinuidades, dados um número suficiente de neurônios na camada escondida".

Resultados mais precisos e com um número menor de épocas podem ser obtidos usando-se uma rede neural de duas camadas *purelin/purelin* (que apresenta funções de ativação linear tanto na camada intermediária quanto na camada de saída), como pode ser observado no Apêndice A. Isso se deve ao fato de que a saída dos neurônios com funções sigmóides são limitadas a uma pequena faixa de valores (NEURAL 2006, cap5, p.5).

O desempenho dos emuladores neurais pode ser medido de várias maneiras, como a análise dos erros e comparações entre os alvos e os resultados emulados. Entretanto o MATLAB® possui uma rotina chamada `postreg` que serve justamente para realizar essa comparação e análise. Os resultados dessa análise foram mostrados nas figuras 22 e 28. Essa função retorna o valor do coeficiente de correlação entre as saídas e os alvos (variável R) e uma análise gráfica. O coeficiente de correlação é uma medida de quão perfeita é a correlação entre os alvos e as saídas. Quanto mais próximo esse número é de 1, melhor a correlação. A saída gráfica plota as saídas da rede neural versus a os alvos como círculos. O

melhor ajuste linear é mostrado pela linha pontilhada e o resultado perfeito (que é a saída igual aos alvos) é mostrado pela linha sólida.

Nos emuladores neurais projetados, os valores de R obtidos foram iguais a 1 e a linha pontilhada praticamente sobrepõe a linha sólida. O que indica um bom resultado.

Nas figuras 23 e 29 pode-se verificar os resultados da saída do filtro digital e do emulador neural no domínio do tempo. Pode-se perceber que os gráficos sobrepõem-se o que é um indicativo, também, de bom desempenho.

A sobreposição das linhas pode ser percebida também nos gráficos das figuras 25 e 31, onde são expostas as saídas do filtro digital e do emulador neural no domínio da freqüência.

Nas figuras 24 e 30 pode-se perceber que o emulador neural exerce a função de filtro digital pela comparação do sinal de entrada com o sinal de saída do emulador neural, para ambos os filtros.

Mas como a análise visual da superposição das linhas dos gráficos não é tão precisa, pode-se verificar através das figuras 26 e 32 os gráficos das diferenças entre os sinais filtrados pelos filtros digitais e os sinais filtrados pelos emuladores neurais. Pode-se perceber que essas diferenças ficam na faixa de 10^{-3} a 10^{-4} o que representa uma boa precisão. Essa variação na diferença deve-se ao fato dos parâmetros iniciais serem escolhidos aleatoriamente. Mas em todos os testes realizados elas encontraram-se nessa faixa.

CONCLUSÃO

O trabalho apresentou a implementação de dois filtros digitais, um passa-altas e um passa-baixas seguidos de suas emulações através de redes neurais artificiais de múltiplas camadas.

Levando-se em conta o grande número de tipos de redes neurais artificiais e suas possíveis configurações, para a configuração escolhida os resultados obtidos foram satisfatórios. Isso mostra que é possível treinar uma rede neural artificial para que ela desempenhe o papel de um filtro digital.

O uso do MATLAB® como ferramenta de criação dos filtros e das redes mostrou-se bastante construtivo devido a sua facilidade de uso e por ele já apresentar em sua biblioteca um toolbox de design de filtros e um toolbox redes neurais. Devido ao fato da não necessidade de criação de algoritmos para a emulação das redes, puderam-se testar várias configurações até que se achasse a mais eficiente.

Devido ao grande número de topologias e algoritmos de treinamento das redes neurais pode-se sugerir como trabalhos futuros a emulação usando outros tipos de filtros e outras configurações de redes neurais. Como o estudo de redes neurais é relativamente recente suas utilidades ainda encontram-se muito inexploradas tornando vasto o campo de pesquisa nessa área.

9 REFERÊNCIAS

BARRETO, JORGE M. **Introdução às Redes Neurais Artificiais,** Florianópolis: UFSC. 2002. 57p.

BARRETO, JORGE M. **Inteligência Artificial**: No limiar do século XXI. Florianópolis: Duplic Edições,1997. 392p.

HAYKIN, SIMON, **Neural Network:** A Comprehensive Foundation, 2.ed. Hamilton: Prentice Hall, 1998. 842p.

HEBB, DONALD, **The Organization of Behavior** : A Neuropsychological Theory. New York: John Wiley & Sons, 1949. 368p

HAGAN, MARTIN T; BEALE, MARK H, DEMUTH, HOWARD B, **Neural Network Design.** 1.ed. Boulder, Colorado: PWS Publishing, 1996.

INTRODUCTION TO DIGITAL FILTERS. Disponivel em <http://www.dsptutor.freeuk.com/digfilt.pdf >. Acesso em 14 jun. 2006

MCCULLOCH, W.S. & PITTS, W. **A Logical Calculus of the Ideas Imminent in Nervous Activity**. Bulletin of Mathematical Biophysics 5, 1943, 115-33.

MINSKY, M. L. & PAPERT, S. A. **Perceptrons**: An Introduction to Computacional Geometry. Expanded Edition. Massachussets: The MIT Press, 1969. 275p.

NETO, OTACILIO A. R. **Projeto de Filtros Ativos Utilizando Ferramentas Computacionais**. Campina Grande, 2006. Trabalho de Conclusão de Curso do Curso de Graduação em Engenharia Elétrica da UFCG.

NICHOLLS, J. G. et a.l From **Neuron to Brain. Sinauer Associates**, Inc., 2000.

OPPENHEIM, ALAN V.; SCHAFER,RONALD W. **Discrete-Time Signal Processing,** 2.ed. Upper Saddle River: Prentice-Hall, 1999. 879p.

ROISENBERG, M. **Emergência da Inteligência em Agentes Autônomos através de Modelos Inspirados na Natureza**. Florianópolis, 1998. Tese de Doutorado. Departamento de Engenharia Elétrica, Universidade Federal de Santa Catarina.

SMITH, STEVEN W. **The Scientist and Engineer's Guide to Digital Signal Processing,** 2.ed. San Diego: California Technical Publishing, 1999. 640p.

SIGNAL Processing Toolbox User's Guide, The MathWorks, Disponível em: < http://www.mathworks.com/access/helpdesk/help/toolbox/signal/>. Acesso em 14 jun. 2006.

NEURAL Network Toolbox User's Guide. The MathWorks Disponível em: < http://www.mathworks.com/access/helpdesk/help/toolbox/nnet/>. Acesso em 14 jun. 2006.

VALENÇA, MÊUSER, **Aplicando Redes Neurais:** Um Guia Completo, 1.ed. Olinda: Livro Rápido, 2005. 284p.

WINDER, STEVE, **Analog and Digital Filter Design**, 2.ed. [S.l.]: Newnes, 2002. 450p.

APÊNDICE A – EMULADOR NEURAL PASSA-BAIXAS LINEAR

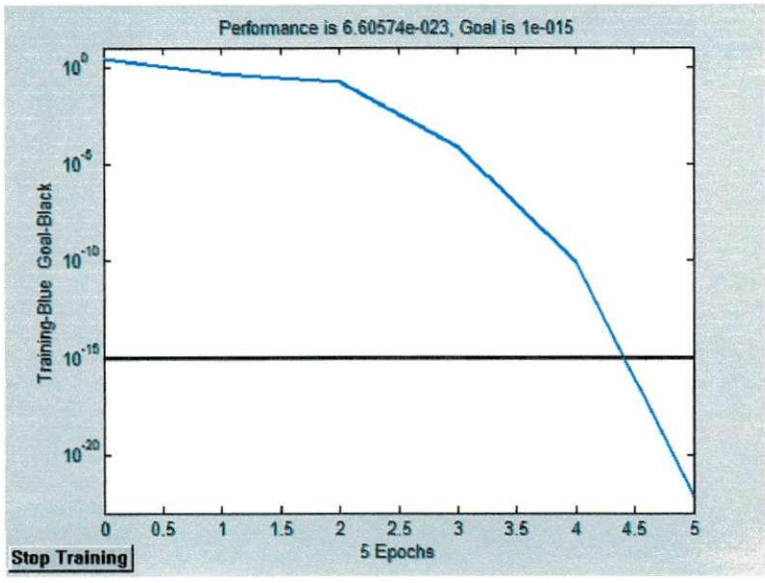

Figura 33: Treinamento do Emulador Neural Passa Baixas Linear

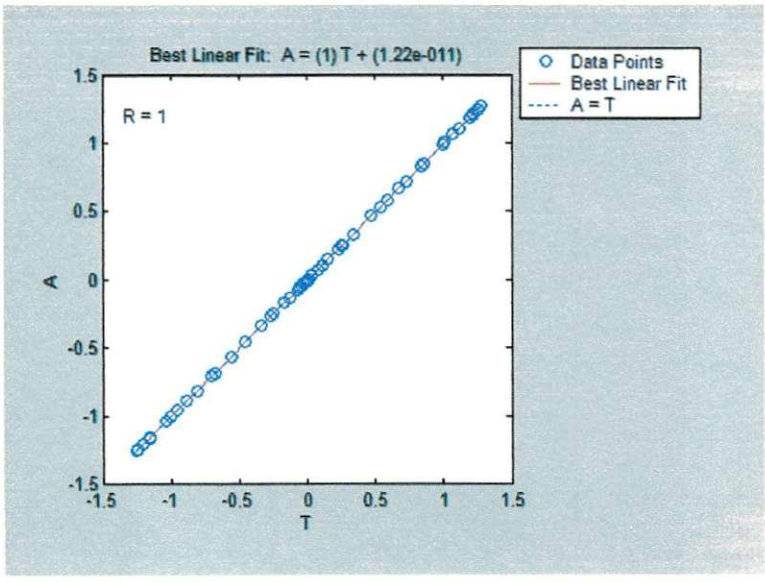

Figura 34: Análise do Desempenho da Generalização do Emulador Neural Passa-Baixas Linear

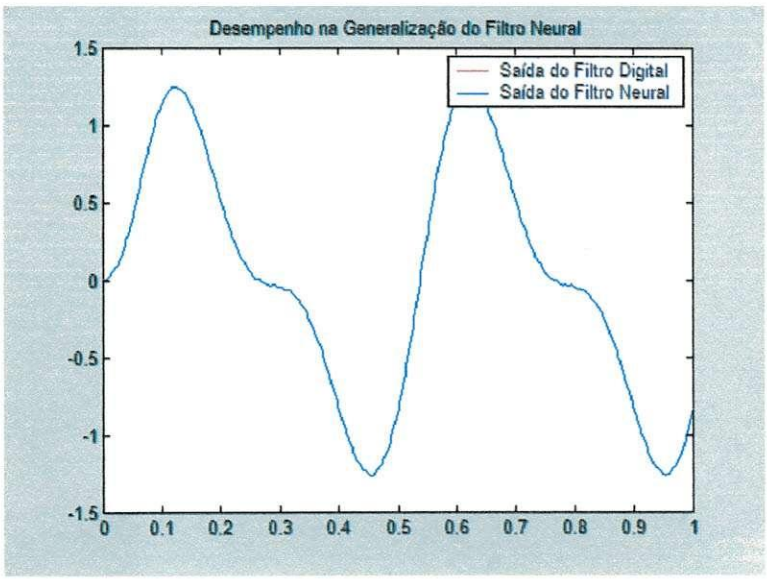

Figura 35: Análise do Desempenho no Domínio do Tempo do Emulador Neural Passa-Baixas Linear

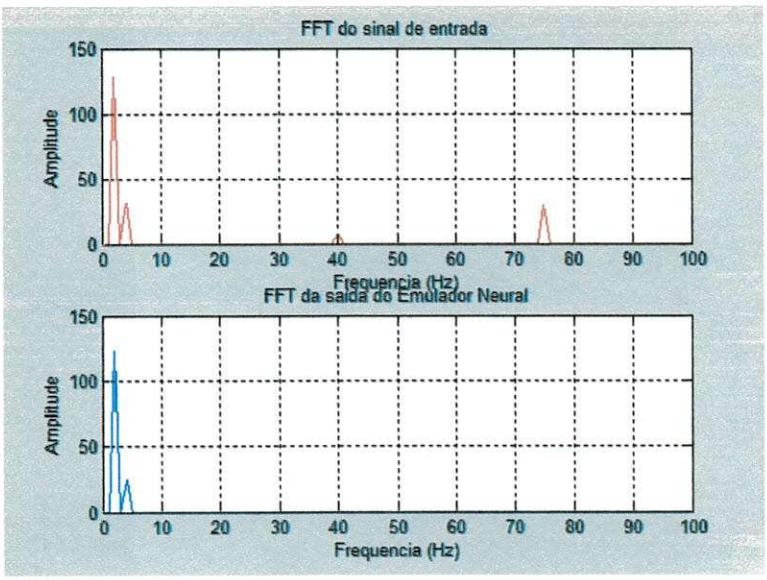

Figura 36: Análise do Desempenho no Domínio da Freqüência do Emulador Neural Passa-Baixas Linear

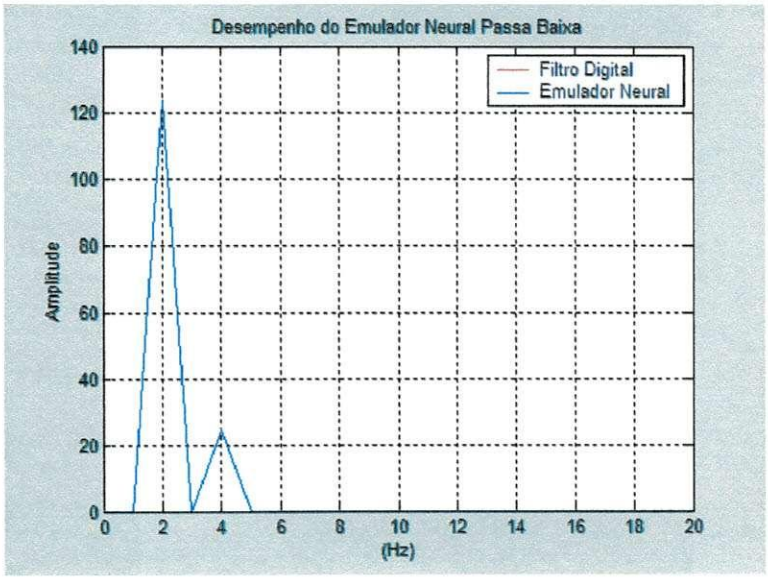

Figura 33: Comparação Entre o Filtro Digital e o Emulador Neural Passa-Baixas Linear

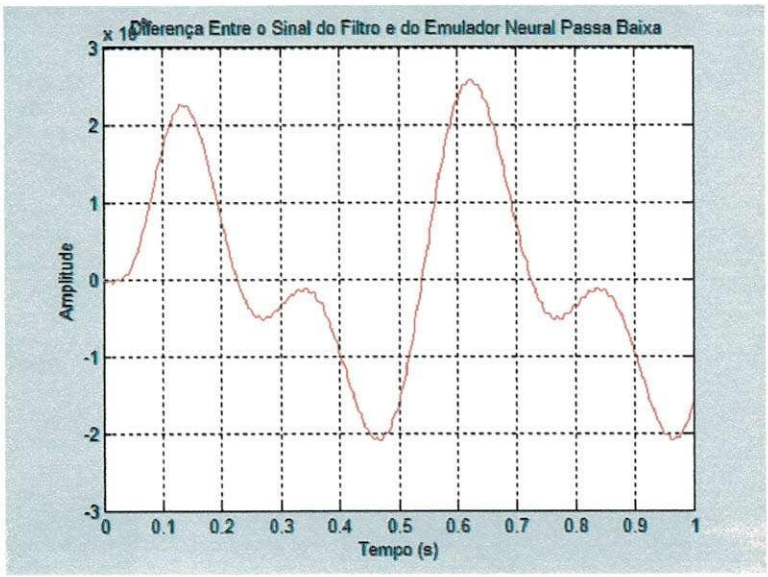

Figura 34: Diferença Entre o Sinal Filtrado Pelo Filtro Digital e Pelo Emulador Neural Passa-Baixas Linear

APÊNDICE B – ROTINA DO MATLAB® QUE IMPLEMENTA O EMULADOR NEURAL PASSA-BAIXAS

```
%
%  Script File: EmuladorNeuralPB.m
%
%  Este roteiro MATLAB desempenha a função de um Filtro Passa-Baixas
%  de Ordem 2, tipo Butterworth mediante a utilização de uma Rede Neural
%  e compara o resultado com a saída de um filtro digital.
%
%*****************************************************************
% Autores: Hiran de Melo e Carlos Eduardo Mendes Alves Pinto
% % Departamento de Engenharia Elétrica
% e@mail: hiran@dee.ufpb.edu.br
%         a20121117@grad.dee.ufcg.edu.br
% Criado: 02/02/2005
% Atualizado: 15/06/2006
%*****************************************************************
%
close all;
clear all;
clc

% ***** A. Definição dos sinais ****

% 1. Freqüência de amostragem = 128Hz
fa = 128;
t = 0:1/fa:1;
X = sin(2*pi*t*3)+sin(2*pi*t*50); % Sinal de entrada, X(t)
%X = sin(2*pi*t*3)+0.2*randn(size(t));

% ***** B. Projeto do Filtro Butterworth Passa-Baixas *****

% 1. Freqüência de Nyquist
fn = fa/2;

% 2. Normalização dos valores das freqüências

% 2.1 faixa de passagem: [0, 5Hz]
fp = 5/fn;

% 2.2 faixa de bloqueio: [20Hz, fn]
fs = 20/fn;

% 3. Especificações das atenuações mínima e máxima
Rp = 3; % atenuação na banda passante: 3 dB
Rs = 20; % atenuação na banda de rejeição: 20 dB

% 4. Ordem do Filtro e freqüencia de corte normalizada
[N,Wn] = buttord(fp,fs,Rp,Rs);

% 4.1 Saída do Filtro para o sinal de entrada X(t)
[B,A] = butter(N,Wn);
```

```
figure(1);
freqz(B,A,64,fa)
title('Resposta em Freqüência do Filtro Passa-Baixas')
pause(1);

% 5. Função de Transferência do Filtro Passa-Baixas
F = filter(B,A,X);

% 6. Resposta ao Impulso do Filtro Passa-Baixas

% 6.1 Definição do Impulso unitário
X1 = zeros(1,size(X,2));
X1(1) = 1;

% 6.2 Determinação da resposta ao impulso unitário
FImp = filter(B,A,X1);
figure(2);
plot(t,FImp);
title('Resposta ao Impulso do Filtro Passa-Baixas')
xlabel('Tempo(s)'),ylabel('Amplitude'),grid


% *****  C. Projeto do Emulador Neural Passa-Baixas *****

% 1. Definição da Arquitetura da Rede Neural:

% Camada de entrada: 10 neurônios com 5 entradas cada
% Camada de saída: 1 neurônio com 5 entradas.
% Notação: RNMC(5,10,1)

% 2.  Construção da matriz de entrada

% 2.1 Determinação do número amostras do sinal de entrada.
Q=size(X,2);

% 2.2 Definição das dimensões da matriz de entrada
P = zeros(5,Q);

% 2.3 Composição da matriz a partir do sinal de entrada, X(t)
P(1,1:Q) = X(1,1:Q);
P(2,2:Q) = X(1,1:(Q-1));
P(3,3:Q) = X(1,1:(Q-2));

P(4,2:Q) = F(1,1:(Q-1));
P(5,3:Q) = F(1,1:(Q-2));

% 3. Criação da Rede Neural
  net = newff(minmax(P), [10 1], {'tansig', 'purelin'}, 'trainlm'); %Rede
Neural Multicamadas
% net = newff(minmax(P), [10 1], {'tansig', 'purelin'}, 'trainbr');
% net = newff(minmax(P), [5 1], {'purelin', 'purelin'}, 'trainlm');
% net = newelm(minmax(P), [5 1], {'tansig', 'purelin'});
% net = newff(minmax(P), [10 1], {'tansig', 'purelin'}, 'traingdx');
%  net  =  newff(minmax(P),  [10  10  1],  {'tansig',  'tansig',  'purelin'},
'trainbr');

% 3.1 Inicialização da RNMC
```

```
net.layers{1}.initFcn='initwb';
net.inputWeights{1,1}.initFcn='rands';
net.layerWeights{2,1}.initFcn='rands';
net.biases{1}.initFcn='rands';
net.biases{2}.initFcn='rands';

net=init(net);

% 3.2 Simulaçao da Rede Neural sem treinamento:
y1 = sim(net,P);

% 4. Treinamento da Rede Neural
disp(['    '])
tpr= input ('Treinar Filtro Neural? : Sim(1) ou Não (0) tpr =  ');
if isempty(tpr), tpr=1; end;
%
if tpr==1,
%
disp(['     Parâmetros da Rede Neural'])
disp(['    '])
epocas= input ('Números Máximo de Treinamentos:(30000) epocas =  ');
if isempty(epocas), epocas=30000; end;
disp(['    '])
rms= input ('Mínimo Erro Médio Quadrático(1e-15) : rms =  ');
if isempty(rms), rms=1e-15; end;
disp(['    '])
%
net.trainParam.epochs=epocas;
net.trainParam.show=epocas/5;
net.trainParam.goal=rms;
net.trainParam.min_grad=rms*1e-20;

net = train(net,P,F);

else
   load Filtro01w1;load Filtro01w2;
   load Filtro01b1;load Filtro01b2;
net.IW{1,1}=w1;
net.LW{2,1}=w2;
net.b{1}=b1;
net.b{2}=b2;
end;
%
w1=net.IW{1,1};
w2=net.LW{2,1};
b1=net.b{1};
b2=net.b{2};
%
save Filtro01w1 w1; save Filtro01w2 w2;
save Filtro01b1 b1; save Filtro01b2 b2;

% 4.1 Simulação da Rede Neural com treinamento
y2 = sim(net,P);
figure(3);
plot(X,F,'o',X,y1,'*',X,y2,'+')
title('Relação do Sinal de Entrada com a Rede Neural')
legend('Filtro Digital','Sem Treinamento', 'Com Treinamento')
```

```
pause(1);

% 5. Analise de Desempenho do Treinamento da Rede Neural
y2r = y2(1:10:end);
Fr = F(1:10:end);
figure(4);
title('Valor de Correlação entre as Saídas e os Alvos')
[m,b,r] = postreg(y2r,Fr);
pause(1);
r % Valor de correlação entre as saidas e os alvos. Quanto mais prox de 1
melhor.

% 5.1 Analise de Desempenho no Domínio do Tempo
figure(5);
plot(t,F,'r',t,y2,'b')
title('Treinamento: Desempenho do Filtro Neural')
legend('Saída do Filtro Digital', 'Saída do Filtro Neural')
pause(1);


% ***** D. Teste de Generalização *****

% 1. Novo Sinal de entrada, Z(t)
Q = 8*Q;
t = linspace(0,8,Q);

%Z = sin(2*pi*t*3);
%Z = sin(2*pi*t*3)+.25*sin(2*pi*t*30);
%Z = sin(2*pi*t*3)+.5*sin(2*pi*t*40);
%Z = sin(2*pi*t*3)+0.5*randn(size(t));
Z = sin(2*pi*t*2)+sin(2*pi*t*4)+.5*sin(2*pi*t*30)+.5*sin(2*pi*t*50);

% 1.1 Saída do Filtro para o novo sinal de entrada Z(t)
F1 = filter(B,A,Z);

% 2. Composição da matriz a partir do sinal de entrada, Z(t)
P1 = zeros(5,Q);
P1(1,1:Q) = Z(1,1:Q);
P1(2,2:Q) = Z(1,1:(Q-1));
P1(3,3:Q) = Z(1,1:(Q-2));

S(1) = sim(net,P1(:,1));
P1(4,2) = S(1);
S(2) = sim(net,P1(:,2));
P1(4,3) = S(2);
P1(5,3)= S(1);
S(3) = sim(net,P1(:,3));

for jj=4:Q
P1(4,jj) = S(jj-1);
P1(5,jj) = S(jj-2);
S(jj) = sim(net,P1(:,jj));
end

% 3. Simulação da rede neural generalizada sem novo treinamento
y3 = sim(net,P1);
```

```
% 4. Análise de desempenho da generalização da rede neural
y3r = y3(1:10:end);
Fr = F1(1:10:end);
figure(6);
title(' Valor de Correlação entre as Saidas e os Alvos')
[m,b,r1] = postreg(y3r,Fr);
pause(1)
r1 % Valor de correlação entre as saidas e os alvos. Quanto mais prox de
1 melhor.

% 4.1 Análise de desempenho no domínio do tempo
figure(7);
plot(t,F1,'r',t,y3,'b')
axis([0 1 -2 2])
title('Desempenho na Generalização do Filtro Neural')
legend('Saída do Filtro Digital', 'Saída do Filtro Neural')
pause(1);

% 4.2. Análise de desempenho no domínio da Freqüência
nfft = length(Z);
nfft1 = nfft/2;
nfft2 = nfft1-1;
f = fa*(0:nfft2)/nfft;
ii=find(f<=100);
Ta=1/fa;
%
Zf= fft(Z,Q);  % Transformada Rápida de Fourier do Sinal de Entrada
Zf= Zf.* conj(Zf)/Q;
Ff=  fft(F1,Q);  % Transformada Rápida de Fourier da Saída do Filtro
Digital
Ff = Ff.* conj(Ff)/Q;
Yf= fft(y3,Q); % Transformada Rápida de Fourier da Saída do Filtro Neural
Yf = Yf.* conj(Yf)/Q;
%
figure(8);
subplot(2,1,1)
h=plot(f(ii), Zf(ii),'r');
title('FFT do sinal de entrada')
xlabel('Frequencia (Hz)'),ylabel('Amplitude'),grid
subplot(2,1,2)
h=plot(f(ii), Yf(ii),'b');
title('FFT da saída do Emulador Neural')
xlabel('Frequencia (Hz)'),ylabel('Amplitude'),grid
pause(1)

% 4.3 Comparação entre o filtro digital e o emulador neural Passa-Baixas
figure(9);
ii=find(f<=20);
h=plot(f(ii), Ff(ii),'r',f(ii), Yf(ii),'b');
title('Desempenho do Emulador Neural Passa-Baixas')
xlabel('(Hz)'),ylabel('Amplitude'),grid
legend('Filtro Digital', 'Emulador Neural')
pause(1)

% 4.4 Erro entre o sinal filtrado e o emulador neural passa-baixas
erro= F1 - y3;
ii=find(t<=1);
```

```
figure(10);
h=plot(t(ii),erro(ii),'r');grid;
title('Diferença Entre o Sinal do Filtro e do Emulador Neural Passa-
Baixas','Fontsize',8);
xlabel('Tempo (s)'); ylabel('Amplitude')
```

APÊNDICE C – ROTINA DO MATLAB® QUE IMPLEMENTA O EMULADOR NEURAL PASSA-BAIXAS

```
%
%  Script File: EmuladorNeuralPA.m
%
%  Este roteiro MATLAB desempenha a função de um Filtro Passa-Altas
%  de Ordem 2, tipo Butterworth mediante a utilização de uma Rede Neural
%  e compara o resultado com a saída de um filtro digital.
%
%*************************************************************
% Autores: Hiran de Melo e Carlos Eduardo Mendes Alves Pinto
% % Departamento de Engenharia Elétrica
% e@mail: hiran@dee.ufpb.edu.br
%         a20121117@grad.dee.ufcg.edu.br
% Criado: 02/02/2005
% Atualizado: 15/06/2006
%*************************************************************
%
close all;
clear all;
clc

% ***** A. Definição dos sinais *****

% 1. Freqüência de amostragem = 512Hz
fa = 128;
t = 0:1/fa:1;
X = sin(2*pi*t*3)+sin(2*pi*t*50); % Sinal de entrada, X(t)
%X = sin(2*pi*t*3)+0.1*randn(size(t));

% ***** B. Projeto do Filtro Butterworth Passa-Altas *****

% 1. Freqüência de Nyquist
fn = fa/2;

% 2. Normalização dos valores das freqüências

% 2.1 faixa de passagem:[20Hz, fn]
fp = 20/fn;

% 2.2 faixa de bloqueio: [0, 5Hz]
fs = 5/fn;

% 3. Especificações das atenuações mínima e máxima
Rp = 3; % atenuação na banda passante: 3 dB
Rs = 20; % atenuação na banda de rejeição: 20 dB

% 4. Ordem do Filtro e freqüencia de corte normalizada
[N,Wn] = buttord(fp,fs,Rp,Rs);

% 4.1 Saída do Filtro para o sinal de entrada X(t)
[B,A] = butter(N,Wn,'high');
figure(1);
freqz(B,A,64,fa)
```

```
title('Resposta em Freqüência do Filtro Passa-Altas')
pause(1);

% 5. Função de Transferência do Filtro Passa-Altas
F = filter(B,A,X);

% 6. Resposta ao Impulso do Filtro Passa-Altas

% 6.1 Definição do Impulso unitário
X1 = zeros(1,size(X,2));
X1(1) = 1;

% 6.2 Determinação da resposta ao impulso unitário
FImp = filter(B,A,X1);
figure(2);
plot(t,FImp);
title('Resposta ao Impulso do Filtro Passa-Altas')
xlabel('Tempo(s)'),ylabel('Amplitude'),grid


% *****  C. Projeto do Emulador Neural Passa-Altas *****

% 1. Definição da Arquitetura da Rede Neural:

% Camada de entrada: 10 neurônios com 5 entradas cada
% Camada de saída: 1 neurônio com 10 entradas.
% Notação: RNMC(5,10,1)

% 2.  Construção da matriz de entrada

% 2.1 Determinação do número amostras do sinal de entrada.
Q=size(X,2);

% 2.2 Definição das dimensões da matriz de entrada
P = zeros(5,Q);

% 2.3 Composição da matriz a partir do sinal de entrada, X(t)
P(1,1:Q) = X(1,1:Q);
P(2,2:Q) = X(1,1:(Q-1));
P(3,3:Q) = X(1,1:(Q-2));

P(4,2:Q) = F(1,1:(Q-1));
P(5,3:Q) = F(1,1:(Q-2));

% 3. Criação da Rede Neural
net = newff(minmax(P), [10 1], {'tansig', 'purelin'}, 'trainlm'); %Rede
Neural Multicamadas
% net = newff(minmax(P), [5 1], {'tansig', 'purelin'}, 'trainbr');
% net = newff(minmax(P), [5 1], {'purelin', 'purelin'}, 'trainlm');

% 3.1 Inicialização da RNMC
net.layers{1}.initFcn='initwb';
net.inputWeights{1,1}.initFcn='rands';
net.layerWeights{2,1}.initFcn='rands';
net.biases{1}.initFcn='rands';
net.biases{2}.initFcn='rands';
```

```
net=init(net);

% 3.2 Simulaçao da Rede Neural sem treinamento:
y1 = sim(net,P);

% 4. Treinamento da Rede Neural
disp(['    '])
tpr= input ('Treinar Filtro Neural? : Sim(1) ou Não (0) tpr =  ');
if isempty(tpr), tpr=1; end;
%
if tpr==1,
%
disp(['     Parâmetros da Rede Neural'])
disp(['    '])
epocas= input ('Números Máximo de Treinamentos:(30000) epocas =  ');
if isempty(epocas), epocas=30000; end;
disp(['    '])
rms= input ('Mínimo Erro Médio Quadrático(1e-15) : rms =  ');
if isempty(rms), rms=1e-15; end;
disp(['    '])
%
net.trainParam.epochs=epocas;
net.trainParam.show=epocas/5;
net.trainParam.goal=rms;
net.trainParam.min_grad=rms*1e-20;

net = train(net,P,F);

else
   load Filtro01w1;load Filtro01w2;
   load Filtro01b1;load Filtro01b2;
net.IW{1,1}=w1;
net.LW{2,1}=w2;
net.b{1}=b1;
net.b{2}=b2;
end;
%
w1=net.IW{1,1};
w2=net.LW{2,1};
b1=net.b{1};
b2=net.b{2};
%
save Filtro01w1 w1; save Filtro01w2 w2;
save Filtro01b1 b1; save Filtro01b2 b2;

% 4.1 Simulação da Rede Neural com treinamento
y2 = sim(net,P);
figure(3);
plot(X,F,'o',X,y1,'*',X,y2,'+')
title('Relação do Sinal de Entrada com a Rede Neural')
legend( 'Filtro Digital','Sem Treinamento','Com Treinamento')
pause(1);

% 5. Analise de Desempenho do Treinamento da Rede Neural
y2r = y2(1:10:end);
Fr = F(1:10:end);
figure(4);
```

```
title(' Valor de Correlação entre as Saídas e os Alvos')
[m,b,r] = postreg(y2r,Fr);
pause(1);
r % Valor de correlação entre as saídas e os alvos. Quanto mais prox de 1
melhor.

% 5.1 Analise de Desempenho no Domínio do Tempo
figure(5);
plot(t,F,'r',t,y2,'b')
title('Treinamento: Desempenho do Filtro Neural')
legend('Saída do Filtro Digital', 'Saída do Filtro Neural')
pause(1);


% ***** D. Teste de Generalização *****

% 1. Novo Sinal de entrada, Z(t)
Q = 8*Q;
t = linspace(0,8,Q);

%Z = sin(2*pi*t*3);
%Z = sin(2*pi*t*3)+.25*sin(2*pi*t*30);
%Z = sin(2*pi*t*3)+.5*sin(2*pi*t*40);
%Z = sin(2*pi*t*3)+0.1*randn(size(t));
Z = 0.5*sin(2*pi*t*2)+0.5*sin(2*pi*t*4)+sin(2*pi*t*40)+sin(2*pi*t*60);

% 1.1 Saída do Filtro para o novo sinal de entrada Z(t)
F1 = filter(B,A,Z);

% 2. Composição da matriz a partir do sinal de entrada, Z(t)
P1 = zeros(5,Q);
P1(1,1:Q) = Z(1,1:Q);
P1(2,2:Q) = Z(1,1:(Q-1));
P1(3,3:Q) = Z(1,1:(Q-2));

S(1) = sim(net,P1(:,1));
P1(4,2) = S(1);
S(2) = sim(net,P1(:,2));
P1(4,3) = S(2);
P1(5,3)= S(1);
S(3) = sim(net,P1(:,3));

for jj=4:Q
P1(4,jj) = S(jj-1);
P1(5,jj) = S(jj-2);
S(jj) = sim(net,P1(:,jj));
end

% 3. Simulação da rede neural generalizada sem novo treinamento
y3 = sim(net,P1);

% 4. Análise de desempenho da generalização da rede neural
y3r = y3(1:10:end);
Fr = F1(1:10:end);
figure(6);
title(' Valor de Correlação entre as Saidas e os Alvos')
[m,b,r1] = postreg(y3r,Fr);
```

```
pause(1)
r1 % Valor de correlação entre as saidas e os alvos. Quanto mais prox de
1 melhor.

% 4.1 Análise de desempenho no domínio do tempo
figure(7);
plot(t,F1,'r',t,y3,'b')
axis([0 1 -2 2])
title('Desempenho na Generalização do Filtro Neural')
legend('Saída do Filtro Digital', 'Saída do Filtro Neural')
pause(1);

% 4.2. Análise de desempenho no domínio da freqüência
nfft = length(Z);
nfft1 = nfft/2;
nfft2 = nfft1-1;
f = fa*(0:nfft2)/nfft;
ii=find(f<=100);
Ta=1/fa;
%
Zf= fft(Z,Q);  % Transformada Rápida de Fourier do Sinal de Entrada
Zf= Zf.* conj(Zf)/Q;
Ff=  fft(F1,Q);  % Transformada Rápida de Fourier da Saída do Filtro
Digital
Ff = Ff.* conj(Ff)/Q;
Yf= fft(y3,Q); % Transformada Rápida de Fourier da Saída do Filtro Neural
Yf = Yf.* conj(Yf)/Q;
%
figure(8);
subplot(2,1,1)
h=plot(f(ii), Zf(ii),'r');
title('FFT do sinal de entrada')
xlabel('Frequencia (Hz)'),ylabel('Amplitude'),grid
subplot(2,1,2)
h=plot(f(ii), Yf(ii),'b');
title('FFT da saída do Emulador Neural')
xlabel('Frequencia (Hz)'),ylabel('Amplitude'),grid
pause(1)

% 4.3 Comparação entre o filtro digital e o emulador neural passa-altas
figure(9);
ii=find(f<=70);
h=plot(f(ii), Ff(ii),'r',f(ii), Yf(ii),'b');
title('Desempenho do Emulador Neural Passa-Altas')
xlabel('(Hz)'),ylabel('Amplitude'),grid
legend('Filtro Digital', 'Emulador Neural')
pause(1)

% 4.3 Erro entre o sinal filtrado e o emulador neural passa-altas
erro= F1 - y3;
ii=find(t<=1);
figure(10);
h=plot(t(ii),erro(ii),'r');grid;
title('Diferença Entre o Sinal do Filtro e do Emulador Neural Passa-
Altas','Fontsize',8);
xlabel('Tempo (s)'); ylabel('Amplitude')
```