UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA E INFORMÁTICA INDUSTRIAL

DANIEL ROSSATO DE OLIVEIRA

PROJETO DE FILTROS DIGITAIS IIR COM TÉCNICAS DE COMPUTAÇÃO EVOLUCIONÁRIA

DISSERTAÇÃO

CURITIBA

2011

DANIEL ROSSATO DE OLIVEIRA

PROJETO DE FILTROS DIGITAIS IIR COM TÉCNICAS DE COMPUTAÇÃO EVOLUCIONÁRIA

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre em Ciências, do Programa de Pós-graduação em Engenharia Elétrica e Informática Industrial, Universidade Tecnológica Federal do Paraná. Área de Concentração: Engenharia de Computação.

Orientador: Prof. Dr. Heitor Silvério Lopes

CURITIBA

2011

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
Câmpus Curitiba

Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial

Título da Dissertação Nº 582

"Projeto de Filtros Digitais do tipo IIR utilizando Computação Evolucionária"

por

Daniel Rossato de Oliveira

Esta dissertação foi apresentada como requisito parcial à obtenção do grau de MESTRE EM CIÊNCIAS – Área de Concentração: Engenharia da Computação, pelo Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial – CPGEI – da Universidade Tecnológica Federal do Paraná – UTFPR – Câmpus Curitiba, às 8h do dia 06 de dezembro de 2011. O trabalho foi aprovado pela Banca Examinadora, composta pelos professores:

Prof. Heitor Silvério Lopes, Dr.
(Presidente – UTFPR)

Prof. Mauricio Kugler, Dr.
(NIT – Japão)

Prof. Ricardo Hiroshi Caldeira Takahashi, Dr.
(UFMG)

Prof. Fábio Kurt Schneider , Dr.
(UTFPR)

Visto da coordenação:

Prof. Fábio Kurt Schneider, Dr.
(Coordenador do CPGEI)

A Folha de Aprovação assinada encontra-se na Coordenação do Programa.

AGRADECIMENTOS

À minha namorada Ana Paula, que me apoiou nos momentos mais difíceis e brigou comigo quando as broncas eram necessárias.

Aos meus gatos Chorão, Atena e Pantera, que me alegraram durante a tristeza da escrita solitária.

Ao meu orientador Prof. Heitor S. Lopes, que me guiou, ajudou, e cobrou na medida necessária para tirar o melhor da minha capacidade.

Aos meus vários colegas de laboratório durante estes 2 anos, que me ajudaram tanto no trabalho quanto na descontração.

Ao Prof. Richard Demo Souza, que sempre me atendeu e ajudou cordialmente quando pedi, além de fornecer material bibliográfico de suma importância.

Aos meus alunos, pelo que aprendi com cada um, pelo que cresci ao tentar passar meu conhecimento para todos, e pela mais que prazerosa convivência dentro e fora de sala de aula.

Ao Deus Metal, pela inspiracão e força a mim cedidos através de sua gloriosa música. Que nossas guitarras toquem mais alto, nossa bateria seja mais rápida, nosso baixo tenha mais peso, e nossa voz sempre evoque seu épico poder.

’Cause only one thing really sets me free:
Heavy Metal, loud as it can be!

(Joe DeMaio)

RESUMO

OLIVEIRA, Daniel R.. Projeto de Filtros Digitais IIR com Técnicas de Computação Evolucionária. 92 f. Dissertação – Programa de Pós-graduação em Engenharia Elétrica e Informática Industrial, Universidade Tecnológica Federal do Paraná. Curitiba, 2011.

O projeto de filtros digitais do tipo IIR é um problema clássico da engenharia. Filtros digitais possuem diversas aplicações, e muitas variantes de comportamento. Existem vários métodos especialistas consagrados na literatura, cada um com suas peculiaridades e adequações a diferentes casos. Porém, a maior parte destes métodos não é flexível, impedindo a especificação de todos os parâmetros importantes de um filtro, pois alguns desses são um sub-produto do processo de cálculo. Por isso, este problema também é atacado na literatura através de técnicas de computação evolucionária. Diversos estudos foram encontrados com esta abordagem, mas em quase todos, o enfoque era dado no algoritmo evolucionário em vez de no filtro, utilizando este apenas como uma função de *benchmark*. Além disso, a estabilidade do filtro - condição imprescindível para a utilização do mesmo - é desprezada em quase todos os trabalhos. Portanto, este trabalho propõe uma função de *fitness* e uma nova codificação para este problema, de forma a possibilitar a obtenção de bons filtros, dentro das especificações, com algoritmos de Computação Evolucionária na forma canônica, isto é, sem modificações estruturais. A função de *fitness* proposta busca corrigir distorções causadas pela função tradicional, que não leva em conta a obediência às especificações do filtro. A codificação mapeia o espaço de busca apenas para as soluções estáveis, sem excluir nenhuma solução válida nesta transformação. Além disso, um pós-processamento permite equalizar a resposta em fase do filtro, isto é, tornar o atraso de fase na banda de passagem linear, condição necessária para a utilização em diversos sistemas, especialmente os de telecomunicações. O desempenho das modificações é comparado com as abordagens clássicas utilizadas na literatura, e o conjunto escolhido como o mais eficiente é utilizado para comparar os dois algoritmos mais utilizados em Computação Evolucionária, o PSO e o AG. Após esse passo, experimentos extensivos de ajuste de parâmetros foram realizados, para que a versão final fosse comparada com o método especialista mais poderoso, que é o cálculo de filtros elípticos. Os resultados mostraram que o conjunto de modificações proposto fez com que excelentes filtros fossem obtidos, com uma taxa de obediência às especificações muito superior à obtida sem o mesmo. Comparando com o método especialista, o desempenho foi semelhante, com pontos a favor e contra cada um, mostrando que o projeto de filtros IIR através de Computação Evolucionária pode ser utilizado em sistemas reais. Em trabalhos futuros poderão ser estudadas novas modificações na funçao de *fitness*, além do desempenho obtido com outros algoritmos evolucionários. A utilização em sistemas *online* é uma aplicação promissora, e o comportamento deste método com especificações não-estacionárias, oriundas de informações de estimação de canal também deve ser investigado.

Palavras-chave: Filtros Digitais, Computação Evolucionária.

ABSTRACT

OLIVEIRA, Daniel R.. IIR Digital Filter Design using Evolutionary Computing Techniques. 92 f. Dissertação – Programa de Pós-graduação em Engenharia Elétrica e Informática Industrial, Universidade Tecnológica Federal do Paraná. Curitiba, 2011.

IIR digital filter design is classic problem in Electrical Engineering. Digital filter have many applications, and dozens of different behaviors. There are many specialist methods in literature, each with specific differences and characteristics. But most of these methods are not flexible, preventing one from fixing all parameters, because some of them are a sub-product of the calculation process. Therefore, this problem is also solved in literature with Evolutionary Computing techniques. Several works were found with this approach, but in almost all, the focus was in the evolutionary algorithm rather than on the filter problem, using this only as a benchmark function. Moreover, the filter stability - prerequisite for the filter application - is neglected in almost all of them. So, this work proposes a new fitness function, as well as a new codification, in order to obtain good filters, obedient to the specifications, with Evolutionary Computing algorithms in the canonic form, this is, without structural modifications. The proposed fitness function tries to correct distortions caused by the traditional one, which does not takes into account the filter gain specifications. The new coding maps the search space only to the stable solutions, not excluding any valid solution in the transformation. Moreover, a post-processing allows the filter phase equalization, this is, to make the pass band phase delay linear. This property is necessary in many types of applications, specially the telecommunication ones. The modifications performance is compared to the classic approaches found in the literature, and the most efficient set is use to compare the two most used Evolutionary Computing algorithms, GA and PSO. After this, extensive parameter tuning experiments were made, producing a final version of the method to compare with the specialist one. The chosen specialist method was the Elliptic filter, because the filter response has similarities with the one found by the proposed algorithm, and also because it is the most powerful one. Results showed excellent filters found with the proposed modifications, with a specification obedience rate well above the ones obtained with the classical approaches. Comparing with the specialist method, a similar performance was achieved, showing that IIR filter design with Evolutionary Computing can be used in real systems. Future work will address new modifications in the fitness function, and the performance of different EC algorithms in this problem. The application in online systems is promising, so the behavior of this approach with non-stationary specifications coming from channel estimation techniques should also be investigated.

Keywords: Digital Filters, Evolutionary Computing.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Sinais são representações de quantidades físicas variantes no tempo ou no espaço. Processamento de sinais é a análise e manipulação destes sinais para a extração de informações ou para torná-los aptos a alguma utilização específica que não seria possível com os sinais em seu estado original. Estes sinais podem ser originários de diversas origens, como som, imagem, dados médicos como um sinal de ECG ou EEG, a leitura de um sensor industrial que informe temperatura ou velocidade de um motor, transmissões de rádio, etc.

Dentro da área de processamento de sinais, existe uma classe de sistemas chamada de filtros. Filtros são estruturas construídas de forma a eliminar ou acentuar determinados aspectos de um sinal. Por exemplo, um aparelho de som que reforce as componentes graves de uma música possui um filtro que tenha esse comportamento. É comum a necessidade de se eliminar o ruído causado pela rede elétrica em sinais médicos, sendo necessário também um filtro para isso. Receptores de rádio (e qualquer outro dispositivo que receba informação por radiofreqüência) usam filtros para selecionar apenas um canal dentre os vários sendo recebidos. Em processamento de imagens há filtros para, entre outras aplicações, atenuar as distorções causadas pela exibição em pixels, chamados de filtros *anti-aliasing*. Estes mesmos filtros são utilizados em quase todas as aplicações para diminuir distorções causadas pela sub-amostragem.

Existem filtros de diversos tipos, de acordo com sua implementação. Podem ser ativos ou passivos, dependendo da necessidade de alimentação externa; digitais, se a representação do sinal for discreta, ou analógicos, se ela for contínua, isto é, poder expressar infinitos níveis diferentes. Esta propriedade do sinal é extrapolada para o filtro, sendo o mesmo também "discreto"ou "contínuo"de acordo com o tipo do sinal por ele processado. Além disso, os filtros são classificados entre FIR e IIR de acordo com a duração do sinal na sua saída ao ser submetido a uma entrada impulsiva, sendo finito e infinito respectivamente. Por fim, a funcionalidade ou comportamento do filtro também é uma das formas de classificá-lo. Um filtro que atenua as componentes de alta freqüência de um sinal e ressalta as baixas é chamado de Passa-Baixas, enquanto que um com comportamento oposto é chamado de Passa-Altas.

O uso de filtros digitais é muito difundido na engenharia elétrica. Ao contrário dos filtros analógicos, o aumento da ordem do filtro, que a grosso modo determina a eficiência do mesmo, não implica necessariamente em aumento de tamanho físico ou custo de componentes, apenas em custo computacional. Além disso, a resposta do filtro não varia conforme a precisão de componentes discretos como resistores e capacitores, que estão sujeitos a flutuações de temperatura, umidade, e até um eventual desgaste. Outra vantagem dos filtros digitais é a flexibilidade, pois pode-se alterar completamente as características do mesmo apenas com modificações simples no software, enquanto que em um filtro analógico seria necessário trocar vários componentes fisicamente.

Filtros digitais podem ser completamente caracterizados por uma série de coeficientes, que também são os elementos necessários para a implementação do filtro. Pela importância do tema, o cálculo destes coeficientes é vastamente estudado na literatura, com vários métodos consagrados apresentando, cada um, vantagens e desvantagens características. É uma atribuição do projetista a escolha do algoritmo mais adequado para cada aplicação, de acordo com os requisitos do sistema.

Porém, a maioria destes métodos clássicos para o cálculo de filtros não é flexível. Alguns não permitem a especificação de todos os parâmetros ao mesmo tempo, tendo um ou mais destes como produto do cálculo, juntamente com os coeficientes. Alguns dos métodos exigem cálculos complexos, que dificultam sua implementação em sistemas que recalculem o filtro em tempo de execução.

Outro problema é que nenhum dos métodos tradicionais produz um filtro IIR cuja resposta tenha atraso de fase constante. Essa é uma característica muito importante, pois a distorção de fase é um problema grave em diversos sistemas. Existem procedimentos chamados de equalização de fase que corrigem esta resposta através de um segundo filtro. Porém, estes métodos são computacionalmente caros, e necessitam que o filtro esteja pronto para que o procedimento seja realizado, ou seja, são executados apenas depois que o filtro com a resposta em magnitude desejada já esteja calculado.

Os algoritmos de Computação Evolucionária são tradicionalmente usados para otimização de problemas numéricos. Para sua utilização, é necessário que qualquer dada solução do problema seja passível de ser representada como um vetor numérico, e que exista uma forma de mensurar objetivamente a qualidade de cada solução proposta, sendo esta medida chamada de *fitness*. O problema do cálculo de filtros obedece estes critérios, tanto que é vasta na literatura a utilização dos mais diversos algoritmos de CE para esta aplicação. A grande vantagem desta abordagem é a sua flexibilidade. É possível alterar a forma de mensuração da qualidade

do filtro, e assim conduzir a busca para soluções mais específicas. Também se pode buscar a otimização simultânea de diversos parâmetros independentes, enquanto os métodos tradicionais de cálculo em geral permitam a fixação de apenas alguns, tendo os outros como sub-produto do cálculo.

1.1 OBJETIVOS

Portanto, o objetivo geral deste trabalho é apresentar um método de otimização evolutiva para o cálculo de filtros digitais do tipo IIR. Este tipo de filtro foi escolhido por apresentar um *roll-off* (taxa de queda no ganho entre a banda de passagem e a banda de rejeição) mais acentuado do que o dos filtros FIR para um mesmo número de coeficientes. O método produz um filtro já equalizado em fase, ou seja, que apresente um atraso de fase constante na banda de passagem.

Os objetivos específicos são:

- Propor uma codificação mais adequada ao problema do cálculo de filtros, e comparar seu desempenho ao alcançado pela codificação clássica.
- Avaliar o método de CE mais adequado para o problema
- Comparar os resultados da abordagem proposta com os métodos clássicos

1.2 ORGANIZAÇÃO DO TRABALHO

Este trabalho está dividido da seguinte maneira: na Seção 2, serão apresentados os fundamentos de análise de sinais e sistemas necessários para o desenvolvimento e entendimento do trabalho, assim como os métodos especialistas mais utilizados para cálculo de filtros. Os algoritmos evolucionários mais comuns também serão mostrados, e trabalhos correlatos encontrados na literatura serão discutidos. Na Seção 3, será apresentada a função de *fitness* utilizada na otimização de filtros digitais, assim como as modificações propostas. Em seguida, será justificada a escolha do método especialista para comparação de resultados, com uma análise mais profunda do mesmo. A codificação tradicional utilizada em otimização será discutida juntamente com as alterações propostas. E por último, serão apresentados os experimentos que serão realizados. Na Seção 4 serão mostrados os resultados obtidos nos experimentos, e aqueles serão discutidos na Seção 5.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 SISTEMAS LINEARES INVARIANTES NO TEMPO DISCRETO

O processamento de sinais é a área da Engenharia Elétrica que estuda as propriedades dos sinais, isto é, da representação numérica de informações. Dentro desta grande área, existe o processamento digital de sinais (PDS), que diz respeito aos sinais correspondentes a variáveis discretas e os sistemas utilizados para processá-los. Esta é uma disciplina de grande penetração em nosso cotidiano, estando presente em transmissões de vídeo digital, telefones celulares, satélites e receptores GPS, reprodutores de música MP3, câmeras fotográficas e de vídeo, entre outros. Enquanto o processamento analógico trabalha com sinais que variam continuamente no tempo, no processamento digital tem-se sequências de números, geralmente chamadas de amostras.

No processamento analógico, há sistemas que alteram um dado sinal de forma também contínua. Ou seja, dado um sinal de entrada $x(t)$ contínuo, após passar pelo sistema de processamento, tem-se outro sinal $y(t)$, também contínuo. Portanto, esta modificação pode ser descrita através de equações diferenciais, que caracterizam e descrevem o sistema.

Já no processamento digital de sinais, não é possível a utilização de equações diferenciais, sendo utilizadas as equações de diferenças (DINIZ et al., 2004). De forma análoga, estas equações descrevem um sistema e as modificações que ele exerce em um sinal de entrada $x(n)$ para transformá-lo em outro sinal de saída $y(n)$. Este sistema é implementado através de um *hardware* digital, que vai receber esta sequência através de alguma interface com o mundo analógico, ou de outro circuito digital.

A maior vantagem do PDS é a facilidade com que se pode realizar cálculos complexos que seriam extremamente complicados de se implementar utilizando circuitos analógicos (OPPENHEIM; SCHAFER, 1975). Operações como radiciação, exponenciação, logarítmos e funções trigonométricas são muito difíceis de serem executadas com circuitos analógicos, enquanto um processador digital consegue realizá-las de forma precisa e confiável. Para tanto, a menos que o sinal original já seja discreto, é necessário que haja um meio de converter o sinal

contínuo $x(t)$ em um sinal discreto $x(n)$, e também o sinal de saída discreto $y(n)$ em $y(t)$.

Este trabalho é focado em filtros digitais, que são uma classe de sistemas digitais costumeiramente lineares e invariantes no tempo discreto. Os tipos de filtro aqui tratados também são causais, outra propriedade a ser discutida. Os filtros estudados neste trabalho são sistemas LSI (*Linear Shift Invariant*), cujas propriedades são discutidas na Seção 2.2.

2.2 PROPRIEDADES DE SISTEMAS LSI

Nesta seção, são estudadas as propriedades mais importantes dos sistemas LSI, assim como suas implicações.

2.2.1 Linearidade de Sistemas Digitais

Um sistema no tempo discreto $H\{x(n)\}$ é linear se e somente se

$$H\{a \cdot x(n)\} = a \cdot H\{x(n)\} \tag{1}$$

e

$$H\{x_1(n) + x_2(n)\} = H\{x_1(n)\} + H\{x_2(n)\} \tag{2}$$

para qualquer constante a e para quaisquer sequências $x(n)$, $x_1(n)$ e $x_2(n)$.

Ou seja, para que um sistema no tempo discreto seja considerado linear, o resultado do processamento de um sinal qualquer multiplicado por uma constante deve ser igual ao resultado do processamento do sinal original, multiplicado pela mesma constante. Além disto, a saída resultante de um sinal composto pela soma de dois sinais diferentes deve ser igual à soma das saídas que o sistema produziria caso cada sinal fosse processado de forma independente (DINIZ et al., 2004).

2.2.2 Invariância no tempo discreto

Esta propriedade diz respeito à mudança ou não da resposta proveniente de um sistema a partir de um sinal, caso o instante em que este começa a ser recebido seja diferente. Ou seja, um sistema é invariante no tempo discreto se e somente se

$$H\{x(n-n_0)\} = y(n-n_0) \tag{3}$$

onde $y(n) = H\{x(n)\}$. Ou seja, caso o sinal seja deslocado, a resposta do sistema deverá ser idêntica e deslocada exatamente na mesma medida (OPPENHEIM; SCHAFER, 1989). Esta propriedade também é chamada de invariância ao deslocamento, já que não necessariamente o sinal sendo analisado foi amostrado no tempo, podendo ser amostrado no espaço, na temperatura, etc. Portanto, um sistema discreto, linear e invariante no deslocamento é chamado de LSI, do inglês *Linear Shift-Invariant*, em analogia aos sistemas contínuos LTI, ou *Linear Time-Invariant*.

2.2.3 Causalidade

Um sistema é dito causal quando a saída em um instante n_0 não depende de nenhuma amostra do sinal de entrada posterior a n_0. Em outras palavras, a saída não depende de amostras futuras (DINIZ et al., 2004). Ou seja, um sistema é causal se e somente se

$$H\{x_1(n)\} = H\{x_2(n)\}, \text{ para } n < n_0 \tag{4}$$

quando $x_1(n) = x_2(n)$ para $n < n_0$. Esta propriedade é importante porque, claramente, sistemas não-causais não podem ser implementados em tempo real. Caso o processamento seja *offline*, o sistema tem acesso a toda a sequência e, portanto pode já no instante inicial utilizar o valor das últimas amostras para calcular a saída (OPPENHEIM; SCHAFER, 1975). Porém, isto não é possível em um sistema que só vai receber o valor da amostra seguinte após o instante em que deve devolver a saída. Como uma grande parte das aplicações de PDS é *online*, ou seja, processa ao mesmo tempo em que recebe as amostras, a causalidade é condição *sine qua non* para que a implementação seja possível.

2.2.4 Caracterização pela resposta ao impulso

É possível demonstrar que um sistema LSI $H\{x(n)\}$ é completamente caracterizado pela sua resposta ao impulso unitário $h(n)$ (OPPENHEIM; SCHAFER, 1989). Esta observação vem da equação

$$y(n) = \sum_{k=-\infty}^{\infty} x(k) \cdot h(n-k), \tag{5}$$

que também significa que a saída de um sistema LSI é a convolução da sequência de amostras de entrada $x(n)$ com a sequência que contém a resposta ao impulso $h(n)$ do sistema $H\{x(n)\}$. Também é possível demonstrar que a resposta ao impulso de um sistema composto por dois outros sistemas ligados em série, ou seja, a saída do primeiro é alimentada na entrada do segundo, é igual à convolução das respostas ao impulso destes dois sistemas. Matematicamente,

$$y_3(n) = y_1(n) * y_2(n), \tag{6}$$

onde $y_3(n)$ é a resposta ao impulso do sistema $H_3\{x(n)\}$, composto pelos sistemas $H_1\{x(n)\}$ e $H_2\{x(n)\}$ ligados em cascata (série), ou seja, $H_3\{x(n)\} = H_2\{H_1\{x(n)\}\}$.

Estas propriedades são extremamente importantes no estudo de processamento digital de sinais. Elas facilitam tanto a análise quanto a síntese dos mesmos. Várias técnicas de projeto de filtros, por exemplo, utilizam a propriedade do cascateamento de sistemas para dividir o cálculo de um filtro complexo em vários filtros mais simples.

2.2.5 Coeficientes de um sistema LSI

Um sistema contínuo pode ser representado por uma equação diferencial, enquanto um sistema discreto é representado por uma equação de diferenças (DINIZ et al., 2004). Esta equação assume genericamente a forma mostrada na Equação 7.

$$\sum_{i=0}^{N} a_i y(n-i) - \sum_{l=0}^{M} b_i x(n-l) = 0 \tag{7}$$

Como se deseja encontrar a equação que fornece a saída do sistema a cada instante, deve-se isolar esta saída, que é $y(n)$. Obtem-se então a Equação 8 (a_0 é considerado igual a 1).

$$y(n) = \frac{\sum_{l=0}^{M} b_i x(n-l)}{\sum_{i=1}^{N} a_i y(n-i)} \tag{8}$$

Desta forma, o sistema LSI pode ser descrito pelos vetores de coeficientes a_k e b_k, que multiplicam as saídas e entradas anteriores, respectivamente. Estes coeficientes são utilizados para a construção dos sistemas na forma canônica, e o projeto destes sistemas consiste basicamente no cálculo dos coeficientes (DINIZ et al., 2004).

2.2.6 Estabilidade

Para um sistema ser considerado estável, ele deve obedecer às condições de BIBO (*Bounded Input Bounded Output*), o que significa que para qualquer sinal de entrada para o qual exista uma constante finita maior que o módulo de todas as amostras, o sinal de saída também obedecerá a este critério (DINIZ et al., 2004). Esta condição é satisfeita quando a resposta ao impulso for absolutamente somável. Matematicamente, para um sistema $H\{x(n)\}$ cuja resposta ao impulso seja $h(n)$, é necessário que:

$$\sum_{n=-\infty}^{\infty} |h(n)| < \infty \tag{9}$$

Como a Inequação 9 possui infinitas soluções, não há como utilizá-la neste formato para verificar a estabilidade de um sistema $H\{x(n)\}$ qualquer. Para isto, o mais usual é efetuar a transformada Z da função de transferência do sistema, encontrando $H(z)$. A partir dela, se encontram os pólos do sistema, isto é, os valores de z que tornam nulo o denominador da função. A condição necessária para que este sistema seja estável é que todos os pólos estejam dentro do círculo de raio unitário, isto é, tenham módulo menor que 1 (OPPENHEIM; SCHAFER, 1989).

2.3 FILTROS DIGITAIS

Filtros digitais são sistemas que efetuam uma série de operações matemáticas em um sinal digital discreto, de modo a reduzir ou ampliar características específicas deste sinal (HAMMING, 1997). Além de sua implementação, matematicamente a maior diferença entre esta classe de filtros e os filtros analógicos é a natureza do sinal, pois os analógicos processam sinais contínuos. Através de circuitos como conversores analógico-digitais e digitais-analógicos, condicionadores de sinal e outras interfaces, é possível filtrar um sinal contínuo através de um filtro digital, efetuando a transformação do sinal em digital e discreto, e depois convertendo o sinal de saída novamente para contínuo (DINIZ et al., 2004).

Estes filtros tem inúmeras vantagens, algumas delas já citadas no Capítulo 1, como flexibilidade, invariabilidade da resposta, etc. Porém, há algumas desvantagens, como a necessidade de filtros anti-recobrimento (ou *anti-aliasing*), filtros de reconstrução, apresentam ruído de quantização, entre outros.

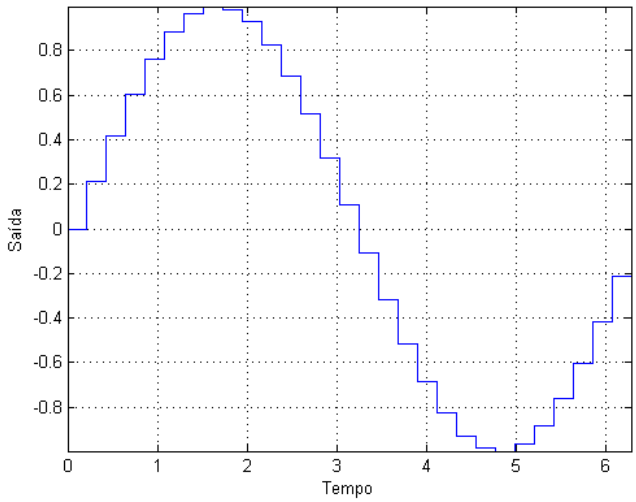
Filtros anti-recobrimento são utilizados antes do processo de amostragem do sinal contínuo. Sua função é eliminar ou atenuar ao máximo as componentes de frequências acima do limite de Nyquist, isto é, metade da frequência de amostragem (OPPENHEIM; SCHAFER,

1989). Por exemplo, se a amostragem ocorre a 10k amostras por segundo, a frequência de amostragem é de 10kHz, e o limite de Nyquist é 5kHz. Isto implica na necessidade de um filtro *anti-aliasing* que elimine as componentes acima de 5kHz do sinal contínuo. Obviamente, como este filtro se localiza antes da amostragem, ele precisa ser um filtro analógico. A natureza da necessidade desde filtro não faz parte do escopo deste trabalho, mas é uma demonstração clássica da área de Sinais e Sistemas e fácil de ser encontrada.

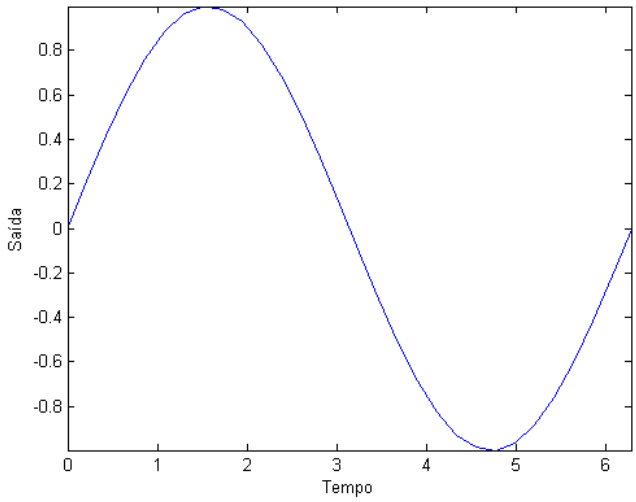
De forma análoga ao filtro *anti-aliasing*, o filtro de reconstrução serve para eliminar as componentes de frequência introduzidas no sinal pelo processo, após a conversão de digital para analógico. Este filtro é o último passo de todo o processo de filtragem digital, e como ocorre após a reconversão para sinal contínuo, também deve ser analógico. A Figura 1a mostra um sinal após esta última conversão. É possível perceber pequenos "degraus"no sinal, resultantes do chaveamento do valor de saída a cada amostra. É importante ressaltar que este sinal já é um sinal contínuo, apesar destas mudanças abruptas que se assemelham com descontinuidades. O filtro de reconstrução elimina estas mudanças abruptas, deixando o sinal final como na Figura 1b.

O ruído de quantização é originado pela impossibilidade de se representar com precisão um valor real dentro de um sistema digital (HAMMING, 1997). Um sistema que armazene os valores em variáveis de 8 bits, por exemplo, pode representar 2^8 ou 256 valores diferentes. Porém, entre quaisquer dois números, existem outros infinitos valores. Se, por exemplo, o valor máximo neste sistema representar um valor real equivalente a 5 e o mínimo for zero, o sistema pode representar 256 números diferentes entre 0 e 5, dentre os infinitos valores possíveis. Isto faz com que o processo de amostragem não produza um sinal digital realmente equivalente ao sinal contínuo, mas uma aproximação. Se no exemplo anterior o sinal amostrado em um instante k qualquer tiver o valor $4,2$, o valor armazenado seria aproximadamente $4,199$. Existem casos em que este ruído é muito prejudicial, como em aplicações de áudio de alta fidelidade. Como quase todo ruído, não há como eliminá-lo, apenas reduzí-lo. Esta redução é alcançada aumentando-se o número de bits utilizados no armazenamento de cada valor. Porém, este aumento implica em aumento do esforço computacional, que implica em aumento de consumo, preço, necessidade de processadores e conversores mais complexos e caros, etc.

Um filtro pode ser classificado em dois tipos diferentes: recursivo e não-recursivo (DINIZ et al., 2004). Um filtro recursivo recebe este nome pois o valor de cada amostra de saída depende dos valores de amostras anteriores, isto é, existe uma realimentação do sinal. Como consequência, a resposta ao impulso deste filtro é infinita, ou seja, após a excitação do filtro por um sinal impulsivo, a resposta nunca cessa mesmo que o sinal de entrada cesse. Um filtro não-recursivo, por não ter esta realimentação, possui a resposta ao impulso finita. Esta



(a) Sinal fornecido por um conversor DAC



(b) Sinal após o filtro de reconstrução

Figura 1: Sinais de saída de um sistema digital

ausência de realimentação se traduz como:

$$a_k = 0, \forall k \in N \tag{10}$$

ou seja, os coeficientes a_k referentes à Equação 8 são nulos. Afinal, estes são os coeficientes relativos às amostras de saída que, como foi dito, não influenciam nas novas amostras.

É mais usual referir-se a estes filtros como FIR (*Finite Impulse Response*) e IIR (*Infinite Impulse Response*), que em geral são respectivamente não-recursivos e recursivos. Estas siglas serão utilizadas daqui em diante neste trabalho.

2.3.1 Filtros FIR

Como citado na Seção 2.3, os filtros FIR são filtros digitais do tipo não-recursivo, isto é, não possuem realimentação. Com isto, são representados unicamente pelo vetor de coeficientes b_k. Dentre as vantagens desta classe de filtros, pode-se destacar a facilidade de implementação, estabilidade inerente, facilidade em garantir resposta em fase linear, e a ausência de erros cumulativos que seriam gerados pela realimentação, se esta existisse (HAMMING, 1997).

Por não precisarem armazenar os valores de saída nem processar os valores anteriores a cada iteração, a implementação dos filtros do tipo FIR é mais simples do que a dos filtros IIR. Porém, por necessitarem de mais coeficientes que um filtro IIR de desempenho equivalente, o esforço computacional exigido em geral é maior (DINIZ et al., 2004).

É possível demonstrar que todos os pólos de um filtro FIR estão localizados na origem do plano Z e, portanto, estes filtros são estáveis (DINIZ et al., 2004). Como sabe-se que $h(n)$ tem duração finita, a Equação 9 será sempre satisfeita se todos os coeficientes forem finitos, o que é necessário para a factibilidade do filtro (OPPENHEIM; SCHAFER, 1989).

A linearidade em fase da resposta de um sistema diz respeito ao atraso que um filtro provoca nas componentes de cada frequência do sinal. Não há como eliminar este atraso, ou o filtro não seria causal (OPPENHEIM; SCHAFER, 1975). A minimização do mesmo pode ser ou não desejada, mas esta necessidade não faz parte do escopo deste trabalho. Porém, uma condição necessária para boa parte das aplicações de filtros digitais é a constância do atraso de grupo, o que necessariamente implica que o atraso de fase gerado pelo sistema seja uma função linear da frequência. Isto é necessário para que não haja distorção de fase, um fenômeno prejudicial para diversas aplicações. Filtros FIR alcançam esta condição desde que o vetor de coeficientes b_k, que caracteriza o filtro, seja simétrico ou anti-simétrico (HAMMING, 1997).

Operações efetuadas por um sistema numérico digital geram um certo erro na resposta, diferente conforme o tipo de operação, a largura em bits dos registradores utilizados na operação, etc. Caso múltiplas operações sejam realizadas sobre um certo número, o erro introduzido a cada operação vai se acumulando. Em sistemas FIR, isto não gera muitos problemas, pois a falta de realimentação impede que os resultados de cálculos anteriores sejam utilizados em novas contas.

2.3.2 Filtros IIR

Os filtros IIR são filtros recursivos, isto é, a saída do sistema em um instante k depende não só das entradas anteriores, mas das saídas também. A principal vantagem desta classe

de filtros é que conseguem um fator de *roll-off* (a velocidade da transição entre as bandas de passagem e rejeição) muito maior do que o obtido em filtros FIR com o mesmo número de coeficientes (HAMMING, 1997). Porém, esta característica tem um custo: a resposta em fase do filtro muito raramente é linear, e o processo de cálculo é mais complexo devido à necessidade de se garantir a sua estabilidade.

Como a linearidade em fase é um requisito importante para muitas aplicações, é comum a utilização de um filtro equalizador (NOCETI FILHO, 1998). Este consiste de um segundo filtro ligado em cascata com o filtro principal, com ganho unitário para todas as frequências e resposta em fase que, casada com a resposta do primeiro filtro, resulte em fase linear.

Os principais métodos para o cálculo de filtros IIR são: Butterworth, Chebyshev tipo I e tipo II, e elíptico. Cada um deles tem características e vantagens próprias, que devem ser levadas em consideração pelo projetista de acordo com as especificações da aplicação em que o filtro será utilizado.

A aproximação de Butterworth é uma das mais utilizadas no projeto de filtros, tanto digitais quanto analógicos. Sua principal característica é a monotonicidade do ganho tanto na banda de passagem quanto na banda de rejeição. Porém, dentre os métodos citados anteriormente, este é o que possui o pior *roll-off*. A única maneira de aumentar este fator é aumentando o número de pólos do filtro, o que implica em mais coeficientes. Um exemplo de resposta de um filtro calculado através deste método é dado na Figura 2.

As aproximações de Chebyshev se diferem da de Butterworth por possuírem ganho monotônico em apenas uma das faixas. O tipo I possui a banda de rejeição monotônica, enquanto o tipo II possui a banda de passagem monotônica. Esta perda de monotonicidade implica em ondulações no ganho, mais usualmente chamadas de *ripple*, como mostrado nas Figuras 3 e 4. A quantidade de *ripple* pode ser diminuída em troca de aumento no número de coeficientes, ou piorada taxa de *roll-off*. É possível demonstrar que quando o *ripple* tende a zero, o filtro de Chebyshev se transforma em um filtro de Butterworth. Esta capacidade de se manipular *ripple* e *roll-off* sem alterar o número de coeficientes do filtro torna a aproximação de Chebyshev muito mais flexível do que a de Butterworth. Porém, não é possível especificar todos os parâmetros ao mesmo tempo. O projetista deve escolher entre fixar o *ripple* ou fixar o número de coeficientes.

Os filtros elípticos são os que possuem a melhor taxa de *roll-off*, obtendo as mais rápidas transições entre as bandas de passagem e rejeição. O custo disto é o *ripple* em toda a resposta do filtro, ou seja, tanto na banda de rejeição quanto na banda de passagem. Porém, não é possível especificar ao mesmo tempo a ordem do filtro e as frequências que compõem cada

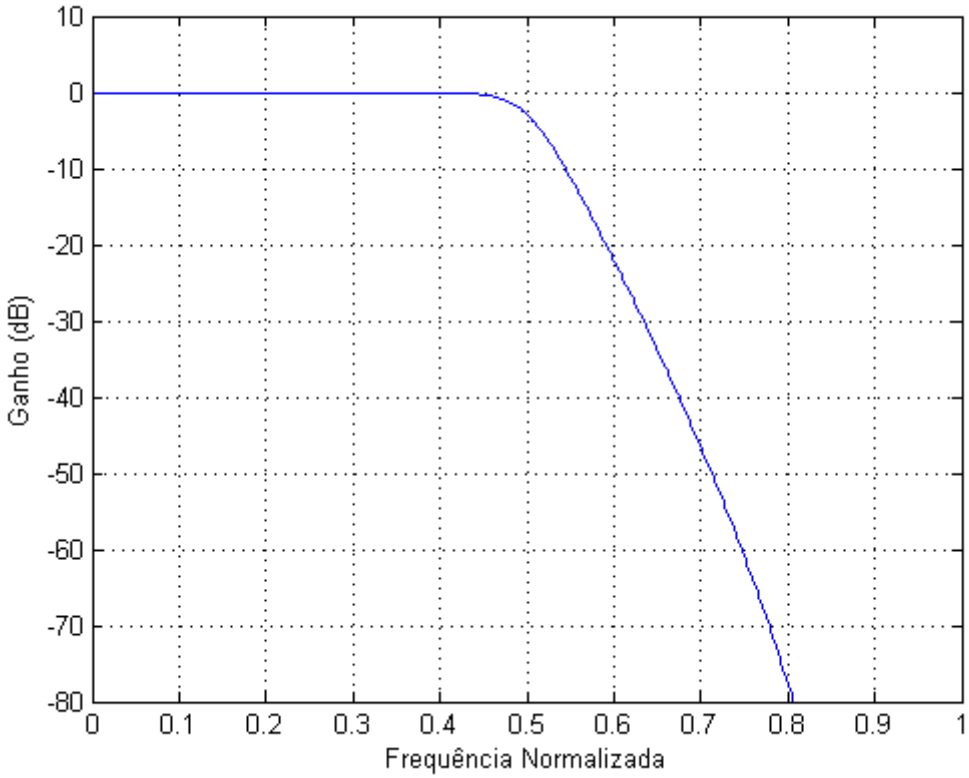


Figura 2: Resposta de um filtro de Butterworth de oitava ordem

Fonte: Autoria própria.

banda. Além disto, este tipo de filtro apresenta resposta em fase fortemente não-linear, o que dificulta o processo posterior de equalização. Um exemplo de filtro elíptico é dado na Figura 5, que mostra como o *roll-off* é mais acentuado em comparação com o dos filtros anteriores, mostrados nas Figuras 2, 3 e 4.

Para se alcançar a monotonicidade, é necessário obedecer algumas restrições matemáticas em relação aos coeficientes do filtro. Caso a abordagem utilizada para o cálculo destes coeficientes seja através de otimização numérica, estas restrições não são práticas de serem aplicadas. Além disto, é possível demonstrar que as aproximações de Butterworth e Chebyshev garantem a melhor resposta ótima dentro de suas restrições (DINIZ et al., 2004).

2.3.3 Equalização em Fase

As aproximações mais utilizadas para o projeto de filtros IIR produzem filtros cuja resposta em fase é não-linear, como citado na Seção 2.3.2. Para sanar este problema, são utilizados equalizadores de fase. Um equalizador nada mais é que um segundo filtro, colocado em

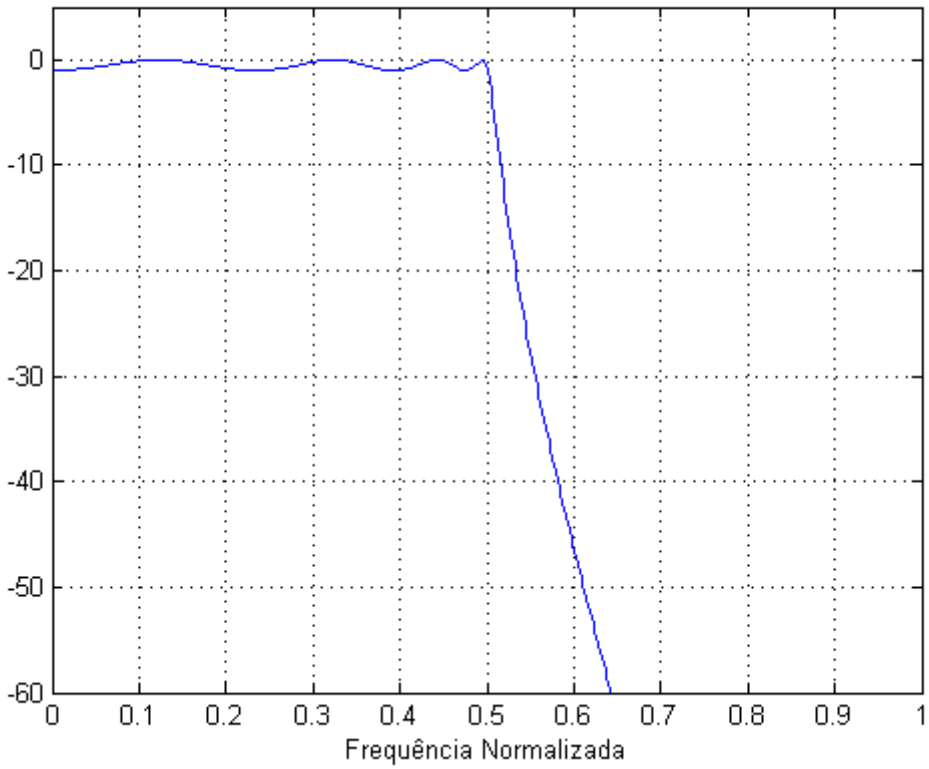


Figura 3: Resposta de um filtro de Chebyshev tipo I de oitava ordem

Fonte: Autoria própria.

cascata com o primeiro, com o objetivo de tornar linear a resposta em fase do sistema completo. Obviamente, um requisito básico deste filtro é ter ganho unitário em todo o espectro, para não modificar a resposta que foi projetada conforme as especificações. Para obedecer a este critério, sua função de transferência precisa obedecer à seguinte condição:

$$H(z) = \frac{a_n \cdot z^n + a_{n-1} \cdot z^{n-1} + \ \ldots \ + a_2 \cdot z^2 + a_1 \cdot z + 1}{z^n + a_1 \cdot z^{n-1} + \ \ldots \ + a_{n-2} \cdot z^2 + a_{n-1} \cdot z + a_n} \tag{11}$$

Ou seja, o vetor de coeficientes b_k deve ser uma versão espelhada do vetor a_k. Esta característica facilita a solução deste filtro por métodos de otimização, pois o número de graus de liberdade cai pela metade, reduzindo grandemente o espaço de busca. Outro fator importante é que, salvo aplicações específicas, só é necessário garantir a linearidade da resposta em fase na banda de passagem, já que as informações contidas na banda de rejeição serão descartadas e, portanto, não precisam ser preservadas.

Os métodos mais utilizados para a equalização de filtros são por simetria de resposta ao impulso, e por minimização de dispersão em torno do atraso de fase (NOCETI FILHO, 1998).

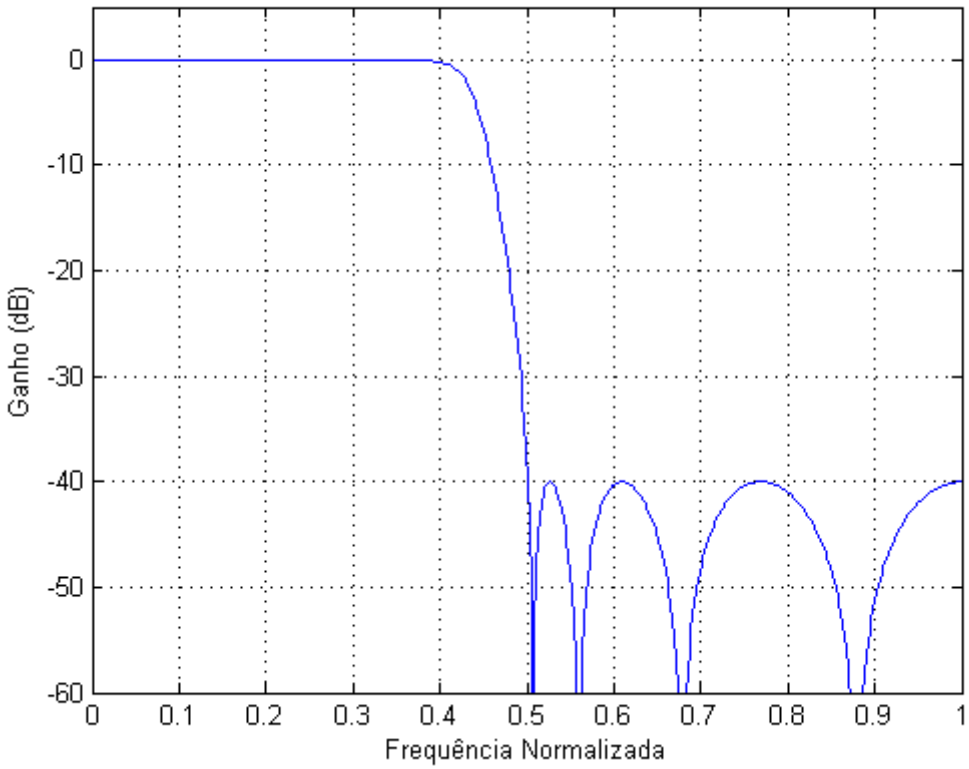


Figura 4: Resposta de um filtro de Chebyshev tipo II de oitava ordem

Fonte: Autoria própria.

O atraso de fase é utilizado em lugar do atraso de grupo, pois a utilização do segundo pode levar a uma função incrementalmente linear, ao invés de linear (NOCETI FILHO, 1998). Ou seja, se a fase não passar pela origem ou um múltiplo de π em $\omega = 0$, ocorre a chamada interceptação de fase. Desta forma, ter atraso de grupo constante não quer dizer necessariamente que não haja distorção de fase (DINIZ et al., 2004).

A equalização por simetria de resposta ao impulso parte do princípio de que um sistema, cuja resposta ao impulso é simétrica em torno de um instante T_s, possui resposta em fase linear. Este método da simetria busca então minimizar o erro de simetria, isto é, encontra os coeficientes do filtro que, posto em cascata com o filtro original, torna simétrica a resposta ao impulso do sistema.

2.4 COMPUTAÇÃO EVOLUCIONÁRIA

A Computação Evolucionária (CE) é a área de Inteligência Computacional que busca a resolução de problemas de otimização através de processos iterativos e estocásticos. A cada iter-

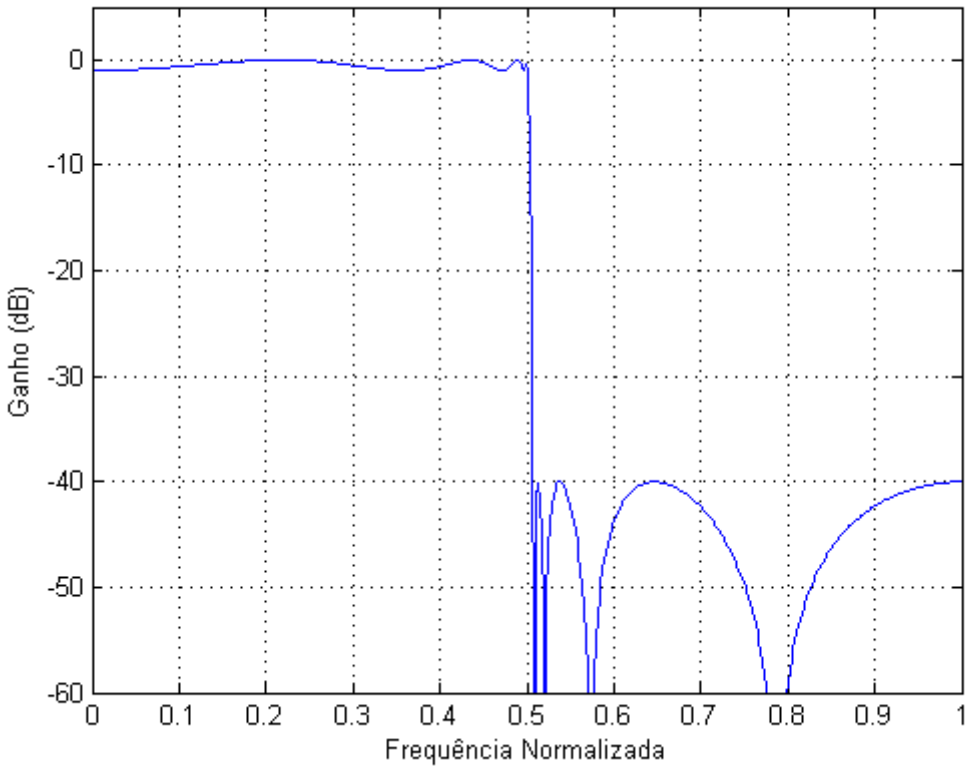


Figura 5: Resposta de um filtro elíptico de oitava ordem

Fonte: Autoria própria.

ação são geradas novas soluções candidatas, geralmente chamadas de indivíduos, baseadas nas soluções anteriores, através de regras de meta-heurística. Estas meta-heurísticas são inspiradas em processos naturais, como mecanismos biológicos. Através da sucessiva aplicação destas regras no conjunto dos indivíduos, normalmente chamado de população, chega-se a soluções aproximadas potencialmente boas para o problema.

Existem diversos algoritmos de CE populares na literatura. Pode-se citar os Algoritmos Genéticos (AG) (GOLDBERG, 1989), Otimização por Enxame de Partículas (PSO) (KENNEDY; EBERHART, 1995), Otimização por Colônia de Formigas (ACO) (DORIGO; STÜTZLE, 2004), *Harmony Search* (HS) (GEEM et al., 2001), Programação Genética (GP) (KOZA, 1992), Otimização por Enxame Bioluminescente (BSO) (OLIVEIRA et al., 2011a), entre outros. Os mais utilizados são o PSO e o AG, que serão melhor explicados nas Seções 2.4.1 e 2.4.2.

2.4.1 Otimização por Enxame de Partículas

A Otimização por Enxame de Partículas (PSO) foi proposta por (KENNEDY; EBERHART, 1995), e é um algoritmo inspirado no movimento sincronizado de cardumes de peixes e bandos de pássaros. Neste algoritmo, cada solução candidata é representada por uma partícula, cuja posição em um espaço N-dimensional é a representação numérica desta solução. Cada partícula possui também velocidade própria e inércia, isto é, a força da influência da velocidade atual na mudança de velocidade na próxima iteração. A navegação do grupo de partículas, também chamado de enxame, é guiada pelo melhor resultado encontrado pelo enxame como um todo, chamado de máximo social ou *gbest*, pela melhor solução já encontrada pela própria partícula, chamado de máximo cognitivo ou *lbest*, e pela velocidade anterior da partícula. O peso de cada um destes resultados na alteração de velocidade e posição das partículas é dado pelas constantes c_g e c_l, que regem respectivamente a influência de *gbest* e *lbest*. A inércia é dada por w, e o tamanho do enxame ou número de partículas é chamado de M. Algumas implementações utilizam também o *pbest*, que é o melhor resultado encontrado pelo grupo do qual a partícula faz parte.

As partículas são inicializadas em velocidade e posições aleatórias, no início da execução. O Algoritmo 1 descreve os passos tomados em cada iteração, buscando-se otimizar uma função $f(\vec{x})$.

A equação mostrada na linha 9 do Algoritmo 1 diz respeito à velocidade da partícula, que é o núcleo da heurística do PSO. Seu efeito é levar a partícula a ser atraída ao mesmo tempo para o *gbest* e para o *lbest*, balanceando a relação entre busca global e busca local, respectivamente, pelas constantes c_g e c_l.

O ajuste destes dois parâmetros, juntamente com o momento de inércia w, é objeto de extensos estudos na literatura (TEWOLDE et al., 2009; LEONG W. F. ; YEN, 2008; YAMAGUCHI; YASUDA, 2006; TANG; ZHAO, 2009). Além de testes exaustivos para a calibração do algoritmo para problemas específicos, existe também a possibilidade de se adicionar uma outra camada de otimização, que busque otimizar o ajuste destes parâmetros de forma heurística. Esta abordagem é chamada de meta-otimização (TEWOLDE et al., 2009).

2.4.2 Algoritmos Genéticos

O Algoritmo Genético (AG) é um método de otimização baseado no mecanismo de mutação e seleção natural observado na natureza. Foi proposto por (HOLLAND, 1975) e mais tarde popularizado por (GOLDBERG, 1989). Ele combina conceitos de genética com as idéias

Algoritmo 1: Otimização por Enxame de Partículas (PSO)

```
1  Configura parâmetros: M, c_g, c_l, w;
2  Inicializa de forma aleatória o vetor de posição das partículas x_i;
3  Inicializa de forma aleatória o vetor de velocidade das partículas v_i;
4  Inicializa o vetor lbest_i = x_i;
5  Encontra o ótimo global gbest;
6  enquanto Critério de parada não for satisfeito faça
7      para cada partícula i faça
8          //atualiza velocidade, onde rand é um número
           aleatório;
9          v_i ← w·v_i + c_g·rand·(gbest − x_i) + c_l·rand·(lbest − x_i);
10         //atualiza posição;
11         x_i ← x_i + v_i;
12         se f(x_i) > f(lbest_i) então
13             //atualiza ótimo da partícula;
14             lbest_i ← x_i;
15         fim se
16         se f(x_i) > f(gbest) então
17             //atualiza ótimo do enxame;
18             gbest ← x_i;
19         fim se
20     fim para
21 fim enqto
22 Pós-processamento e visualização de dados;
```

evolutivas propostas por Charles Darwin. Neste algoritmo, cada solução é representada por um indivíduo de uma população. A cada iteração, aqui chamadas de geração, os indivíduos geram soluções "filhas"através de cruzamento e mutação, formando assim uma nova geração de indivíduos. A probabilidade de cada indivíduo participar da criação de um indivíduo da próxima geração cresce com a qualidade da solução que ele representa, qualidade esta chamada de *fitness*. Portanto, os indivíduos com o melhor *fitness* são os melhor adaptados ao meio, distribuindo sua carga genética para uma parte maior da próxima geração.

Os principais parâmetros que controlam esta evolução são o tamanho da população (*popsize*), a probabilidade de mutação (*pmut*), a probabilidade ocorrer cruzamento (*pcross*), e o número de gerações (*maxgen*). Em alguns casos, o melhor ou alguns dos melhores indivíduos de uma geração são mantidos na seguinte, gerando o que se chama de elitismo. O AG canônico é exemplificado no Algoritmo 2.

Algoritmo 2: Algoritmo Genético (AG)

```
1  Configura parâmetros: popsize, pmut, pcross;
2  Inicializa de forma aleatória o vetor de cromossomos dos indivíduos x_i;
3  enquanto Critério de parada não for satisfeito faça
4      Calcula o fitness de cada indivíduo;
5      para i = 1 até popsize faça
6          Seleciona 2 indivíduos pais, com maior probabilidade para os mais aptos
           (maior fitness);
7          Efetua operações de cruzamento e mutação entre os pais;
8          Insere o novo indivíduo na próxima geração;
9      fim para
10 fim enqto
11 Pós-processamento e visualização de dados;
```

2.5 TRABALHOS CORRELATOS

No artigo de (CHEN; CHEN, 2006), é proposta uma modificação no algoritmo PSO para a evolução de filtros digitais IIR. Esta nova abordagem utiliza diferentes pesos no cálculo de velocidade de cada partícula, de acordo com a qualidade da solução representada por ela. Esta modificação busca calibrar a busca do algoritmo alterando a ênfase entre local e global. Porém, este peso individual faz com que partículas com boas soluções efetuem uma busca mais local, enquanto partículas com baixo *fitness*, que provavelmente estão em vizinhaças de baixa qualidade, efetuarão sua busca de forma global, para explorar melhor o espaço. São propostas duas variações deste esquema, chamadas de DPVU-PSO e CPVU-PSO. Para avaliar a qualidade destas modificações, foi analisado o desempenho dos algoritmos em problemas de *benchmarking*, e na identificação de filtros IIR. Os resultados encontrados são melhores do que os de outras variações do PSO inclusive o canônico, mas não é feita uma comparação com métodos clássicos de cálculo de filtro.

Já em (GAO et al., 2008), a modificação feita é chamada de CPSO, ou *Chaotic* PSO. Nela, é inserido um fator caótico em vez de aleatório no cálculo da velocidade das partículas. Além disto, são geradas várias populações iniciais em vez de apenas uma, e a melhor delas é selecionada para ser trabalhada pelo algoritmo. O resultado do CPSO é comparado apenas com o PSO canônico e com a resposta ideal de cada filtro. Ou seja, como não é mostrado o desempenho de um método como Chebyshev ou Butterworth em comparação com o obtido pelo CPSO, não é possível saber se há alguma vantagem nesta proposta, já que o custo computacional é obviamente maior. Porém, o CPSO mostrou desempenho melhor do que o PSO canônico.

A introdução de fatores caóticos é novamente utilizada em (JIA et al., 2009). Aqui, é

utilizada a chamada mutação caótica, em que a melhor partícula de cada iteração sofre mutação de forma caótica, para melhorar a busca local do algoritmo. O resultado é comparado com o do PSO canônico, assim como com o dos métodos de Butterworth, Chebyshev tipo I e Chebyshev tipo II. Embora o CPSO tenha sido superado apenas pelo Chebyshev tipo II, apenas um filtro é utilizado para este teste. Sendo assim, é difícil dizer se o método realmente é satisfatório.

O trabalho de (KRUSIENSKI; JENKINS, 2004) aborda a equalização de filtros IIR utilizando um PSO canônico. Os resultados obtidos são satisfatórios, mas não são comparados com nenhum dos métodos clássicos já citados. Além disto, apenas equalizadores de primeira e segunda ordem são testados. Por mais que esta ordem reduzida fosse suficiente para equalizar os filtros propostos, o baixo número de dimensões da partícula torna difícil avaliar o PSO para o problema de forma mais genérica, já que otimizar problemas de apenas uma ou duas variáveis costuma ser muito simples, sendo viável em boa parte das vezes a aplicação de busca exaustiva.

Outra variação do PSO utilizada para o cálculo de filtros IIR é o PSO-QI, proposta em (LUITE; VENAYAGAMOORTHY, 2008). Nela, as partículas do PSO são tratadas como partículas da física quântica, em que velocidade e posição não podem ser determinadas com precisão ao mesmo tempo. Desta forma, as partículas são representadas por uma função de densidade de probabilidade, que resulta na probabilidade de uma partícula estar em determinada posição num determinado instante. Esta abordagem mostrou resultados melhores que o PSO, mas não foi comparada com métodos fortes.

Um estudo mais profundo da evolução de filtros é feita em (PAN, 2011). Aqui, a evolução de filtros IIR é testada utilizando PSO, AG, Evolução Diferencial (DE) e uma versão modificada do AG chamada de *Improved Genetic Algorithm* (IGA). O autor deduz um método de verificação de estabilidade que exige menos esforço computacional, e compara o desempenho de todos os algoritmos. Porém, apenas filtros de segunda ordem são utilizados nos experimentos, tornando os problemas simples de serem resolvidos por otimização numérica. Outro problema é a ausência de comparação com os métodos clássicos, que daria uma real medida da qualidade dos resultados obtidos pelos algoritmos.

Em (YU et al., 2009), é proposto um PSO com esquema adaptativo para o momento de inércia. Nele, cada partícula possui um momento de inércia próprio, baseado em seu *fitness* e corrigido por uma função sigmóide. De forma similar à proposta de (CHEN; CHEN, 2006), o objetivo é dar ênfase à busca local nas proximidades das melhores soluções encontradas, e utilizar as partículas em vizinhanças ruins para explorar o espaço de busca de forma mais global. O algoritmo proposto é chamado de AIW-PSO, e apresentou resultados melhores do que o PSO canônico. Porém, o resultado não foi comparado com filtros obtidos pelos métodos

clássicos.

O trabalho de (ZHANG et al., 2010) propõe um esquema de redistribuição de partículas em caso de convergência prematura. Esta redistribuição nada mais é do que a reinicialização das partículas, chamada em outros trabalhos de explosão. Neste artigo, o critério utilizado para a detecção de convergência é a distância média entre cada partícula e o melhor resultado encontrado. Esta verificação acarreta em aumento do custo computacional, já que é mais comum utilizar o número de iterações sem um dado aumento na qualidade da melhor solução. O resultado desta proposta, chamada de RPSO, é comparada com o PSO canônico, QPSO, CPSO e o algoritmo de Cultura Diferencial (DC). O RPSO alcançou melhores resultados, mas estes não foram comparados com métodos fortes.

2.5.1 Análise

De todos os trabalhos correlatos mostrados na Seção 2.5, todos utilizam a mesma codificação, através dos coeficientes do filtro, e a mesma função de *fitness*, através do erro quadrático médio (MSE). Nenhum propõe alguma mudança na codificação do problema de filtros digitais, nem no cálculo da função de *fitness*. Além disto, a análise da estabilidade dos filtros só é tratada em (PAN, 2011). Como os outros autores não citam se avaliaram ou não a estabilidade, é arriscado comparar estes resultados com os obtidos por outros métodos, já que esta comparação é provavelmente injusta.

Outro problema presente em praticamente todas estas referências é a escassez de comparações entre os resultados obtidos por suas abordagens e os alcançados pelos métodos de Butterworth, Chebyshev e Elíptico. Afinal, ao se propor uma nova abordagem para um problema que possui soluções clássicas e largamente utilizadas, espera-se que o desempenho entre a proposta e o tradicional seja comparado, com suas diferenças ressaltadas.

De forma geral, o cálculo de filtros IIR por otimização é tratado na maior parte da literatura como um simples problema de *benchmark*, e não como um problema de Engenharia. A maior parte dos autores utiliza-o para comparar o desempenho entre algoritmos de CE e variações propostas, em vez de buscar novas abordagens na codificação e cálculo de *fitness* que possibilitem o projeto de filtros por otimização melhores que os obtidos por métodos clássicos.

3 METODOLOGIA

3.1 CÁLCULO DE FITNESS DE FILTROS DIGITAIS EM CE

Conforme citado no Capítulo 2, a otimização de problemas numéricos através de algoritmos de Computação Evolucionária depende diretamente da possibilidade de se comparar a qualidade de duas soluções distintas de forma objetiva. Isto é, é necessário que exista uma função $f(\vec{x})$ que forneça uma medida numérica da qualidade da solução candidata $\vec{x}$. Esta função é chamada de função de *fitness*.

Na otimização de filtros digitais, é usual a utilização do Erro Quadrático Médio (MSE) para medir a qualidade das soluções. Esta abordagem compara a resposta em frequência do candidato com a resposta de um filtro ideal com as mesmas especificações, mensurando a diferença na forma do MSE. Quando mais próximo do filtro ideal, menor será a MSE, portanto este problema é tratado como minimização. Sendo $f(\vec{x})$ o *fitness* da solução $\vec{x}$, $X(z)$ a resposta em frequência do filtro proposto por esta solução e $Y(z)$ a resposta do filtro ideal, tem-se a Equação 12.

$$f(\vec{x}) = \sum_{i=1}^{N} \frac{[Y(i) - X(i)]^2}{N} \tag{12}$$

Esta abordagem tem alguns problemas. É possível que um filtro que esteja dentro das especificações tenha um *fitness* pior do que um filtro fora das especificações. Isto ocorre porque utilizando o filtro ideal como alvo, as margens de tolerância especificadas não são utilizadas no cálculo. Um filtro ideal possui ganho $-\infty$dB na banda de rejeição, ganho este inalcançável pelo filtro sendo otimizado. Porém, um filtro que possua ganhos próximos de -80dB nesta faixa terá um erro menor do que outro perto de -50dB. Mas se a especificação dizia que era desejável um ganho abaixo de -40dB, ambos estão aceitáveis na faixa de rejeição.

Supondo que este mesmo filtro admita um *ripple* de 1dB na banda de passagem, que o primeiro filtro tenha um ripple de 1.1dB e o segundo de 0.9dB, é fácil concluir que o segundo filtro é o melhor, pois atende completamente às especificações. Porém, é provável que o *fitness*

do primeiro seja melhor, já que o erro em comparação com o filtro ideal é menor. Estas situações são exemplificadas nas Figuras 6 e 7.

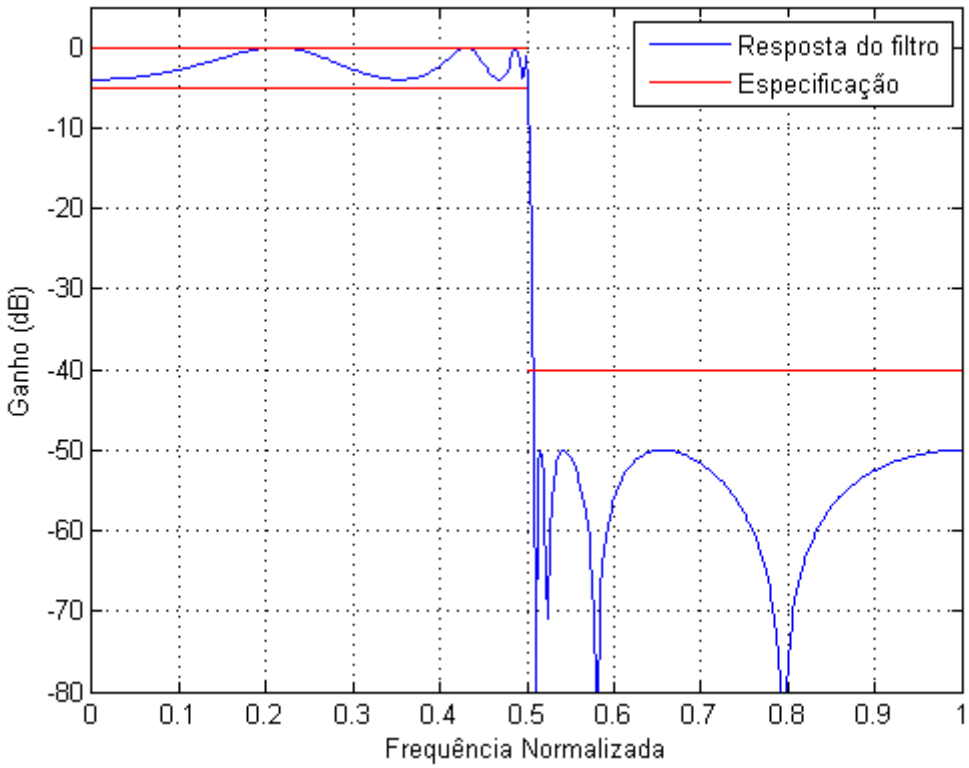


Figura 6: Filtro dentro das especificações

Fonte: Autoria própria.

As especificações utilizadas nas Figuras 6 e 7 são 5dB de *ripple* na banda de passagem, e 40dB de atenuação na banda de rejeição. O filtro da Figura 6 obedece a estes critérios, como é possível notar. Já o filtro da Figura 7 tem um *ripple* maior do que o permitido, ou seja, apesar de ter a atenuação melhor na banda de rejeição, não obedece os critérios na banda de passagem. Portanto, o melhor filtro é o primeiro, da Figura 6.

Porém, como já citado, a qualidade do filtro pela MSE não é baseada na especificação, e sim no filtro ideal. E como o erro entre a banda de rejeição ideal e a banda de rejeição do primeiro filtro é muito maior do que o erro entre a banda de passagem ideal e a banda de passagem do segundo filtro, a MSE acusa o segundo filtro como sendo melhor.

Outro problema da abordagem por MSE é que a banda mais larga terá maior influência no erro. Por exemplo, se um filtro passa-baixa com frequência de corte $0.2w_s$ estiver sendo otimizado, amostrando-se em 100 pontos na resposta em frequência, 20 pontos pertencerão à banda de passagem e 80 pontos pertencerão à banda de rejeição. Desta forma, erros grandes na

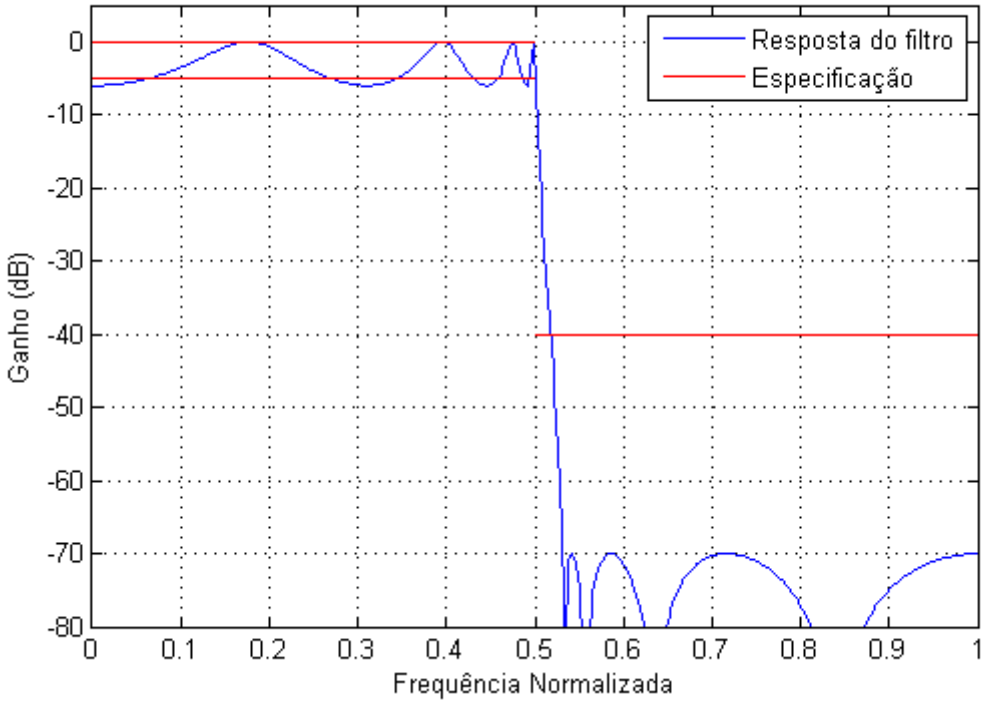


Figura 7: Filtro fora das especificações

Fonte: Autoria própria.

banda de passagem serão mascarados por erros pequenos na banda de rejeição, levando filtros ruins a terem *fitness* altos.

Para suprir a necessidade de uma forma mais precisa de medição de qualidade dos filtros, propõe-se neste trabalho uma nova função de *fitness*. Esta função avalia o filtro de acordo com critérios de obediência às especificações, respeitando diferenças de largura entre as bandas.

Para medir o *fitness* do filtro, primeiramente deve-se calcular a resposta em frequência do mesmo, já que é esta característica que deve ser otimizada. Sendo a função de transferência generalizada dada na Equação 13

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + ... + b_M z^{-M}}{a_0 + a_1 z^{-1} + a_2 z^{-2} + ... + a_N z^{-N}} \tag{13}$$

deve-se substituir z por e^{jw}, chegando-se na resposta em frequência dada em 14:

$$H(e^{jw}) = \frac{b_0 + b_1 e^{-jw} + b_2 e^{-j2w} + ... + b_M e^{-jMw}}{a_0 + a_1 e^{-jw} + a_2 e^{-j2w} + ... + a_N e^{-jNw}} \tag{14}$$

Esta resposta é então amostrada em k pontos igualmente espaçados, sendo que mais pontos induzem maior precisão, mas também aumentam o esforço computacional envolvido na tarefa. Como apenas a magnitude da resposta em frequência é interessante para esta parte do trabalho, é calculado o módulo de cada elemento deste vetor. Além disto, metade dos elementos é descartada. Isso ocorre por que a resposta em frequência é simétrica em relação a $\frac{Fs}{2}$, ou seja, apenas a primeira metade da resposta é necessária para efetuar os cálculos. Não somente isto, mas o critério de Nyquist (LATHI, 1998) diz que não se pode utilizar sinais com componentes em frequência acima de $\frac{Fs}{2}$. Logo, apenas a metade inferior da resposta em frequência é relevante.

Este vetor de amostras da resposta em frequência é chamado de $\vec{r}$, sendo que a metade superior já foi descartada. De posse de $\vec{r}$, separa-se este vetor em $\vec{r_{rej}}$ e $\vec{r_{pas}}$, respectivamente, a resposta na banda de rejeição e a resposta na banda de passagem. Desta forma é mais simples fazer a avaliação da qualidade do filtro, pois o requisito é constante dentro de cada banda.

Obtendo-se as respostas nas bandas especificadas, pode-se calcular o *fitness* deste filtro, composto por quatro termos a serem otimizados, conforme a Equação 15.

$$fit(\vec{x}) = \frac{1}{1 + p1 \cdot specRej + p2 \cdot specPas + p3 \cdot idealRej + p4 \cdot idealPas} \tag{15}$$

onde $fit(\vec{x})$ é o valor de *fitness* do vetor $\vec{x}$ de coeficientes do filtro a ser avaliado, $specRej$ é o termo que diz respeito à obediência da banda de rejeição às especificações, $specPas$ se relaciona com o atendimento da banda de passagem, $idealRej$ e $idealPas$ dizem o quão próximo se está da resposta ideal em cada banda, e $p1$ a $p4$ são os pesos de cada um dos termos na equação.

Para calcular o valor de $specRej$, é utilizada a Equação 16.

$$specRej = \frac{\sum_{i=1}^{k} corrigeBordaRej \cdot [max(gMaxRej(i), r_{rej}(i)) - gMaxRej]^2}{largRej} \tag{16}$$

onde k é o número de elementos do vetor r_{rej}, $gMaxRej$ é o máximo ganho permitido na banda de rejeição, $corrigeBordaRej$ é o fator de correção de borda, e $largRej$ é a largura em quantidade de amostras da banda de rejeição. Este modo de calcular o erro na banda de rejeição é similar ao MSE, mas apenas considera os elementos da resposta que tenham ganho acima do permitido. Ou seja, mesmo que esta resposta na banda de rejeição varie com um *ripple* muito elevado, ele terá erro zero se este *ripple* estiver sempre abaixo do máximo. Desta forma, este

termo mede de forma eficaz a obediência do filtro à especificação da banda de rejeição. A divisão pela largura da banda visa a normalização do erro por banda. Isto é necessário pois como as bandas de um filtro têm larguras diferentes, um erro grande na banda mais estreita geraria um erro numericamente inferior a um erro pequeno na banda mais larga.

O fator de correção de erro de borda $corrigeBordaRej$ serve para aumentar o peso das amostras próximas da fronteira das bandas no cálculo do erro. A resposta em frequência de um filtro é sempre contínua. Dessa forma, ganhos de frequências próximas também são próximos, o que leva eventuais desobediências às especificações a aparecerem em grupos. Porém, quanto mais próximo da borda da banda, ou seja, do limite onde a banda inicia ou termina, esse grupo de amostras pode estar para fora da banda. Desta forma, pode haver apenas uma ou duas frequências amostradas violando o ganho da banda, mas gerando um erro tão pequeno no cálculo que acaba não influenciando a evolução das soluções.

A alternativa proposta para amenizar este problema é multiplicar cada amostra por um peso diferente, fazendo com que violações próximas à borda influenciem de forma significativa o valor do *fitness* relativo à esta banda. Esta curva de pesos é calculada conforme a Equação 17:

$$corrigeBordaRej(i) = max(90, [\frac{10 \cdot (1-i)}{largRej}]^2) - 89 \tag{17}$$

onde i é o número da amostra correspondente à frequência e $largRej$ é a largura da banda de rejeição. Esta equação gera uma curva de valor 1 por quase toda a banda de rejeição, exceto no primeiro valor, com peso 11, e o segundo, com peso intermediário dependendo da largura da banda. Para uma banda com 40 amostras de largura, este valor é aproximadamente 5,6. Desta forma, temos que a influência das frequências na fronteira da banda de passagem tem peso muito maior no cálculo de *fitness* do que as outras frequências, corrigindo o problema anteriormente citado.

O termo $specPas$, que corresponde ao erro de atendimento à especificação da banda de passagem, é calculado segundo a Equação 18.

$$specPas = \frac{\sum_{i=1}^{k} corrigeBordaPas(i) \cdot [[min(gMinPas, r_{pas}(i)) - gMinPas]^2 + [max(1, r_{pas}(i)) - 1]^2]}{largPas} \tag{18}$$

onde k é o número de elementos do vetor r_{pas}, $largPas$ é a largura da banda de passagem em número de amostras, e $gMinPas$ é o mínimo ganho permitido na banda de passagem, que também pode ser interpretado como o máximo *ripple* permitido, já que, por definição, o ganho

máximo é 1. Este termo então calcula o erro quadrático em relação à faixa de ganho permitida na banda de passagem. Portanto, o ganho pode estar em qualquer ponto entre 1 e *gMinPas*, e terá erro zero. Desta forma, apenas ganhos que desobedecerem a especificação da banda de passagem gerarão erro. O termo de correção de borda é similar, com a diferença de que os valores elevados devem estar no final do vetor, e não no começo. A Equação 19 mostra a modificação efetuada:

$$corrigeBordaPas(i) = max(90, [\frac{10 \cdot i}{largPas}]^2) - 89 \tag{19}$$

As curvas obtidas pelas equações de correção de borda são mostradas na Figura 8.

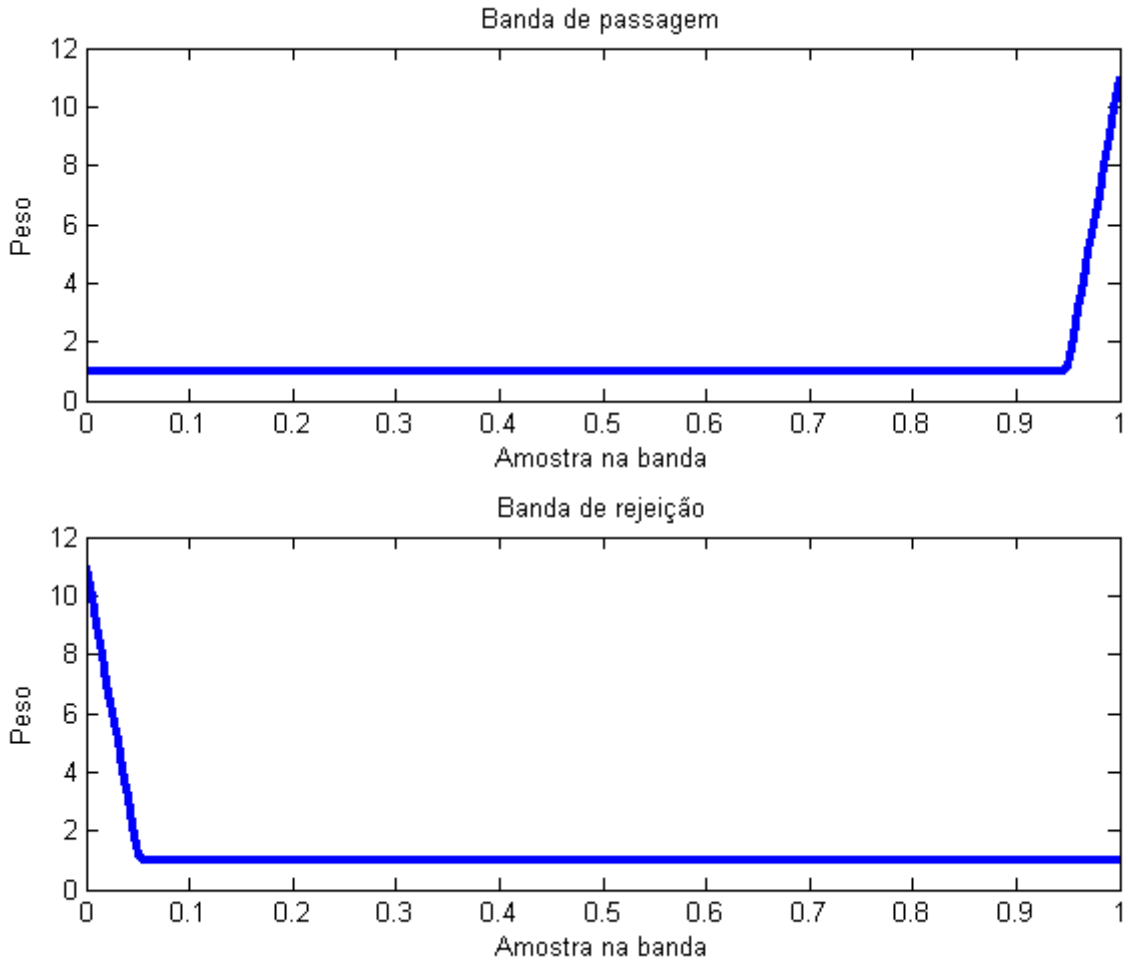


Figura 8: Curvas de correção de borda

Fonte: Autoria Própria.

Se os termos *specPas* e *specRej* avaliam a conformidade do filtro às especificações, os outros fornecem uma medida de qualidade do filtro no sentido de proximidade com a resposta

ideal. Eles são desejáveis pois os termos *spec* não diferenciam entre filtros que obedeçam às especificações, sendo que uma boa função de *fitness* deve levar o algoritmo a buscar um filtro com a melhor resposta possível. As Equações 20 e 21 mostram como estes termos são calculados.

$$idealPas = \frac{\sum_{i=1}^{k} corrigeBordaPas \cdot [1 - r_{pas}(i)]^2}{largPas} \tag{20}$$

$$idealRej = \frac{\sum_{i=1}^{k} corrigeBordaRej \cdot [r_{rej}(i)]^2}{largRej} \tag{21}$$

Estas equações são mais simples pois um filtro ideal tem um valor fixo para o ganho em cada banda, em vez de uma faixa de valores que a especificação supõe. A consequência de ter estes dois termos a mais na equação do *fitness* é que jamais existirá um filtro com *fitness* igual a 1, pois para isto os 4 termos da equação precisariam ser iguais a zero. Enquanto os termos *spec** podem, e devem, chegar a zero ao final da evolução, os termos *ideal** jamais atingirão este valor, já que apenas um filtro ideal teria tal característica.

3.2 NOVA CODIFICAÇÃO DO PROBLEMA

Como citado na Seção 2.5, em todos os trabalhos analisados até então, utilizou-se a codificação direta dos coeficientes como representação dos indivíduos. Esta abordagem é simples de ser implementada, porém, como é necessário garantir a estabilidade do filtro, esta codificação possibilita a existência de filtros inválidos, o que gera a necessidade de um sistema de penalizações (GOLDBERG, 1989). Outro problema é que é difícil de se delimitar o espaço de busca, pois o valor de cada coeficiente varia muito de grandeza. Se o limite for alto, tende-se a perder precisão nos valores pequenos. Se o limite for baixo, perde-se a chance de se explorar regiões do espaço de busca que podem conter boas soluções.

Como a estabilidade de um sistema LSI é atingida quando todos os pólos estão dentro do círculo de raio unitário, isto é, possuem módulo menor que 1, é interessante buscar uma forma de representar através dos coeficientes apenas polinômios com raízes menores do que 1. O trabalho de (PAN, 2011) conseguiu uma solução rápida para verificar a estabilidade de um filtro IIR mas, ainda assim, é necessário avaliar a estabilidade de cada indivíduo a cada iteração do algoritmo de CE. O ideal seria mapear o espaço de busca em um sub-espaço composto apenas pelos filtros estáveis. Esse sub-espaço deve conter todos os filtros estáveis possíveis, e apenas estes.

A Figura 9 mostra a representação por pólos e zeros de um filtro passa-baixas qualquer, calculado pelo método elíptico. Nesta figura, os pólos do sistema são representados pelos X, e os zeros pelos círculos. É possível observar que todos estão dentro do círculo de raio unitário, que como já discutido, é a condição para que o filtro seja BIBO estável. Os coeficientes deste filtro são mostrados na Tabela 10.

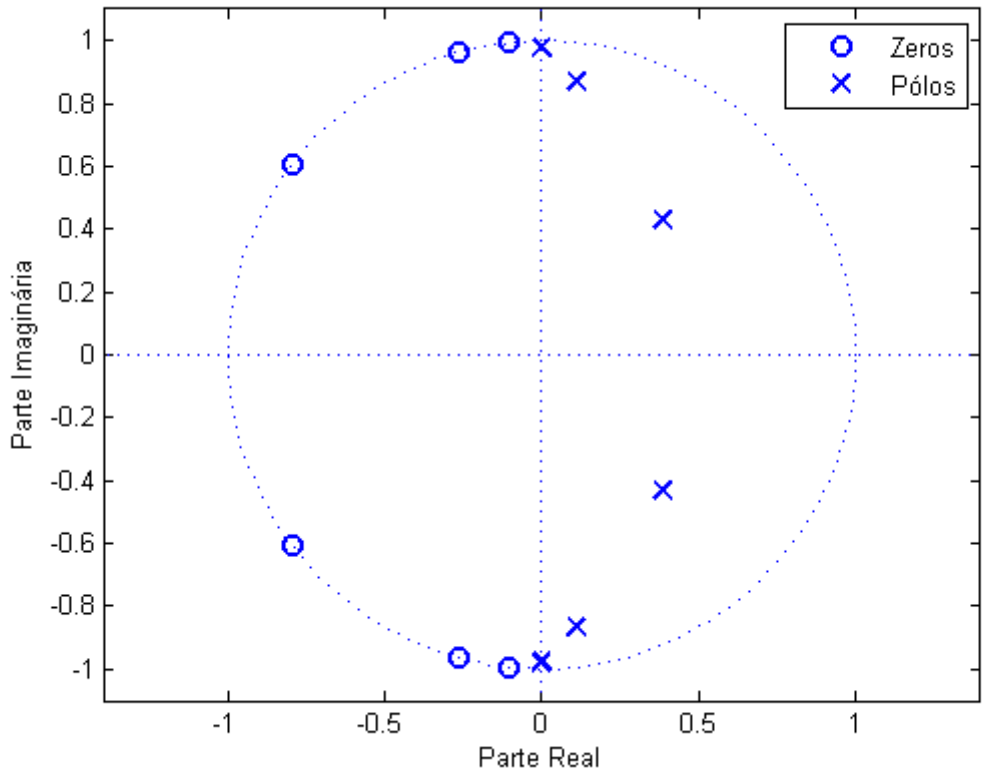


Figura 9: Diagrama de pólos e zeros de um filtro passa-baixas

Fonte: Autoria Própria.

k	0	1	2	3	4	5	6
Numerador	1,0000	-0,9901	2,2229	-1,6076	1,4668	-0,6326	0,2446
Denominador	0,0757	0,1762	0,3244	0,3660	0,3244	0,1762	0,0757

Figura 10: Coeficientes de um filtro passa-baixas

Na Tabela 10, é possível verificar que a magnitude dos valores varia consideravelmente. O coeficiente 2 do numerador é quase 30 vezes maiordo que o coeficiente 0 do denominador. Se o problema for abordado com um algoritmo evolucionário do tipo discreto, como o GA, isso pode tornar necessário uma elevada resolução nas variáveis, aumentando grandemente o espaço de busca. Mesmo com codificações contínuas, a precisão necessária nos valores baixos pode diminuir a velocidade de convergência nos valores elevados. E de qualquer forma, não há

um limite nestes valores que não seja totalmente arbitrário. Não há como garantir que o filtro que melhor atenda as especificações desejadas possua todos os seus coeficientes dentro de uma determinada faixa de valores.

Para sanar estes dois problemas (estabilidade e limites numéricos dos coeficientes), propõe-se aqui uma nova codificação. Esta abordagem representa o filtro diretamente através de seus pólos e zeros, em vez de representá-lo pelos coeficientes dos polinômios. Estes podem ser representados na forma polar ou retangular, sendo esta escolha analisada na seção seguinte.

3.2.1 Representação polar x representação retangular

Para se transformar pólos e zeros em polinômios, utiliza-se aqueles em sua forma retangular. Portanto, em primeira análise esta forma seria a melhor representação, pois para utilizar a forma polar, seria necessário executar uma transformação de sistema de coordenadas para então poder calcular os coeficientes do filtro.

Para representar estes pólos na forma retangular, são utilizadas suas coordenadas x e y em vez de raio e ângulo. Apesar desta abordagem permitir uma fácil limitação dos valores, fica difícil garantir a estabilidade. Para que nenhum filtro instável seja representável, o limite tem que ser de $\pm\frac{1}{\sqrt{2}}$ tanto para x quanto para y. Este limite mapeia o espaço de busca em um quadrado no plano z, centrado em $(0,0)$, tocando o círculo de raio unitário em seus vértices. Este espaço está representado na Figura 11.

O maior problema desta representação retangular é que há uma parte considerável do espaço de busca original com filtros estáveis sendo descartado. Na Figura 11, a parte sombreada em cinza é o espaço mapeado pelo limite de $\pm\frac{1}{\sqrt{2}}$. A porção dentro do círculo de raio unitário que não está sombreada não pode ser representada desta forma. Esse espaço não alcançável representa cerca de 36.3% de todos os filtros estáveis do espaço de busca.

Para representar todos os filtros estáveis possíveis, o limite deveria ser ± 1, como na Figura 12. Com estes limites, o círculo de raio unitário está contido no espaço de busca, ou seja, todos os filtros estáveis estão representados. Porém, uma grande parte da área sombreada está fora do círculo, ou seja, representam filtros instáveis. Isto aumenta o espaço de busca em 21.4% apenas com soluções inválidas. Por isto, a melhor solução é utilizar a representação polar, em que as variáveis que representam cada pólo ou zero são o seu raio, ou seja, a distância até a origem, e o ângulo, medido de forma anti-horária a partir do eixo positivo x. Com esta abordagem, o espaço de busca tem forma circular em vez de retangular. Limitando-se o raio entre 0 e 1, tem-se novamente o espaço de busca representado como a parte sombreada da

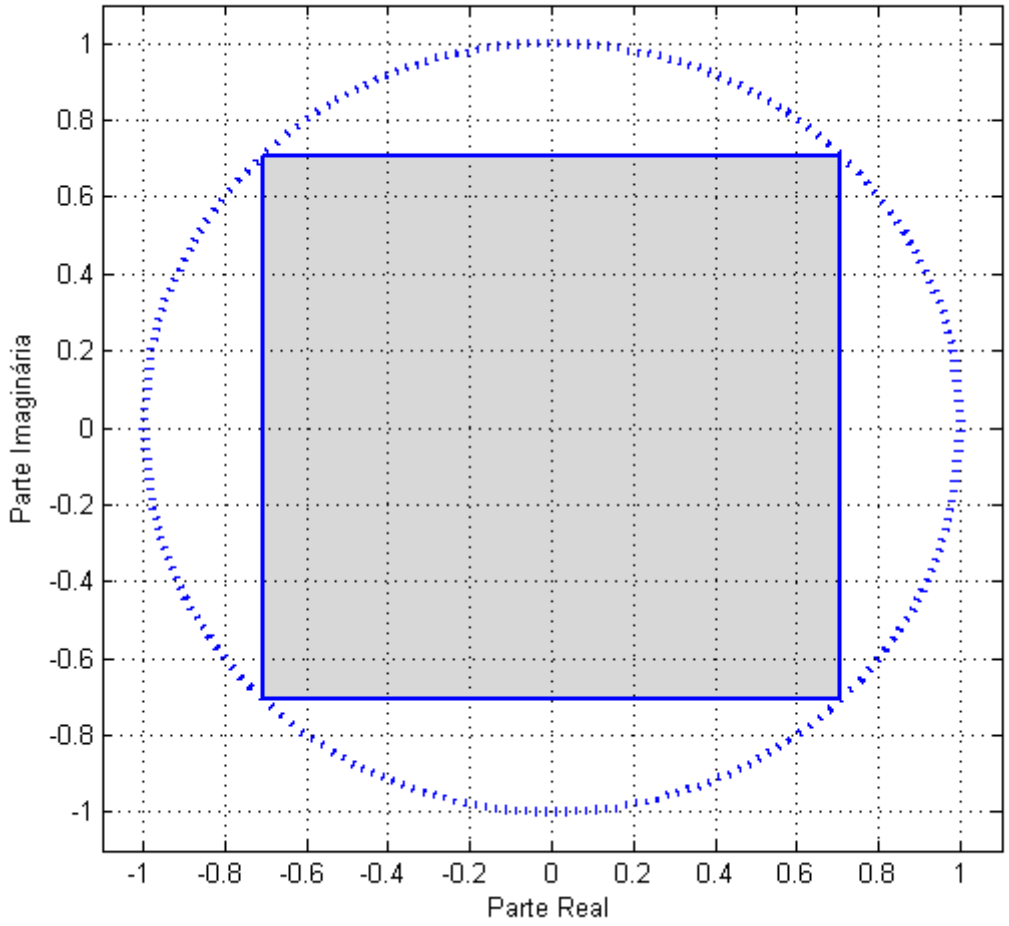


Figura 11: Espaço de busca com representação de pólos de forma retangular

Fonte: Autoria Própria.

Figura 13.

Portanto, foi escolhida a representação dos pólos e zeros na forma polar, ou seja, por seus raios (ou módulos) e ângulos. O módulo dos pólos é limitado no intervalo $[0, 1]$ para garantir a estabilidade, e a Seção 3.2.3 discute a necessidade de se limitar os zeros na mesma faixa.

3.2.2 Condição para que o filtro seja realizável

Os pólos e zeros do sistema são representados por seus raios e ângulos, com módulo limitado para garantir estabilidade. Porém, isto não é suficiente pois, além de estável, o sistema precisa ser realizável, isto é, possível de implementar. Um filtro instável não quer dizer que não é possível implementá-lo, apenas que não pode ser utilizado de forma adequada.

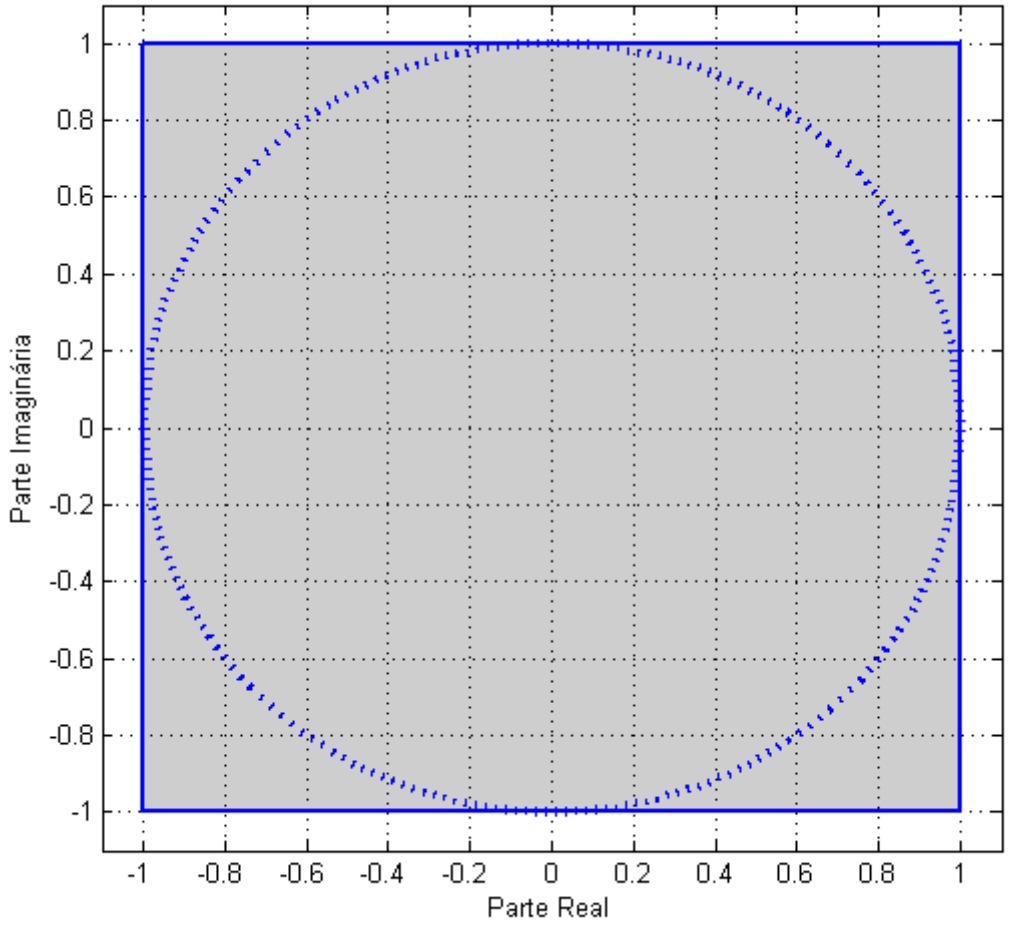


Figura 12: Espaço de busca com representação de pólos de forma retangular com limite maior
Fonte: Autoria Própria.

Para o sistema ser realizável, é necessário que todos os coeficientes sejam números reais puros, isto é, com parte imaginária nula. Para que isto ocorra, cada pólo precisa ser um número real puro, o que limita de forma absurda o espaço de busca, ou precisa ter outro pólo contraposto cujo valor seja seu conjugado. Em outras palavras, esta condição de realizabilidade só é atingida se o sistema possuir, para cada pólo complexo, um outro pólo igual ao conjugado do mesmo.

Para garantir esta condição de pólos conjugados, uma opção é penalizar as soluções que não obedeçam a este critério. Mas isto leva o espaço de busca a possuir pequeníssimas ilhas de soluções válidas, em um oceano de soluções inválidas, tornando a evolução do algoritmo extremamente difícil. A melhor alternativa é tornar implícita a existência de um pólo conjugado para cada pólo representado. Ou seja, se o indivíduo possuir 3 pólos, são adicionados mais 3

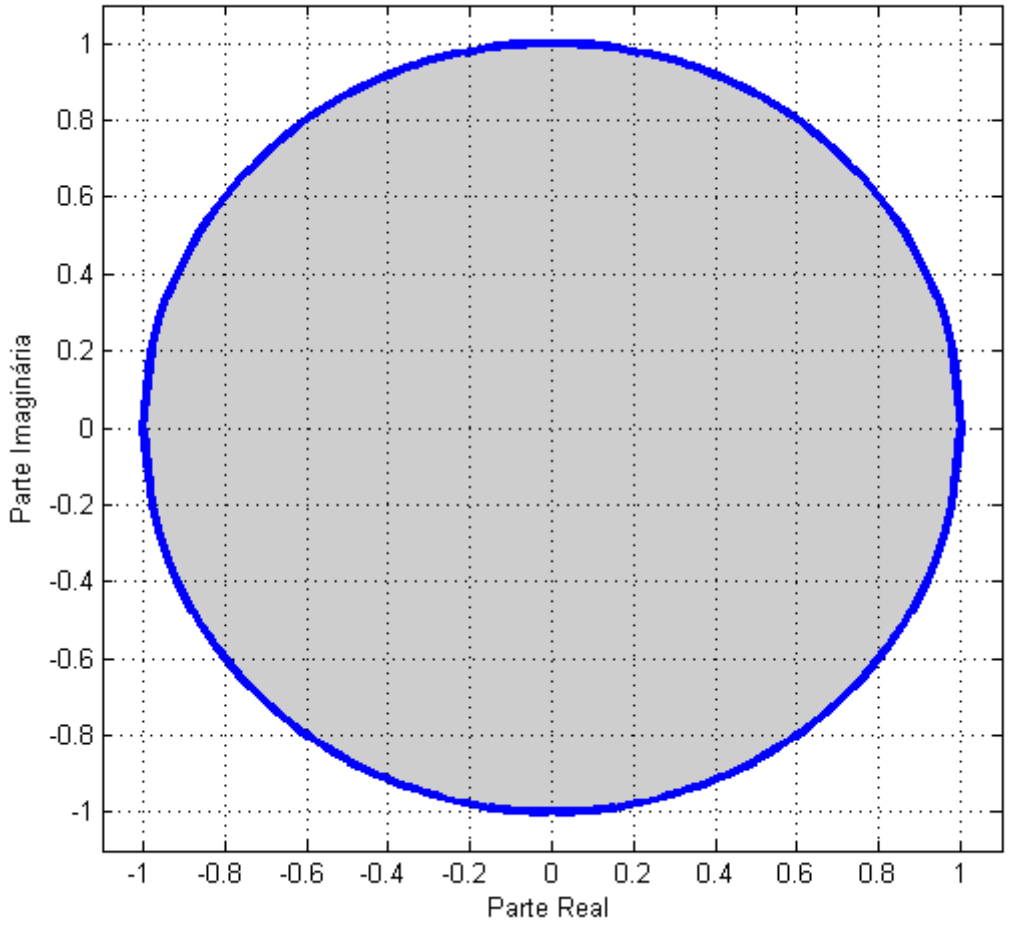


Figura 13: Espaço de busca com representação de pólos na forma polar

Fonte: Autoria Própria.

pólos, sendo estes iguais ao conjugado dos 3 primeiros. Desta forma, mais uma vez é garantido que apenas filtros adequados (estáveis e realizáveis) sejam representados pela codificação, e apenas estes.

3.2.3 Inversibilidade

Outra caracterísica desejável é que o filtro seja inversível. A principal vantagem de filtros inversíveis é que é possível desfazer sem perdas de informação as alterações feitas pelo filtro original. Esta operação é feita passando o sinal filtrado por outro filtro, que é o inverso do primeiro. Esta propriedade tem aplicações em equalizadores, formatadores de sinal, entre outros (PANAHI; VENKAT, 2009). A inversão de um filtro consiste em alternar os vetores de coeficientes A e B, ou seja, trocar o polinômio do numerador pelo do denominador, e vice-versa.

As alterações são desfeitas de forma perfeita, como demonstrado nas Equações 22 a 26, onde $y(z)$ é o sinal a ser filtrado.

$$H(z) = \frac{\sum_{k=0}^{M} b_k \cdot z^{-k}}{\sum_{k=0}^{N} a_k \cdot z^{-k}} \tag{22}$$

$$y'(z) = y(z) \cdot H(z) \tag{23}$$

$$H^{-1}(z) = \frac{\sum_{k=0}^{N} a_k \cdot z^{-k}}{\sum_{k=0}^{M} b_k \cdot z^{-k}} \tag{24}$$

$$y'(z) \cdot H^{-1}(z) = y(z) \cdot H(z) \cdot H^{-1}(z) = y(z) \cdot \frac{\sum_{k=0}^{M} b_k \cdot z^{-k}}{\sum_{k=0}^{N} a_k \cdot z^{-k}} \cdot \frac{\sum_{k=0}^{N} a_k \cdot z^{-k}}{\sum_{k=0}^{M} b_k \cdot z^{-k}} \tag{25}$$

cancelando os termos das frações, tem-se finalmente que

$$y'(z) \cdot H^{-1}(z) = y(z) \tag{26}$$

onde $y(z)$ é o sinal original, $y'(z)$ é o sinal filtrado pelo sistema original $H(z)$, cujo filtro inverso é dado por $H^{-1}(z)$.

Com esta inversão de polinômios, os zeros do sistema se tornam pólos e vice-versa. Como qualquer outro filtro, ele precisa ser estável para ser utilizado. E para que este filtro inverso seja estável, as condições são as mesmas: pólos dentro do círculo de raio unitário. Como os pólos do filtro inverso são os zeros do filtro original, basta garantir que pólos e zeros estejam dentro do círculo, que automaticamente o filtro inverso também será estável. Por esta razão, o módulo tanto de pólos quanto de zeros dos filtros que são calculados com esta codificação é limitado em 1.

3.2.4 Implementação

Apresentadas estas considerações, a representação da codificação é proposta na Figura 14.

A primeira metade dos valores diz respeito aos pólos, e a segunda metade aos zeros. Esta ordem foi adotada por simples convenção e organização. Cada par de valores Mxn e Axn é módulo e ângulo, respectivamente. Este módulo e ângulo representam dois pólos ou zeros conjugados, onde um deles terá ângulo Axn, e o outro terá ângulo -Axn, conforme explicado na

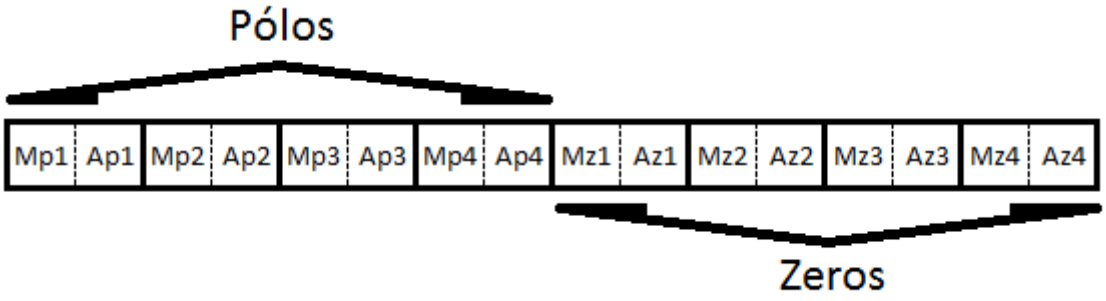


Figura 14: Diagrama da codificação proposta

Fonte: Autoria Própria.

Seção 3.2.2. Portanto, o filtro representado na Figura 14 possui 8 pólos e 8 zeros.

Os limites dos valores Mxn são, por razões já explicadas, 0 e 1. Como pólos com módulo exatamente igual a 1 implicariam em um sistema oscilatório, o intervalo deve ser aberto, portanto, $Mxn \in [0,1[$. Implementar computacionalmente um número para não ser 1 pode ser complicado, então deve-se usar um valor próximo para ser o limite superior em um intervalo fechado. Porém, isso não é crítico, já que dificilmente o valor alcançado numa otimização numérica seria exatamente 1.

Os ângulos Axn poderiam assumir valores de $-\pi$ até π. Mas como cada valor de Axn implica em dois valores simétricos de ângulos para dois pólos ou zeros diferentes, qualquer valor negativo de Axn geraria os mesmos dois pólos ou zeros. Por exemplo, um pólo de módulo 0,5 e ângulo $\frac{\pi}{4} rad$ implica em outro pólo igual ao seu conjugado, ou seja, módulo $0,5$ e ângulo $-\frac{\pi}{4} rad$. Se em vez disto o pólo representado for de 0,5 com ângulo $-\frac{\pi}{4} rad$, o conjugado é $\frac{\pi}{4} rad$, exatamente o mesmo par.

Ou seja, há uma duplicação do espaço de busca, e pode-se utilizar apenas valores não-negativos para Axn sem nenhum prejuízo em termos de soluções representáveis. Logo, $Axn \in [0,\pi]$. Por questão de padronização, este valor será normalizado entre 0 e 1, onde 1 equivale ao ângulo π, $0,5$ equivale a $\frac{\pi}{2}$, etc. Esta normalização facilita a calibração de parâmetros dos algoritmos de otimização utilizados, já que todas as variáveis ficam na mesma faixa de valores.

Portanto, tem-se que cada filtro é representado por um vetor de $2N$ posições, onde N é a ordem do filtro. Cada valor se situa entre 0 e 1. Porém, para a avaliação da qualidade do filtro, é necessária a resposta em frequência do mesmo, que não pode ser obtida pelos pólos e zeros diretamente. É necessário ,portanto, transformar esta informação nos coeficientes do filtro, para então calcular o *fitness* como apresentado na Seção 3.1.

Para se chegar na função de transferência, basta aplicar os pólos e zeros na equação

generalizada do IIR fatorada, mostrada na Equação 27.

$$H(z) = \frac{(z-a_1)\cdot(z-a_2)\cdot\ldots\cdot(z-a_N)}{(z-p_1)\cdot(z-p_2)\cdot\ldots\cdot(z-p_N)} \tag{27}$$

onde a_n são os zeros e p_n são os pólos do filtro. Efetua-se então as multiplicações, chegando-se a equações no formato $a_1z^{-1} + a_2z^{-2} + \ldots$ tanto no denominador quanto no numerador. Este formato é o mesmo da Equação 13, que é a função de transferência do filtro IIR. Desta forma, os coeficientes que multiplicam cada potência de z são os coeficientes do filtro, e pode-se partir desta função para achar a resposta em frequência e calcular o seu *fitness* através da função apresentada na Seção 3.1.

Porém, antes de se calcular o *fitness*, mais um passo é necessário. É muito difícil alterar o ganho em toda a banda quando se tem apenas os pólos e zeros de um sistema. Com os coeficientes é possível multiplicar o numerador por um número qualquer e automaticamente a resposta em frequência deste filtro será inteiramente multiplicada por este número. Esta facilidade é importante, pois torna relevante não o ganho do filtro na banda de passagem e na de rejeição, mas a diferença entre eles. Se a especificação determina que um filtro deve ter ganho de 20dB na banda de passagem e -10dB na banda de rejeição, quer dizer que a diferença entre eles deve ser de 30dB. Por exemplo, se o processo de cálculo resultar num filtro com ganhos de respectivamente 8dB e -22dB, é trivial levá-lo até a especificação.

Desta forma, o usual é especificar a banda de passagem com ganho 0dB, tanto que na maior parte dos métodos citados na Seção 2.3.2, o ganho nesta banda nem faz parte das especificações, sendo assumido como 0dB. Portanto, para facilitar a busca, a resposta do filtro deve ser normalizada, de forma que o máximo ganho seja 0dB. Basta dividir o numerador (ou multiplicar o denominador) pelo ganho máximo encontrado no filtro em toda a banda, e este objetivo é alcançado.

3.3 EQUALIZAÇÃO EM FASE POR CE

Como citado na Seção 2.3.3, a equalização em fase de um filtro significa tornar linear a sua resposta em fase, pelo menos na banda de passagem. Para isto, utiliza-se um filtro equalizador em série com o filtro sendo equalizado. Este filtro equalizador consiste de um filtro passa-tudo, com ganho constante em toda a banda. Esta característica é necessária para não alterar a resposta em módulo do filtro calculado de acordo com as especificações do projeto.

Este equalizador, porém, deve ter sua resposta em fase casada com a resposta em fase

do filtro original, de forma a trazer para a especificação a resposta em fase total, isto é, o mais linear possível, de forma a diminuir a distorção de fase. Para se alcançar esta linearidade, uma opção é minimizar a dispersão em torno do atraso de fase τ_p (NOCETI FILHO, 1998), dado pela Equação 28:

$$\tau_p = \frac{-\theta(\omega)}{\omega} \tag{28}$$

onde $\theta(\omega)$ é a resposta em fase do filtro. Como o sistema final será composto dos dois filtros concatenados, o objetivo é minimizar a dispersão de τ_p total em torno de um valor médio. Pelas propriedades de sistemas LSI tem-se que a resposta em fase resultante de dois sistemas em série é igual à soma da fase dos dois sistemas. O valor médio em torno do qual o atraso de fase está disperso é irrelevante (NOCETI FILHO, 1998). Portanto, esta dispersão é medida apenas através do desvio padrão de τ_p resultante na faixa de passagem. Logo, o erro g a ser minimizado é dado na Equação 29:

$$g = \sigma\left(\frac{-\theta_1(\omega) - \theta_2(\omega)}{\omega}\right) \tag{29}$$

onde σ é a função desvio padrão, θ_1 e θ_2 são as respostas em fase do filtro original e do equalizador respectivamente, e ω são as frequências da faixa de passagem, pois não há motivos para se linearizar a resposta na banda de rejeição.

Pode-se observar que, apesar da medição de *fitness* ser diferente, este problema é muito semelhante à otimização de filtros IIR anteriormente abordada. Afinal, o equalizador não deixa de ser um filtro deste tipo, e está sendo otimizado através de seus coeficientes. Com isto, os mesmos problemas de garantia de estabilidade durante a evolução acontecem. A abordagem apresentada neste trabalho para a otimização da resposta em módulo, através dos pólos e zeros ao invés dos coeficientes, não é vantajosa neste caso porque é necessário garantir a resposta plana em módulo, já que o equalizador não deve alterar este aspecto da resposta final.

Esta característica de planicidade é alcançada garantindo-se que os coeficientes estejam no formato mostrado na Equação 30 (DINIZ et al., 2004), que infelizmente nada garante em relação à estabilidade.

$$H(z) = \frac{a_n \cdot z^n + a_{n-1} \cdot z^{n-1} + \ \ldots \ + a_2 \cdot z^2 + a_1 \cdot z + 1}{z^n + a_1 \cdot z^{n-1} + \ \ldots \ + a_{n-2} \cdot z^2 + a_{n-1} \cdot z + a_n} \tag{30}$$

Outra vantagem de otimizar o equalizador pelos coeficientes é que o espaço de busca é reduzido, já que o vetor denominador é igual ao do numerador, mas em ordem invertida. Desta

forma, um equalizador com 4 pólos e 4 zeros, por exemplo, em vez de ser representado por 10 coeficientes (5 no numerador e 5 no denominador) será otimizado com apenas 5 coeficientes.

Por estas razões, em vez do problema da estabilidade ser abordado através de modificações na codificação, será utilizada uma penalidade (GOLDBERG, 1989) para diminuir o *fitness* das soluções instáveis, pois apenas descartar soluções instáveis torna o espaço de busca difícil de ser navegado. Isto traz outra dificuldade, pois a primeira vista não faz muito sentido quantificar a instabilidade de um filtro, já que ou ele é estável, ou não é. A solução proposta é utilizar a quantidade de pólos com módulo maior do que 1 para isto, onde quanto mais pólos causando instabilidade, pior o *fitness*. Tem-se então a penalidade p_1 dada na Equação 31:

$$p_1 = \frac{1}{1+N_p} \tag{31}$$

onde N_p é o número de pólos causando estabilidade, isto é, fora do círculo de raio unitário. Deve-se notar que esta penalidade assume valor 1 quando não há instabilidade, e diminui rapidamente a cada pólo problemático. Há de ser enfatizado que esta penalidade não diz se uma determinada solução é mais ou menos adequada. Qualquer pólo causador de instabilidade torna o equalizador totalmente inadequado. Sua finalidade é indicar se uma solução está mais distante ou mais próxima de uma solução válida, facilitando a evolução dos indivíduos ou partículas.

Como se deseja que o filtro seja inversível, é necessário garantir que os zeros do equalizador também estejam dentro do critério. Utiliza-se então a mesma fórmula para os pólos, mas desta vez utilizando o número de zeros e chamando o resultado de p_2. Tem-se assim, a função de *fitness* para a otimização do equalizador de fase na Equação 32:

$$fitness = p_1 \cdot p_2 \cdot g \tag{32}$$

onde g é a dispersão medida através da Equação 29.

Caso não seja necessária a condição de inversibilidade do filtro, apenas a penalidade p_1 deve ser aplicada. Esta não pode ser descartada, já que a estabilidade sempre será uma condição de uso do filtro IIR.

4 EXPERIMENTOS E RESULTADOS

Tanto a função de *fitness* proposta quanto a nova codificação foram avaliadas de forma comparativa com as abordagens clássicas encontradas na literatura, e com o método especialista considerado mais adequado, citado na Seção 2.3.2. Desta forma, avaliou-se não apenas o avanço alcançado em relação às técnicas utilizadas em Computação Evolucionária, mas também o desempenho em relação às ferramentas mais utilizadas pelos projetistas. Como citado no Capítulo 2, apenas filtros passa-baixas precisam ser avaliados, já que é possível transformar estes filtros em qualquer outro tipo.

Para avaliar as modificações propostas no Capítulo 3, foram escolhidos 3 diferentes especificações de filtro. Os três diferentes filtros exigem *ripple* máximo de 1dB na banda de passagem e atenuação mínima de -30dB na banda de rejeição. O primeiro possui a frequência limite da banda de passagem $\omega_p = 0,45$ e a frequência limite da banda de rejeição $\omega_r = 0,5$, com ordem 6. Este filtro possui a banda de passagem com largura próxima à da banda de rejeição, e suas especificações de banda são comumente utilizadas na literatura (DINIZ et al., 2004). Segundo o critério de Chebyschev, a ordem mínima é 5 para que este filtro alcance estes valores.

O segundo filtro possui uma banda de passagem larga em comparação com a banda de rejeição, sendo $\omega_p = 0,8$ e $\omega_r = 0,85$. A ordem necessária é 6, sendo também este o valor da ordem utilizada nos experimentos. Já o terceiro filtro possui uma banda de passagem estreita, com $\omega_p = 0,08$ e $\omega_r = 0,1$. A ordem mínima necessária é 5, mas os experimentos utilizarão ordem 4. Desta forma, espera-se avaliar o comportamento do algoritmo em condições impossíveis de serem atendidas. Portanto, o primeiro filtro tem uma ordem maior do que a necessária, o segundo tem a ordem exatamente necessária, e o terceiro tem a ordem abaixo da mínima necessária. Em outras palavras, os filtros de teste estão organizados em ordem crescente de dificuldade.

O número máximo de avaliações da função de *fitness* estipulado para os experimentos foi de 100.000. Este valor foi obtido de forma empírica através de experimentos durante o de-

senvolvimento do trabalho, e de acordo com um trabalho preliminar (OLIVEIRA et al., 2011b). Este valor é maior do que o utilizado em trabalhos de outros autores citados na Seção 2.5. Porém, deve ser frisado novamente que este trabalho considera também a estabilidade dos filtros, condição imprescindível para a utilização de um filtro, e que foi ignorada em praticamente todos os trabalhos encontrados. Este mesmo número de avaliações é utilizado em (PAN, 2011), sendo que tanto a ordem do filtro quanto o fator de *roll-off* (e portanto, a dificuldade do problema) são menores. Desta forma, por mais que o número de avaliações seja alto em comparação com o valor normalmente utilizado para o PSO e o AG, é o valor que costuma atender o critério de estabilidade e as especificações.

Os parâmetros do PSO utilizados foram os padrões utilizados na literatura $c_g = 2,0$, $c_l = 2,0$ e $w = 0.5$ (KENNEDY; EBERHART, 1995), a não ser que outros valores tenham sido explicitados. Neste ponto do trabalho, nenhum esforço foi realizado no sentido de sintonizar estes parâmetros para atingir melhores resultados. A Seção 4.4 mostra os experimentos e resultados obtidos especificamente com este objetivo.

4.1 CODIFICAÇÃO

Primeiramente, é necessário avaliar se a codificação proposta apresenta bons resultados. Para cada codificação (por coeficientes e por pólos/zeros), foram utilizadas as 3 especificações diferentes de filtros, como mostrado na Tabela 1. Foi utilizado o algoritmo PSO com os parâmetros padrão citados anteriormente, onde $N = 1000$ é o número de iterações e $M = 100$ o tamanho do enxame. Cada experimento diferente foi rodado 30 vezes. A função de *fitness* utilizada foi a MSE pura, com as penalidades para instabilidade propostas na Seção 3.3, na Equação 31.

Exp. N^o	Filtro N^o	Codificação	Ordem	ω_p	ω_r	A_p [dB]	A_r[dB]
1	1	Coeficientes	6	0,45	0,5	1	-30
2	1	Pólos e Zeros	6	0,45	0,5	1	-30
3	2	Coeficientes	6	0,8	0,85	1	-30
4	2	Pólos e Zeros	6	0,8	0,85	1	-30
5	3	Coeficientes	4	0,08	0,1	1	-30
6	3	Pólos e Zeros	4	0,08	0,1	1	-30

Tabela 1: Experimentos de codificação

4.1.1 Resultados

Os resultados dos experimentos realizados são mostrados na Tabela 2. A coluna referente ao *fitness* máximo mostra o melhor resultado encontrado em todas as 30 rodadas, *fitness* mínimo é o pior resultado encontrado ao final das rodadas, e *fitness* médio é o resultado médio encontrado nos testes, mostrado juntamente com o desvio padrão. A coluna ρ mostra a probabilidade de que as médias sejam as mesmas pelo teste de t-Student. Ou seja, expressa a chance de que a diferença observada na média do *fitness* obtido seja puramente aleatória e não causada pela codificação diferente. A coluna "Diferença"mostra o intervalo em que se encontra a diferença no *fitness* médio obtido pela mudança de codificação, com 5% de certeza.

Exp. N^o	*Fitness* máximo	*Fitness* mínimo	*Fitness* médio	ρ	Diferença
1	0,7904	0,3114	$0{,}5769 \pm 0{,}1449$	8,45e-14	(0,3101; 0,4237)
2	0,9713	0,8434	$0{,}9438 \pm 0{,}0299$		
3	0,5588	0,1564	$0{,}2182 \pm 0{,}1066$	1,36e-9	(0,3759; 0,6065)
4	0,8850	0,1631	$0{,}7094 \pm 0{,}2843$		
5	0,7371	0,0689	$0{,}3494 \pm 0{,}1586$	3,89e-8	(0,2222; 0,3923)
6	0,8841	0,2010	$0{,}7775 \pm 0{,}2062$		

Tabela 2: Resultados dos experimentos de codificação

A codificação por pólos e zeros apresentou um grande ganho na qualidade das soluções. A média do *fitness* das soluções encontradas foi bem maior do que as obtidas pela codificação por coeficientes. Portanto, é seguro afirmar que a representação dos filtros por pólos e zeros obtem resultados melhores que a representação por coeficientes. A magnitude deste ganho variou para cada filtro, mas todos estão na mesma ordem de grandeza do próprio *fitness*.

A Figura 15 mostra a evolução média da melhor partícula nos experimentos 1 e 2. É possível perceber que quando representadas por coeficientes, as soluções tendem a convergir mais rápido, mas a qualidade dos resultados obtidos é bem mais baixa se houver tempo suficiente para evoluírem as soluções representadas por pólos e zeros. O ponto de cruzamento dos *fitness* é próximo da iteração 200.

Na Figura 16 percebe-se que o algoritmo com a representação por coeficientes convergiu muito rapidamente, mas para soluções de qualidade muito baixa. Já com os pólos e zeros, o algoritmo sequer chegou a convergir dentro do limite estipulado de iterações, pois nota-se uma inclinação na curva de *fitness* que sugere um potencial ainda considerável de melhora da solução. Ainda assim, o contraste na qualidade entre as soluções é visível.

Nos experimentos 5 e 6, ambas as codificações parecem ainda ter um certo potencial de melhora ao final das iterações, mas com pólos e zeros o desempenho é muito melhor. Este é um

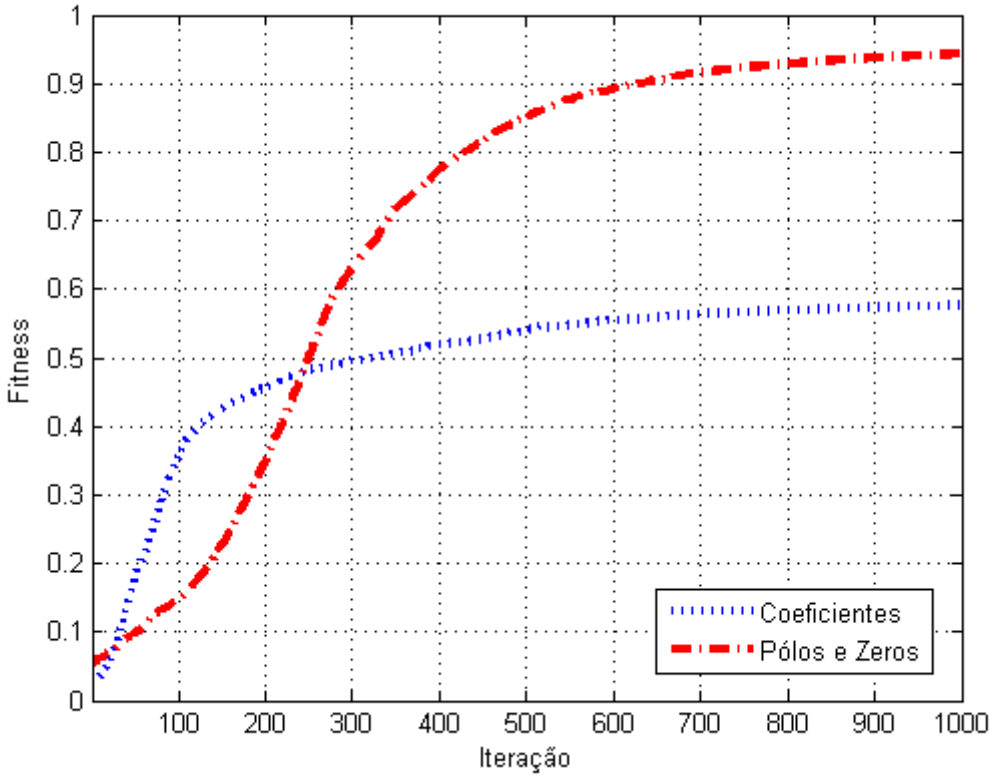


Figura 15: Média da evolução do fitness máximo para os experimentos 1 e 2

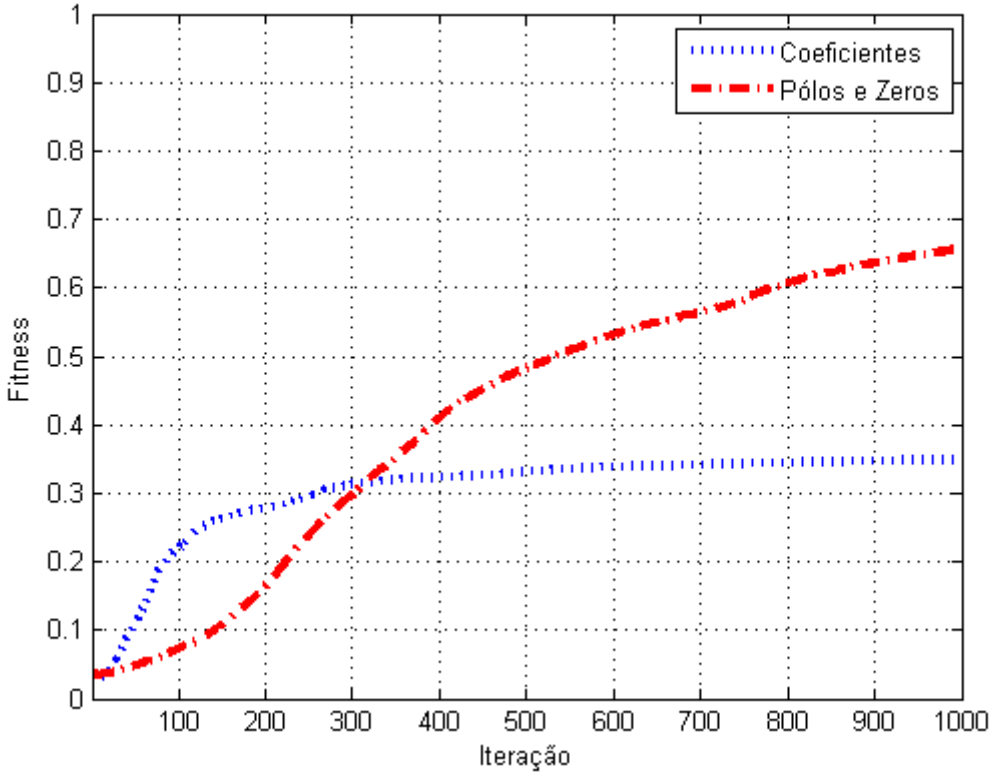


Figura 16: Média da evolução do fitness máximo para os experimentos 3 e 4

filtro mais difícil do que os experimentos anteriores, pois requer um *roll-off* muito acentuado. Isto explica a necessidade de um tempo maior para a convergência, mas a Figura 17 deixa implícito que a codificação por coeficientes não chegará perto de atingir o resultado obtido com pólos e zeros, mesmo com um grande número de iterações.

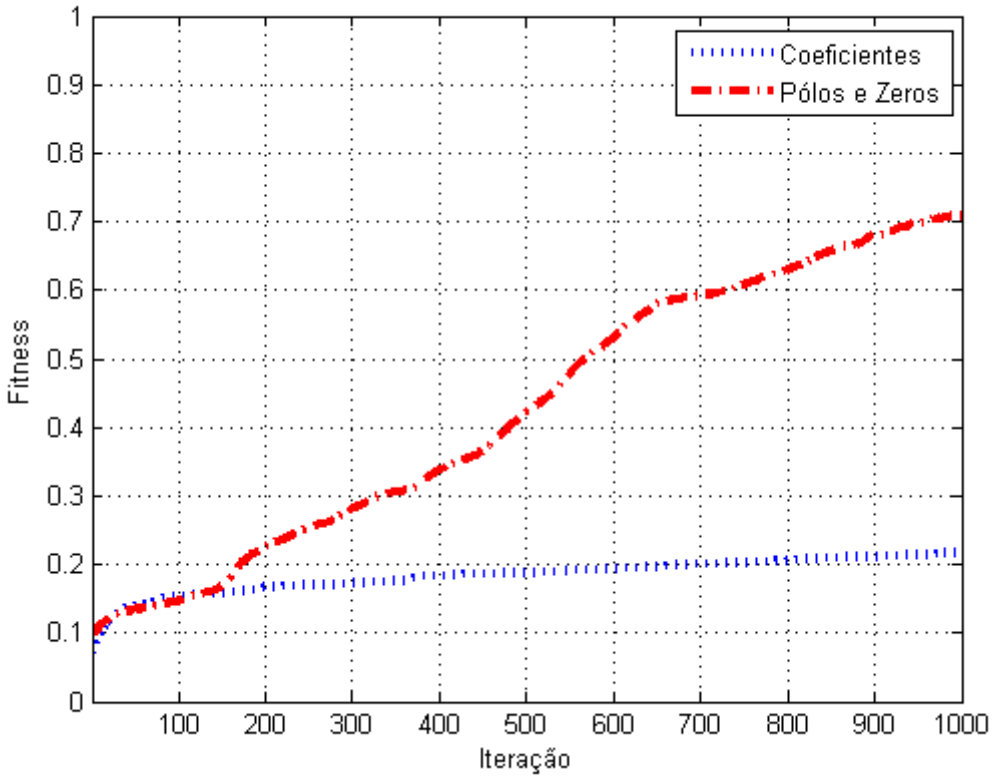


Figura 17: Média da evolução do fitness máximo para os experimentos 5 e 6

Para comparar visualmente a resposta em frequência dos filtros obtidos pelos algoritmos, é inviável analisar o resultado de cada execução dos experimentos. Porém, não faz sentido calcular a média da resposta em frequência dos filtros, como se faz com as curvas de evolução, porque esta resposta média não representa um filtro obtido pelo algoritmo, e sequer resulta numa resposta garantidamente possível de ser obtida. Além disto, uma operação de média tende a atenuar ondulações na resposta, gerando uma falsa qualidade que não foi alcançada nos filtros calculados pelo algoritmo. Comparar dois experimentos utilizando o melhor filtro obtido por cada um também não é justo. Afinal, quanto maior o número de execuções, maior a chance de que aconteça uma anomalia que não representa um resultado tipicamente alcançado pelo algoritmo. O mesmo argumento se aplica à utilização do pior filtro obtido.

A alternativa proposta é utilizar como representante do desempenho médio do algoritmo o filtro cujo *fitness* mais se aproximou do *fitness* médio obtido em um dado experimento. Desta forma, pode-se comparar a resposta média dos filtros obtidos através de um filtro real, e

não de uma resposta obtida através da média da resposta dos filtros, que não representaria um resultado efetivamente alcançado pelo algoritmo. Este filtro médio também é a representação mais acurada da qualidade geralmente obtida por um algoritmo qualquer.

Para os experimentos 1 e 2, a resposta destes filtros médios estão mostradas na Figura 18.

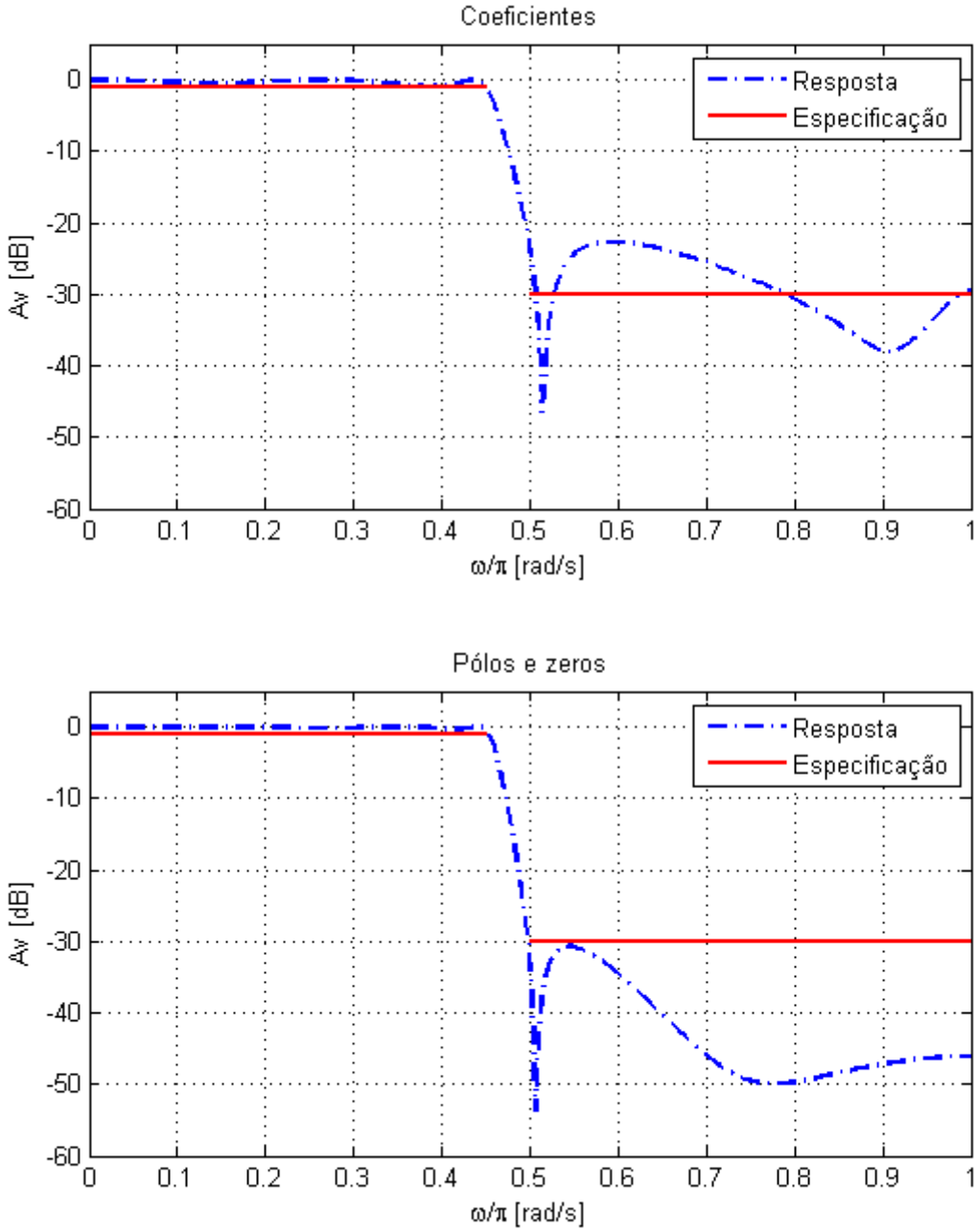


Figura 18: Filtros obtidos pelas duas abordagens no experimento 1

O filtro obtido com a codificação por coeficientes teve um bom um desempenho na banda de passagem, totalmente dentro da especificação. Porém, na banda de rejeição, há uma

faixa grande de frequências em que o ganho está acima do especificado, chegando ao máximo de quase -20dB. Enquanto isto, o filtro conseguido com a representação por pólos e zeros ficou dentro da especificação em ambas as faixas, e apresentou *ripple* menor na banda de passagem, como é possível observar na Figura 18.

Na Figura19 são comparados os filtros médios obtidos no experimento 4. Ambas as abordagens de codificação obtiveram uma boa resposta na banda de passagem, embora a codificação por coeficientes tenha apresentado um *ripple* ligeiramente maior. Porém, na banda de rejeição, as duas falharam na obediência às especificações. Com coeficientes, o ganho nesta banda chegou a -20dB, enquanto não passou de -25dB utilizando pólos e zeros.

Os filtros para comparação visual do experimento 9 foram escolhidos pelo mesmo método descrito anteriormente. A Figura 20 mostra a resposta em frequência de cada um. Neste caso, o resultado obtido utilizando coeficientes ficou bem fora da especificação. Na banda de passagem, apenas algumas frequências tem ganho acima de -1dB, e a banda de rejeição está praticamente inteira na faixa de -20dB, em vez de abaixo de -30dB. Já o filtro calculado com pólos e zeros está praticamente inteiro dentro da especificação. Há apenas um ponto, perto de $\omega = 0.15\pi$, que a resposta na banda de rejeição parece tocar a linha de -30dB, mas é visualmente nítido que se trata de uma diferença muito pequena.

4.1.2 Discussão

Em todos os experimentos realizados, a codificação por pólos e zeros obteve um resultado melhor do que a codificação por coeficientes. Mesmo quando as duas abordagens apresentaram resultados que obedeciam às especificações, a qualidade das soluções em termos de ondulação da resposta foi melhor quando a codificação proposta neste trabalho foi utilizada.

Foi observado que esta nova codificação tem um a evolução um pouco mais lenta, o que levou à convergência mais tardiamente, melhorando o resultado final. Portanto, esta nova abordagem será utilizada no restante do trabalho.

4.2 *FITNESS*

Tendo sido escolhida a melhor codificação, é necessário comparar o ganho de desempenho dado pela função de *fitness* proposta. Porém, não faz sentido comparar valores brutos de *fitness*, já que estes são calculados de forma diferente. Por isto, as abordagens foram comparadas pelo número de vezes que o algoritmo obteve um filtro dentro das especificações, em um determinado número de execuções. As especificações dos filtros são as mesmas utilizadas na

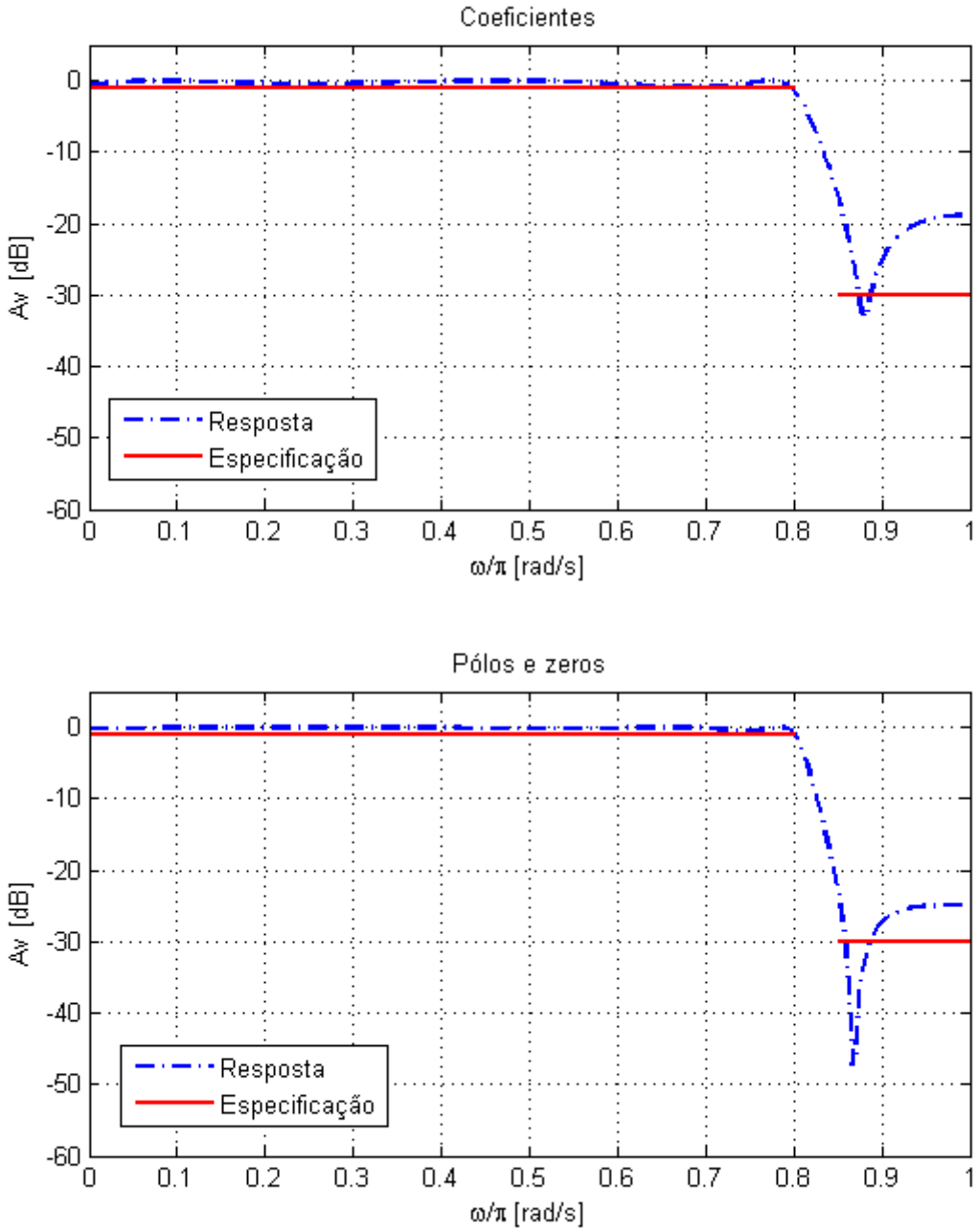


Figura 19: Filtros obtidos pelas duas abordagens no experimento 4

Tabela 1. As configurações utilizadas são mostradas na Tabela 3. Cada experimento foi rodado 30 vezes.

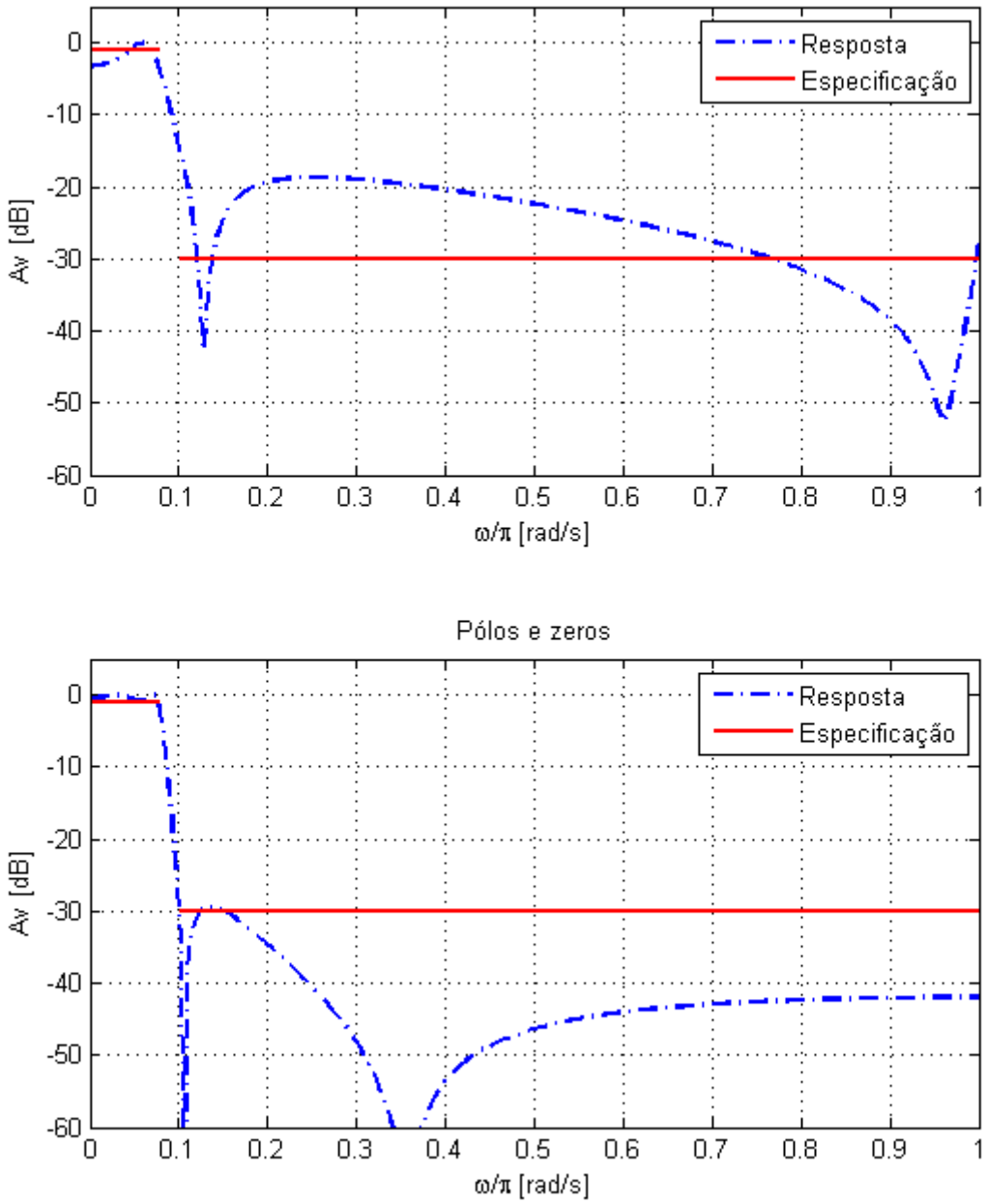


Figura 20: Filtros obtidos pelas duas abordagens no experimento 9

Exp. N^o	Filtro N^o	*Fitness*	Ordem	ω_p	ω_r	A_p [dB]	A_r[dB]
7	1	MSE	6	0,45	0,5	1	-30
8	1	Proposta	6	0,45	0,5	1	-30
9	2	MSE	6	0,8	0,85	1	-30
10	2	Proposta	6	0,8	0,85	1	-30
11	3	MSE	4	0,08	0,1	1	-30
12	3	Proposta	4	0,08	0,1	1	-30

Tabela 3: Experimentos de *fitness*

4.2.1 Resultados

Os resultados dos experimentos com a função de *fitness* tradicional (MSE) são mostrados na Tabela 4. A coluna MSE mostra o Erro Quadrático Médio encontrado. Deve ser destacado que o *fitness* dos filtros obtidos pela MSE está normalizado, mas esta foi a única medida de qualidade utilizada. A última coluna mostra em qual porcentagem das rodadas foi encontrado um filtro dentro das especificações.

Exp. N^o	*Fitness* máximo	*Fitness* mínimo	*Fitness* médio	MSE	Taxa de acerto
7	0,9844	0,8585	$0{,}9636 \pm 0{,}0300$	0,9636	0,0%
8	0,9931	0,9814	$0{,}9886 \pm 0{,}0036$	0,9608	100,0%
9	0,9548	0,5888	$0{,}7762 \pm 0{,}1088$	0,7762	0,0%
10	0,9828	0,5956	$0{,}9284 \pm 0{,}0947$	0,7565	66,7%
11	0,9564	0,5006	$0{,}9083 \pm 0{,}0847$	0,9083	0,0%
12	0,9501	0,2995	$0{,}8054 \pm 0{,}2479$	0,7884	0,0%

Tabela 4: Resultados dos experimentos de *fitness*

A nova função de *fitness* aumentou a taxa de acerto dos filtros 1 e 2 para 100% e 66,7%, respectivamente. Como o filtro 3 (experimentos 11 e 12) não pode ser atingido com a ordem dada, pode-se afirmar que a taxa de acerto melhorou em todos os casos onde este aumento era possível.

Os experimentos 7 e 8 fornecem um ótimo exemplo de como a hipótese suposta no Capítulo 3 parece ser verdadeira: os filtros obtidos possuem MSE similar - portanto, igualmente distantes do filtro ideal - quando a função de *fitness* utilizada é a proposta, mas a taxa de acerto é maior. Ou seja, o cálculo de *fitness* utilizando apenas a MSE leva o algoritmo a filtros numericamente próximos do filtro ideal, mas que não atendem às especificações dadas pelo projetista.

Nos experimentos 9 e 10, a adição dos termos propostos no cálculo do *fitness* conseguiu melhorar ligeiramente a própria MSE, além da taxa de acerto. Em outras palavras, as modificações feitas possibilitaram o alcance de regiões melhores do espaço de busca antes intocadas por alguma razão, como a topografia deste espaço.

Nos experimentos 11 e 12, a MSE dos filtros obtidos pelo método proposto foi menor, e a taxa de acerto não mudou. Portanto, neste caso será necessária uma análise gráfica da resposta em frequência, já que MSE menor não quer dizer necessariamente que um filtro é melhor, e a taxa de acerto neste caso não tem como ser diferente de zero. No experimento 12, nota-se um desvio padrão bem mais elevado que o dos outros experimentos, e o valor mínimo de *fitness* também é mais baixo. Isto indica que em algumas rodadas o algoritmo travou em uma região

de soluções ruins e não conseguiu explorar melhor o espaço de busca.

Para comparar as curvas de evolução, foram colocadas no mesmo gráfico as curvas cujas especificações de filtro fossem as mesmas, mudando apenas o método de cálculo de *fitness* utilizado. Desta forma, as Figuras 21, 22 e 23 mostram as curvas de *fitness* médio obtidas nos experimentos para determinar a melhor função de *fitness* respectivamente nos experimentos 7 e 8, 9 e 10, e 11 e 12.

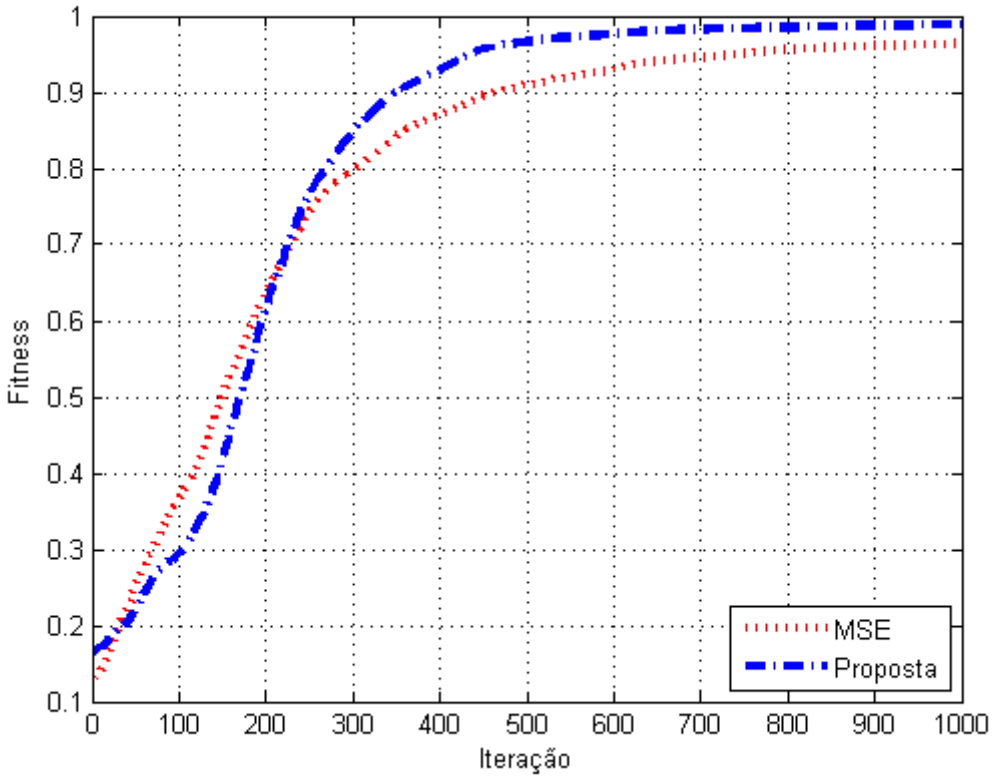


Figura 21: Média da evolução do fitness máximo para os experimentos 7 e 8

Na Figura 21, é possível observar que as duas abordagens convergiram de forma semelhante, embora o método proposto tenha alcançado um *fitness* ligeiramente mais elevado. Em geral, não é desejável que a convergência de um algoritmo de Computação Evolucionária seja rápida, pois isto pode levar a evolução a ficar encalhada em mínimos ou máximos locais. Mas neste caso, perto da iteração 500 o resultado já é bem próximo do máximo.

A evolução média dos experimentos 9 e 10 é mostrada na Figura 22. Neles, a evolução começou de forma similar, mas próximo da iteração 400, a abordagem pela MSE diminui o crescimento, enquanto o método proposto continua em um ritmo maior. O resultado final encontrado pelo método proposto possui *fitness* mais elevado.

A Figura 23 mostra a evolução média obtida nos experimentos 11 e 12. Neste caso,

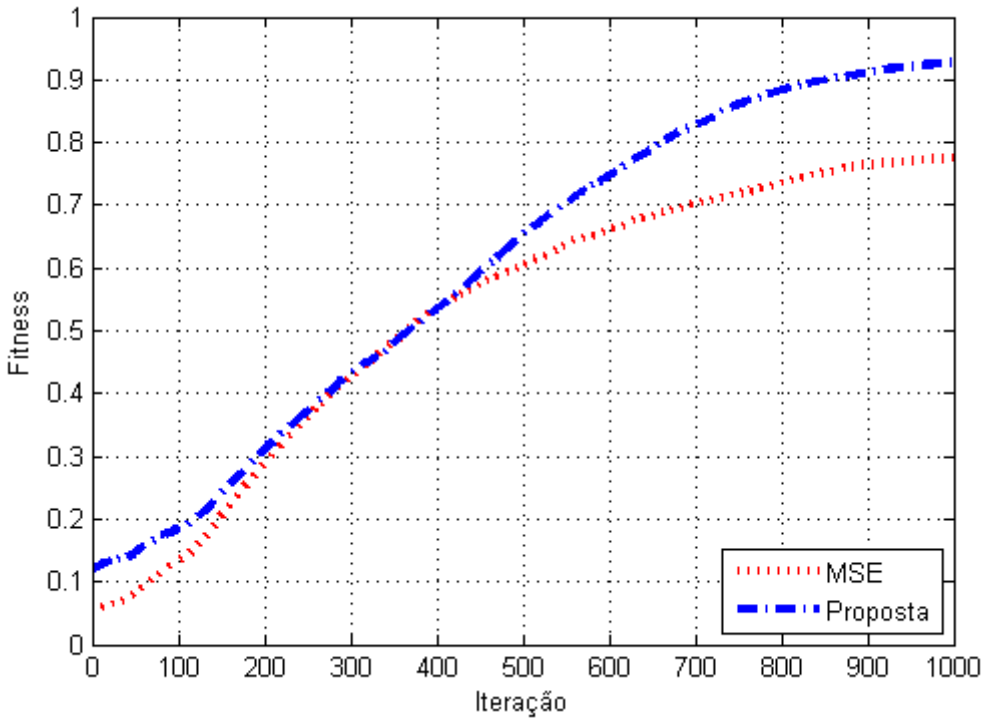


Figura 22: Média da evolução do fitness máximo para os experimentos 9 e 10

o *fitness* encontrado pelo método proposto foi menor durante toda a evolução. Esta diferença vai diminuindo ao longo das iterações, mas as curvaturas não sugerem que em algum ponto ocorrerá o cruzamento.

Por mais que não faça sentido comparar os valores brutos de *fitness*, este comportamento foi diferente dos outros experimentos, em que a MSE alcançou resultados numericamente inferiores aos do método proposto. Porém, existe uma causa bem definida para este fenômeno. Como citado no início deste capítulo, as especificações deste filtro são impossíveis de serem atingidas. Como a nova função de *fitness* penaliza soluções que não atendam às especificações, o valor atingido não tem como ser muito elevado.

A Figura 24 mostra o filtro cujo *fitness* é o mais próximo da média obtida pelo experimento, como discutido na Seção 4.1. Nela, verifica-se o que foi obervado na evolução dos experimentos retratada na Figura 21: ambas as abordagens conseguiram bons resultados, mas a função proposta foi ligeiramente melhor. A banda de passagem está praticamente igual. Já a banda de rejeição da MSE é ligeiramente pior, fora da especificação apenas na borda.

A diferença de qualidade entre filtros com MSE similar mas obtidos com métodos diferentes é fácil de verificar na Figura 25. Aqui tem-se novamente os fitros médios, ou seja, com *fitness* mais próximo da média. Estes valores são muito próximos, como mostrado na

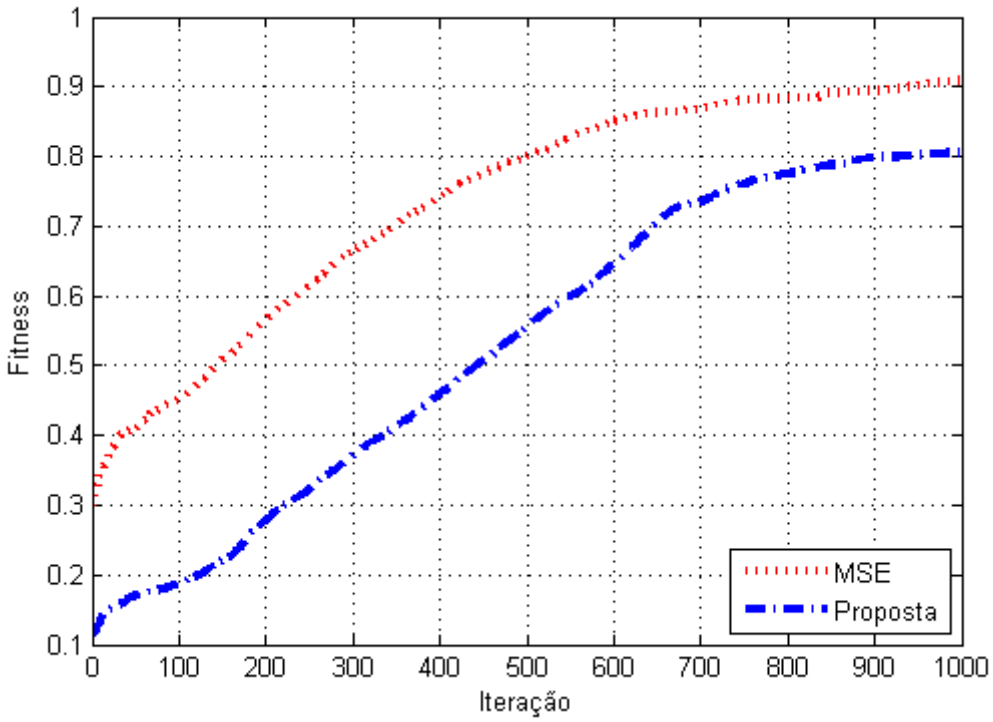


Figura 23: Média da evolução do fitness máximo para os experimentos 11 e 12

Tabela 4. Ainda assim, a diferença de qualidade entre os dois filtros mostrados nesta Figura 25 é grande. O filtro obtido pela MSE desobedece tanto as especificações de banda de passagem quanto de banda de rejeição, sendo esta última de forma acentuada. A banda de passagem está fora dos critérios apenas na borda. Já o filtro obtido pelo método proposto obedece a especificação na faixa de passagem e na banda de rejeição.

A Figura 26 mostra o caso do filtro cuja especificação não pôde ser atingida. A resposta de ambos parece similar, mas uma análise mais atenta mostra que a borda da banda de rejeição do filtro obtido apenas pela MSE chega a atingir -20dB, enquanto pelo método proposto o filtro obedece às especificações na borda, obtendo um ganho acima do limite apenas em frequências um pouco mais altas. Ainda assim, esse ganho não chega a estar tão acima do máximo exigido quanto no filtro do experimento 11. A banda de passagem de ambos também está similar. Ambos desobedecem às especificações apenas na região próxima à borda. Houve um equilíbrio na busca entre qual das bandas priorizar, já que, nos dois casos, as duas bandas foram violadas, mas de forma pouco acentuada. Um resultado possível era que os algoritmos priorizassem uma das bandas, atendendo às especificações, em detrimento da qualidade da resposta na outra, porém isto não ocorreu.

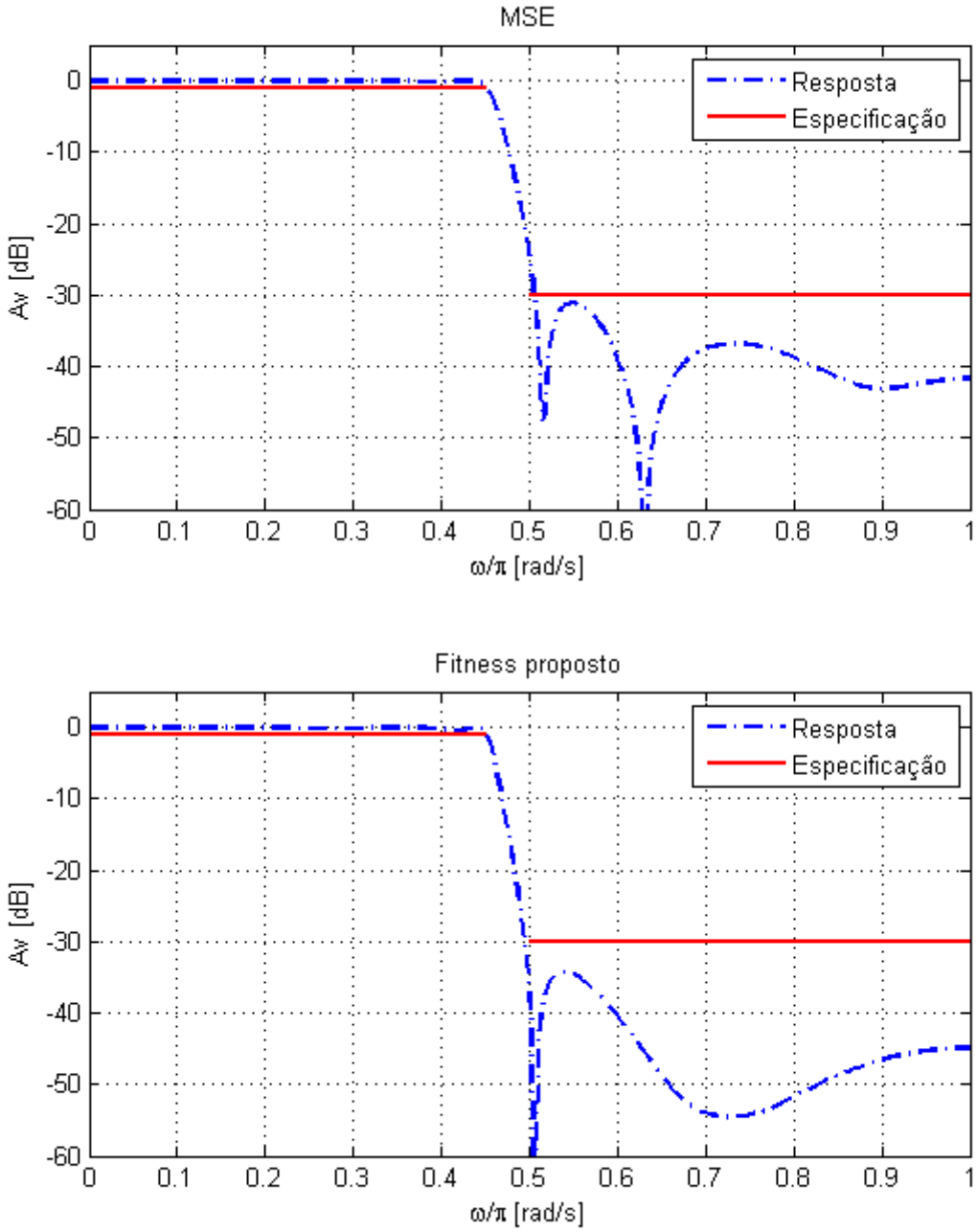


Figura 24: Filtros obtidos nos experimentos 7 e 8

4.2.2 Discussão

Em todos os experimentos realizados, a função de *fitness* proposta obteve resultados melhores do que a função que utiliza exclusivamente a MSE. O próprio erro quadrático médio foi inferior, mostrando que o método proposto também busca uma resposta de filtro ideal.

A nova abordagem também conseguiu melhores taxas de obediência às especificações.

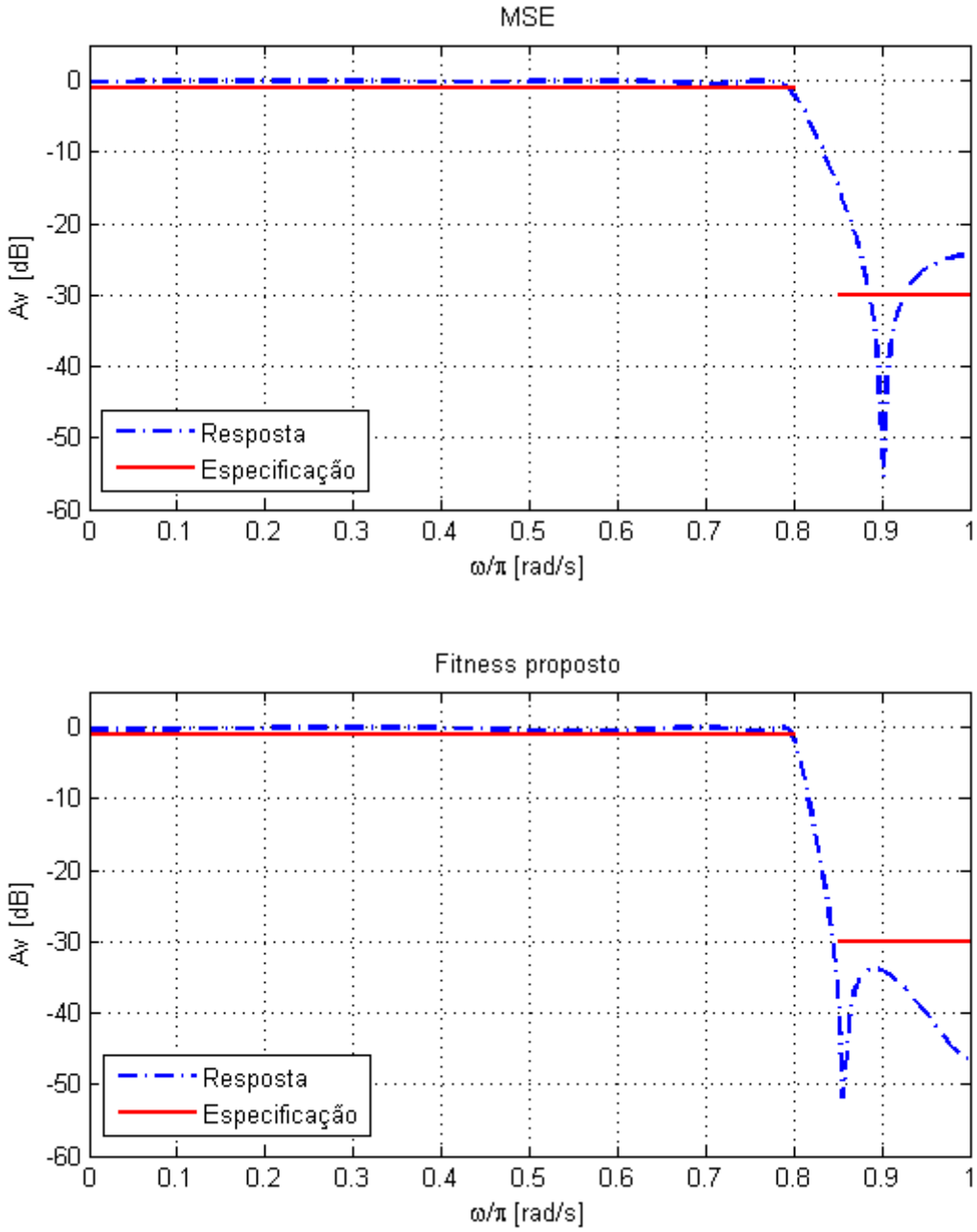


Figura 25: Filtros obtidos nos experimentos 9 e 10

Em dois dos experimentos, quando a MSE não conseguiu gerar filtros obedientes à especificação nenhuma vez, o novo método conseguiu taxas de sucesso de 100% e 67%.

Como esta nova abordagem conseguiu melhor obediência à especificação e maior proximidade com a resposta de um filtro ideal, alcançando melhores resultados mais rapidamente, pode-se considerá-la melhor do que a MSE. Portanto, esta nova abordagem será utilizada em todo o restante do trabalho.

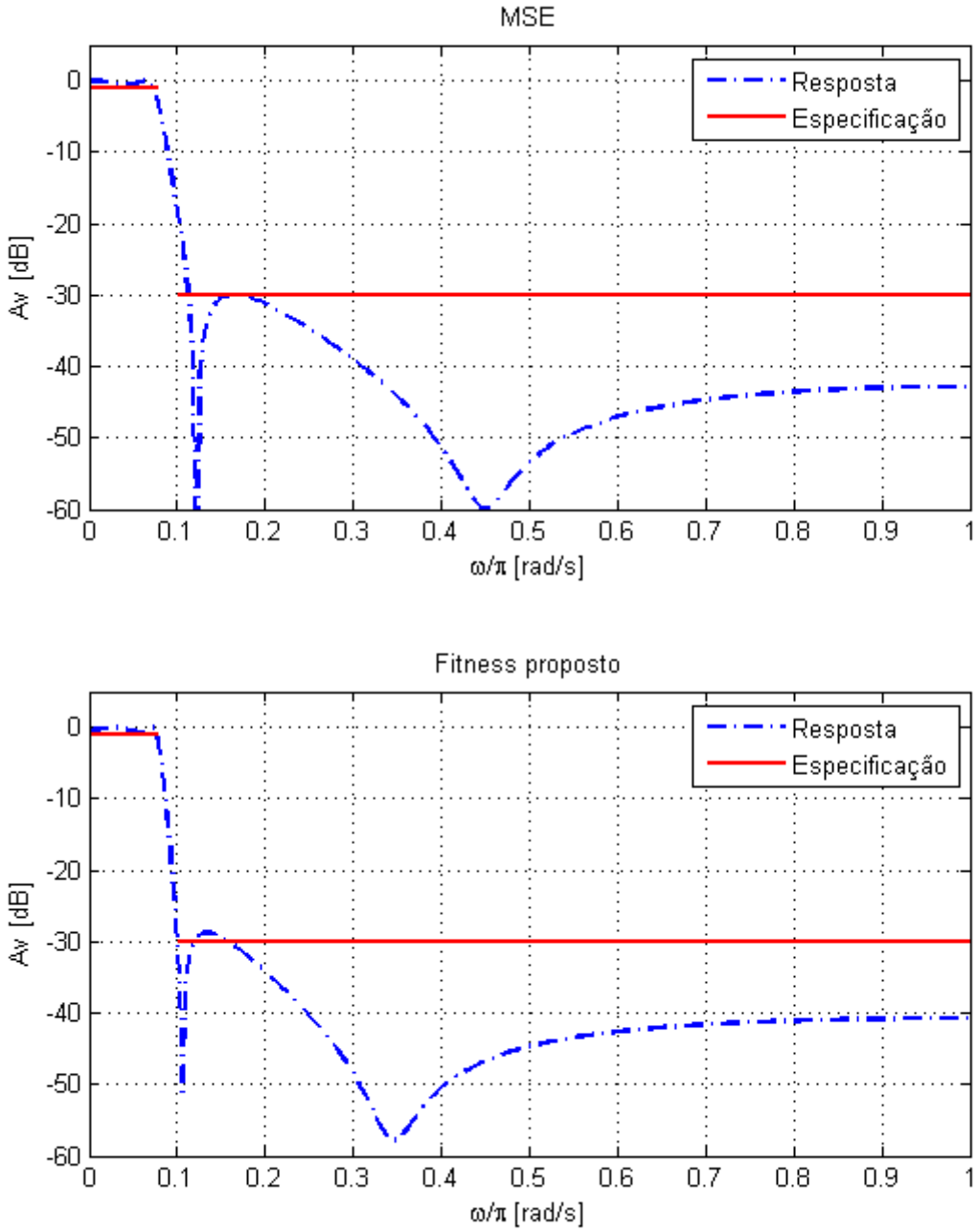


Figura 26: Filtros obtidos nos experimentos 11 e 12

4.3 ALGORITMOS DE CE

Tendo sido definidas a função de *fitness* e a codificação, o próximo passo é comparar o desempenho de diferentes algoritmos de computação evolucionária neste problema. Os algoritmos escolhidos foram o AG e o PSO, por serem os principais algoritmos do gênero utilizados na literatura. As Tabelas 5 e 6 mostram os experimentos realizados com cada um. O método

de seleção utilizado no AG foi o torneio de tamanho 5, e o método de *crossover* foi o uniforme. Este método de *crossover* foi escolhido por obter melhores resultados para problemas com codificação binária (PICEK; GOLUB, 2010). A codificação utilizada foi a binária, com 16 bits para cada variável. Isto gera um espaço de busca de 2^{192} posições para os dois primeiros filtros, e de 2^{128} para o terceiro filtro. O tamanho da população utilizado para o AG foi 500 e o número gerações igual a 200, determinados através de experimentos empíricos. Estes valores podem ser considerados baixos, mas quando se compara o desempenho de algorimos evolucionários, o número de avaliações utilizado por cada um deve ser o mesmo. Caso não haja modificações com busca local, este número de avaliações é dado no PSO por $N \times M$, e no AG por $popsize \times maxgen$.

Nos experimentos até este ponto, apenas os parâmetros padrão do PSO foram utilizados. Como os próximos experimentos comparam 2 algoritmos diferentes, é possível que o conjunto padrão torne o desempenho dos mesmos muito abaixo do seu potencial. Para evitar esse cenário, foram utilizados 3 conjuntos de parâmetros diferentes para cada algoritmo. No PSO, manteve-se $w = 0,5$, e variou-se c_g e c_l 10% para mais e para menos, como mostrado na Tabela 5. No AG, foram combinados os valores usuais utilizados na literatura, formando os conjuntos da Tabela 6.

Exp. N^o	Ordem	ω_p	ω_r	A_p [dB]	A_r[dB]	N	M	c_g	c_l	w
13	6	0,45	0,5	1	-30	100	1000	1,8	1,8	0,5
14	6	0,45	0,5	1	-30	100	1000	2,0	2,0	0,5
15	6	0,45	0,5	1	-30	100	1000	2,2	2,2	0,5
16	6	0,8	0,85	1	-30	100	1000	1,8	1,8	0,5
17	6	0,8	0,85	1	-30	100	1000	2,0	2,0	0,5
18	6	0,8	0,85	1	-30	100	1000	2,2	2,2	0,5
19	4	0,08	0,1	1	-30	100	1000	1,8	1,8	0,5
20	4	0,08	0,1	1	-30	100	1000	2,0	2,0	0,5
21	4	0,08	0,1	1	-30	100	1000	2,2	2,2	0,5

Tabela 5: Experimentos com o algoritmo PSO

4.3.1 Resultados

Os resultados obtidos pelo PSO e pelo AG estão mostrados respectivamente nas Tabelas 7 e 8. É possível verificar que o PSO obteve resultados melhores em todos os experimentos, tanto na média quanto nos melhores filtros obtidos nas 20 rodadas.

As Figuras 27, 28 e 29 mostram as curvas de evolução obtidas pelos dois algoritmos respectivamente em cada filtro. Para cada algoritmo, foram escolhidas as duas configurações que obtiveram o melhor desempenho dentre as três utilizadas. Como a Seção 4.4 efetua o ajuste

Exp. N^o	Ordem	ω_p	ω_r	A_p [dB]	A_r[dB]	*popsize*	*maxgen*	*pmut*	*pcross*
22	6	0,45	0,5	1	-30	500	200	0,01	0,7
23	6	0,45	0,5	1	-30	500	200	0,03	0,7
24	6	0,45	0,5	1	-30	500	200	0,01	0,8
25	6	0,8	0,85	1	-30	500	200	0,01	0,7
26	6	0,8	0,85	1	-30	500	200	0,03	0,7
27	6	0,8	0,85	1	-30	500	200	0,01	0,8
28	4	0,08	0,1	1	-30	500	200	0,01	0,7
29	4	0,08	0,1	1	-30	500	200	0,03	0,7
30	4	0,08	0,1	1	-30	500	200	0,01	0,8

Tabela 6: Experimentos com o algoritmo AG

Exp. N^o	*Fitness* máximo	*Fitness* mínimo	*Fitness* médio
13	0,9882	0,7687	0,9321 $\pm$ 0,0723
14	0,9884	0,9442	0,9800 $\pm$ 0,0111
15	0,9868	0,9633	0,9802 $\pm$ 0,0058
16	0,9552	0,2152	0,6230 $\pm$ 0,2670
17	0,9789	0,4413	0,8687 $\pm$ 0,1677
18	0,9858	0,9159	0,9560 $\pm$ 0,0189
19	0,9136	0,1353	0,6932 $\pm$ 0,2564
20	0,9147	0,1483	0,5844 $\pm$ 0,3340
21	0,9136	0,2034	0,8127 $\pm$ 0,2140

Tabela 7: Resultados dos experimentos com o algoritmo PSO

de parâmetros do algoritmo escolhido e o uso de todas as curvas na montagem dos gráficos deixaria a figura poluída, a pior configuração foi descartada.

Como o número de gerações do AG é diferente do número de iterações utilizado no PSO, para visualizar as duas curvas no mesmo gráfico elas foram normalizadas. Desta forma, o eixo horizontal está marcando a porcentagem concluída das iterações/gerações de cada algoritmo.

Exp. N^o	*Fitness* máximo	*Fitness* mínimo	*Fitness* médio
22	0,9447	0,2263	0,7260 $\pm$ 0,1488
23	0,9707	0,7475	0,8429 $\pm$ 0,0750
24	0,9538	0,6054	0,7979 $\pm$ 0,0902
25	0,8436	0,2584	0,5035 $\pm$ 0,2020
26	0,9039	0,2915	0,7919 $\pm$ 0,1304
27	0,8118	0,2462	0,4178 $\pm$ 0,1932
28	0,9746	0,3304	0,6076 $\pm$ 0,2483
29	0,9872	0,3423	0,8804 $\pm$ 0,1850
30	0,9513	0,2959	0,5513 $\pm$ 0,2426

Tabela 8: Resultados dos experimentos com o algoritmo AG

A Figura 27 mostra que o AG convergiu mais rapidamente que o PSO, e desacelerou a velocidade de melhoria do *fitness*. Esta velocidade pode ser uma das causas para a pior qualidade da solução final.

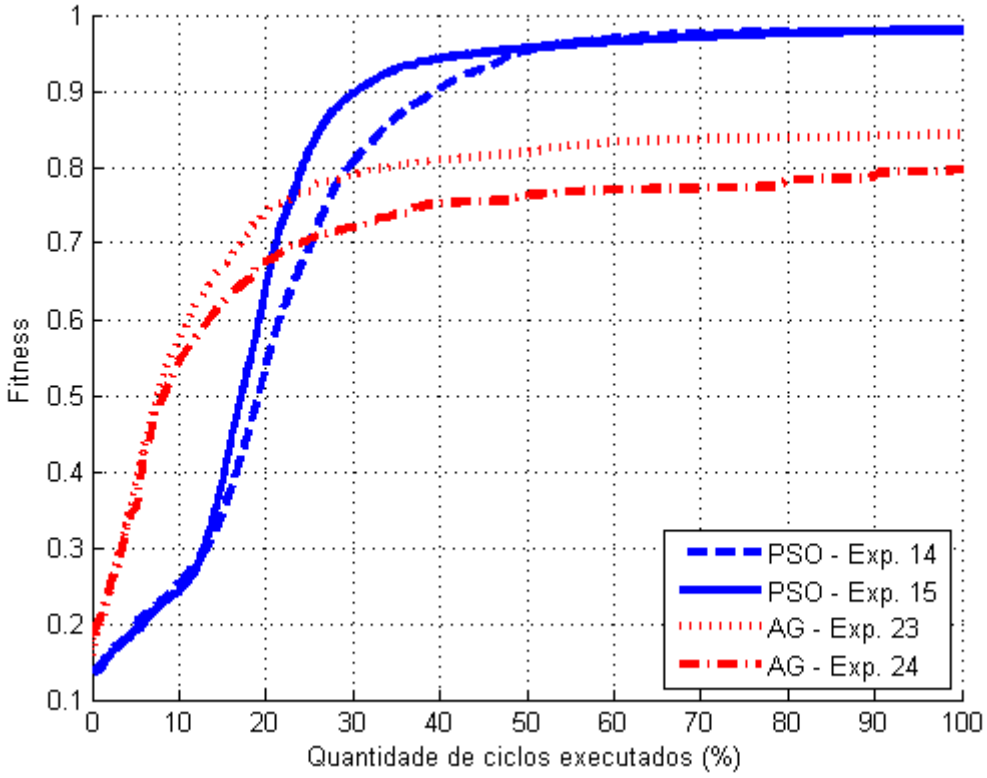


Figura 27: Média da evolução do fitness máximo para o filtro 1

Na Figura 28 tem-se a curva de evolução dos algoritmos para o filtro 2. Aqui novamente as curvas do AG mostram uma rápida convergência seguida de um melhoramento contínuo mas muito lento na qualidade das soluções. Enquanto isso, o PSO convergiu bem lentamente, chegando a um *fitness* final mais elevado. Porém, as curvas do PSO sugerem que ainda há possibilidade de evolução.

A Figura 29 mostra as curvas obtidas nos experimentos com o filtro 3. Outra vez o AG apresentou uma convergência rápida, seguida de uma melhora lenta e constante. Enquanto isto, o PSO evoluiu de forma lenta, chegando a resultados próximos aos alcançados pelo AG apenas no final das iterações. Esta curva também sugere potencial de melhora das soluções, pois não houve tempo suficiente para a evolução do PSO se estabilizar. Pode ser possível acelerar esta convergência por meio de ajuste de parâmetros, que é o objetivo da Seção 4.4.

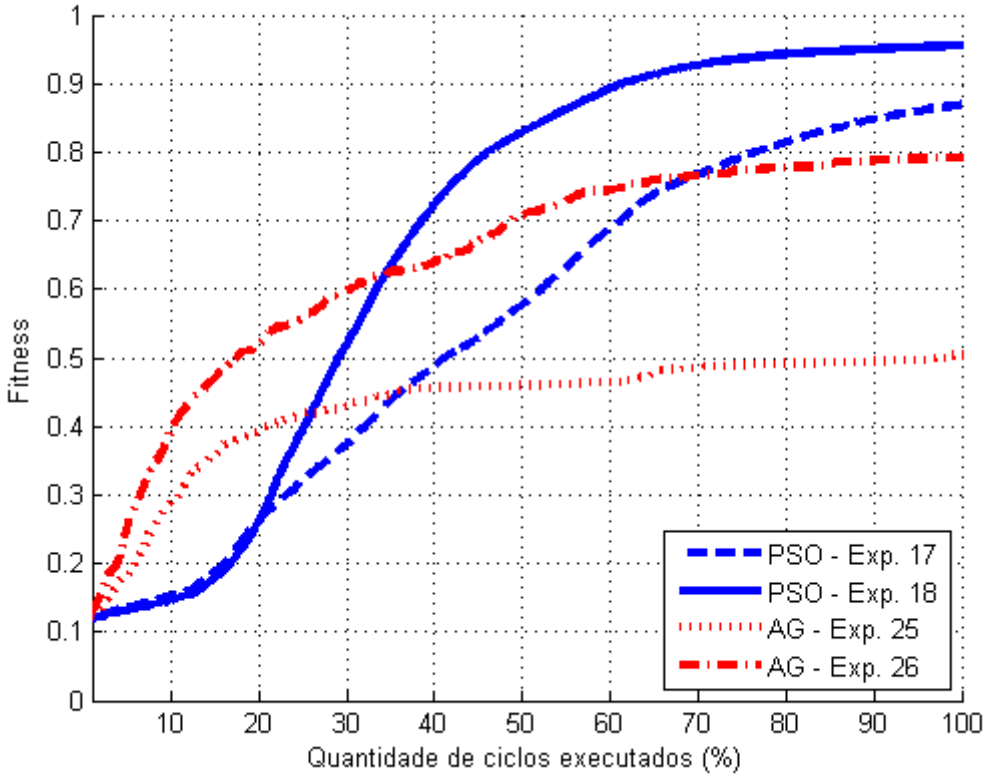


Figura 28: Média da evolução do fitness máximo para o filtro 2

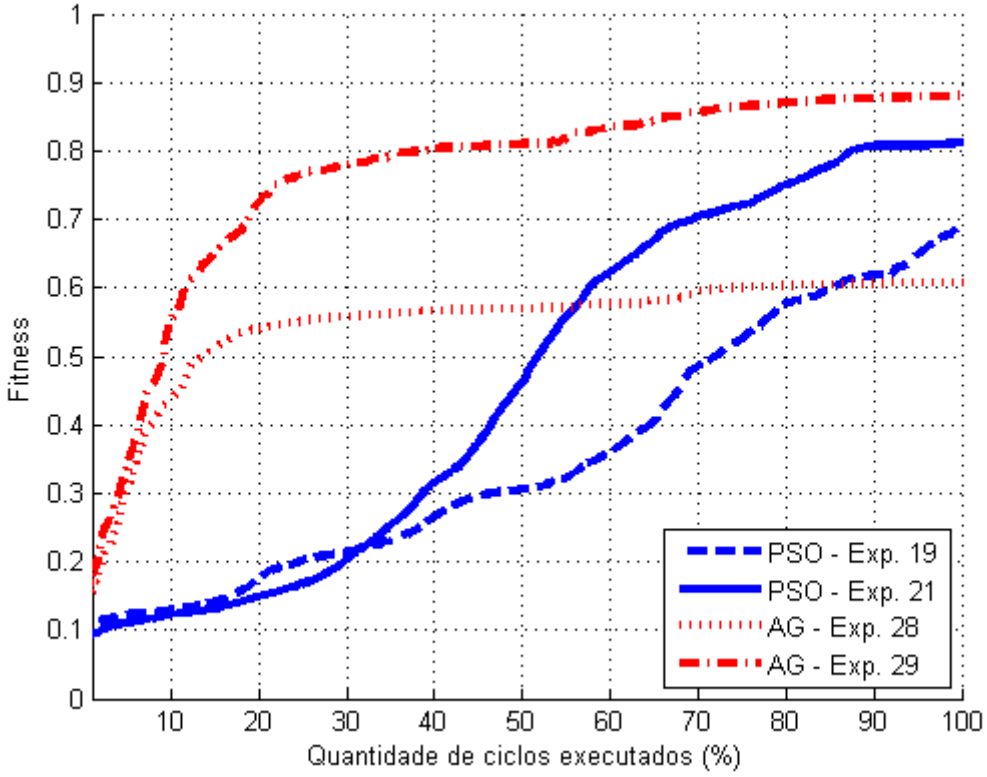


Figura 29: Média da evolução do fitness máximo para o filtro 3

4.3.2 Discussão

O PSO apresentou resultados superiores aos do AG nos filtros 1 e 2, e de qualidade um pouco inferior no filtro 3. Especula-se que uma das principais causas para esta discrepância de desempenho seja a alta velocidade de convergência do AG.

Por este motivo, o PSO foi o algoritmo de Computação Evolucionária escolhido para o restante do trabalho. Porém, deve-se salientar que apesar do AG apresentar resultados piores do que o PSO com o número de avaliações estipulado nestes experimentos, sua rápida convergência faz com que, para poucas avaliações, o AG seja melhor. Além disto, no terceiro filtro, o resultado alcançado pelo AG na melhor configuração foi superior ao resultado do PSO também na melhor configuração.

4.4 AJUSTE DE PARÂMETROS

A partir dos resultados obtidos nos experimentos da Seção 4.3, o PSO foi escolhido como o melhor algoritmo de CE para o projeto de filtros digitais com a nova função de *fitness* proposta. Com isto, o próximo passo é executar uma série de experimentos para se encontrar o melhor conjunto de parâmetros de configuração do PSO para esta função. Estes conjuntos estão mostrados na Tabela 9. Como o número de experimentos desta Seção é bem maior, e o poder computacional exigido seria muito elevado, cada experimento foi executado 20 vezes, em vez das 30 execuções utilizadas nos experimentos anteriores.

Os valores utilizados foram variações em torno dos parâmetros padrão utilizados para o PSO na literatura. Outros experimentos foram realizados utilizando-se valores tanto acima quanto abaixo das faixas apresentadas ma Tabela 9, mas os resultados foram muito inferiores e excluídos do trabalho final para facilitar a visualização dos dados mais relevantes.

4.4.1 Resultados

Os resultados obtidos pelos experimentos são exibidos na Tabela 10.

Nas Figuras 30, 31 e 32, são mostrados os gráficos de Pareto com os resultados dos experimentos de ajuste de parâmetros para cada filtro, respectivamente. A configuração $c_g = 1,8$, $c_l = 1,8$, $w = 0,5$ foi excluída dos três gráficos por ter obtido resultados ruins e discrepantes, necessitando de um aumento na área do gráfico para ser exibido e diminuindo o nível de detalhe mostrado nas regiões de interesse. O eixo horizontal do gráfico mostra $1 - fit_{medio}$ em vez do fit_{medio} de forma direta, já que o padrão na visualização deste tipo de informação é minimizar

Exp. N^o	Ordem	ω_p	ω_r	N	M	c_g	c_l	w
31	6	0,45	0,50	100	1000	1,8	1,8	0,5
32	6	0,45	0,50	100	1000	2,0	2,0	0,5
33	6	0,45	0,50	100	1000	2,2	2,2	0,5
34	6	0,45	0,50	100	1000	1,8	1,8	0,6
35	6	0,45	0,50	100	1000	2,0	2,0	0,6
36	6	0,45	0,50	100	1000	2,2	2,2	0,6
37	6	0,45	0,50	100	1000	1,8	1,8	0,7
38	6	0,45	0,50	100	1000	2,0	2,0	0,7
39	6	0,45	0,50	100	1000	2,2	2,2	0,7
40	6	0,80	0,85	100	1000	1,8	1,8	0,5
41	6	0,80	0,85	100	1000	2,0	2,0	0,5
42	6	0,80	0,85	100	1000	2,2	2,2	0,5
43	6	0,80	0,85	100	1000	1,8	1,8	0,6
44	6	0,80	0,85	100	1000	2,0	2,0	0,6
45	6	0,80	0,85	100	1000	2,2	2,2	0,6
46	6	0,80	0,85	100	1000	1,8	1,8	0,7
47	6	0,80	0,85	100	1000	2,0	2,0	0,7
48	6	0,80	0,85	100	1000	2,2	2,2	0,7
49	6	0,08	0,10	100	1000	1,8	1,8	0,5
50	6	0,08	0,10	100	1000	2,0	2,0	0,5
51	6	0,08	0,10	100	1000	2,2	2,2	0,5
52	6	0,08	0,10	100	1000	1,8	1,8	0,6
53	6	0,08	0,10	100	1000	2,0	2,0	0,6
54	6	0,08	0,10	100	1000	2,2	2,2	0,6
55	6	0,08	0,10	100	1000	1,8	1,8	0,7
56	6	0,08	0,10	100	1000	2,0	2,0	0,7
57	6	0,08	0,10	100	1000	2,2	2,2	0,7

Tabela 9: Experimentos de ajuste de parâmetros

os dois eixos. No eixo vertical está o desvio padrão obtido no *fitness* médio dos experimentos. A escolha do *fitness* como um dos critérios a ser otimizado é óbvia, já que quanto melhor o *fitness* médio obtido por uma certa configuração, melhor serão os filtros que ela fornecerá.

Já a escolha do desvio padrão, apesar de pouco usual, neste caso é justificada. Um alto valor de desvio indica que em algumas rodadas, o algoritmo "travou"em soluções de baixa qualidade e não conseguiu migrar para melhores regiões do espaço de busca. Esta relação foi observada nos resultados completos dos experimentos, que não estão mostrados aqui por razões de espaço e clareza. Porém, é possível perceber os indícios deste acontecimento na Tabela 10, comparando-se a ocorrência de baixos valores de *fitness* mínimo nos casos em que o desvio padrão é maior.

Em casos onde a utilização do algoritmo for *off-line*, este comportamento dificilmente levará a problemas. Nestes casos, um resultado ruim pode simplesmente ser descartado, e uma

Exp. N^o	*Fitness* máximo	*Fitness* mínimo	*Fitness* médio
31	0,9852	0,4740	0,9223 ± 0,1214
32	0,9883	0,9508	0,9815 ± 0,0083
33	0,9876	0,9502	0,9774 ± 0,0101
34	0,9884	0,9705	0,9835 ± 0,0049
35	0,9877	0,9584	0,9798 ± 0,0068
36	0,9855	0,9489	0,9762 ± 0,0093
37	0,9880	0,9547	0,9814 ± 0,0073
38	0,9860	0,9518	0,9704 ± 0,0108
39	0,9852	0,9153	0,9714 ± 0,0140
40	0,9604	0,1780	0,6604 ± 0,2468
41	0,9763	0,4695	0,9265 ± 0,1088
42	0,9645	0,8744	0,9432 ± 0,0189
43	0,9917	0,9281	0,9591 ± 0,0167
44	0,9726	0,9069	0,9510 ± 0,0173
45	0,9777	0,8728	0,9438 ± 0,0276
46	0,9863	0,9311	0,9578 ± 0,0139
47	0,9893	0,8811	0,9531 ± 0,0229
48	0,9904	0,7539	0,9304 ± 0,0622
49	0,9147	0,1411	0,5670 ± 0,3082
50	0,9148	0,0504	0,7126 ± 0,3243
51	0,9148	0,5603	0,8480 ± 0,0877
52	0,9148	0,2034	0,7411 ± 0,2523
53	0,9145	0,2034	0,8155 ± 0,2151
54	0,9140	0,5673	0,8682 ± 0,0878
55	0,9148	0,1743	0,8517 ± 0,1671
56	0,9141	0,2034	0,7747 ± 0,2217
57	0,9134	0,5607	0,7885 ± 0,1123

Tabela 10: Resultados dos experimentos de ajuste de parâmetros

nova rodada do algoritmo provavelmente obterá um filtro melhor.

Mas se este algoritmo for utilizado em sistemas de forma *online*, isto é, onde o cálculo do filtro acontece durante o funcionamento do sistema, este tipo de ocorrência pode ser desastroso. Um filtro com *fitness* muito baixo pode comprometer a operação de forma correta do sistema, e como o cálculo do filtro poderá ser executado sem muita margem para a repetição do processo, é arriscado usar uma configuração que forneça em média filtros muito bons, mas que ocasionalmente resulte em filtros muito ruins.

O objetivo deste tipo de procedimento de ajuste é definir um conjunto padrão de parâmetros que leve o algoritmo a encontrar bons filtros independemente das especificações do mesmo. Nem sempre isto é possível, pois cada problema ou instância de problema pode ter peculiaridades que façam o algoritmo obter um desempenho abaixo do esperado com o conjunto padrão de parâmetros. Ainda assim, é melhor fornecer um ou mais conjuntos de parâmetros do

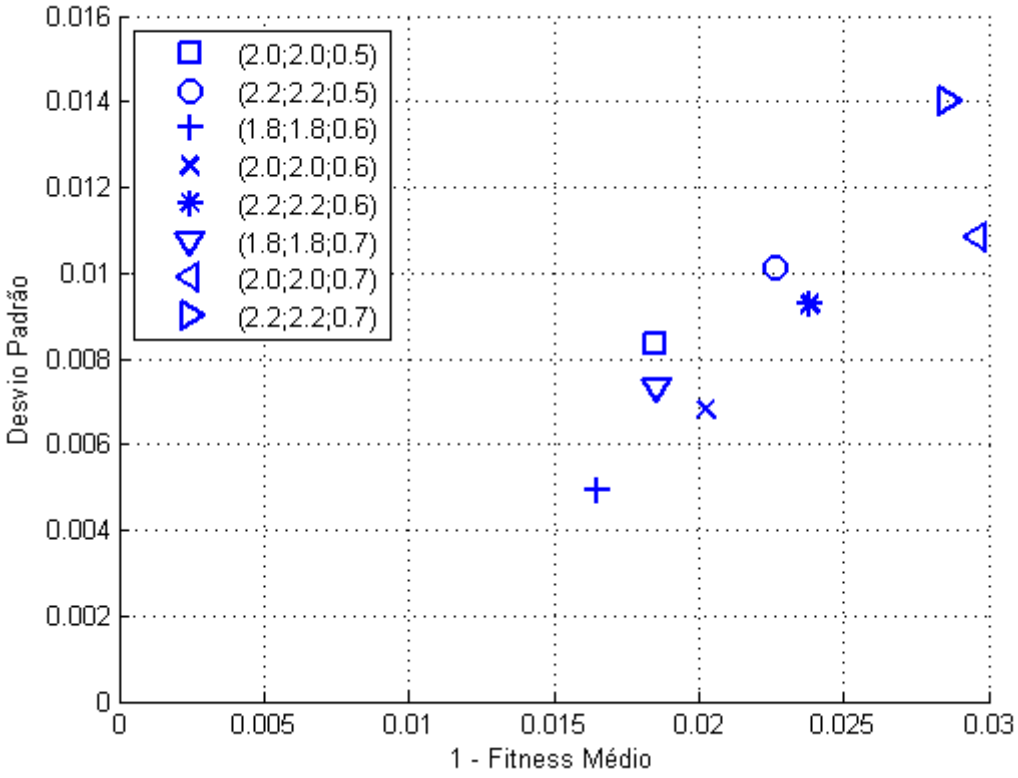


Figura 30: Gráfico de Pareto para o filtro 1

que deixar os valores em aberto.

Na Figura 30, está claro que o melhor conjunto de parâmetros é o $c_g = 1,8$, $c_l = 1,8$, $w = 0,6$. Ele obteve tanto o melhor *fitness* médio quanto o menor desvio padrão, levando a crer que este conjunto é o ideal para filtros com as mesmas características, em qualquer aplicação. Após este, tem-se 3 conjuntos com desempenho aproximadamente semelhante: $c_g = 1,8$, $c_l = 1,8$, $w = 0,7$ e $c_g = 2,0$, $c_l = 2,0$, $w = 0,5$, com os melhores *fitness*, e $c_g = 2,0$, $c_l = 2,0$, $w = 0,6$, com menor desvio padrão, mas *fitness* um pouco pior.

A Figura 31 mostra que para o filtro 2, não houve um conjunto dominante, mas sim dois que obtiveram desempenho similar. O conjunto $c_g = 1,8$, $c_l = 1,8$ e $w = 0,6$ novamente alcançou o melhor *fitness*, enquanto que $c_g = 1,8$, $c_l = 1,8$ e $w = 0,7$ teve um desvio padrão ligeiramente melhor, e *fitness* ligeiramente pior. O resultado obtido pelo conjunto $c_g = 2,0$, $c_l = 2,0$ e $w = 0,5$ foi muito pior do que os outros conjuntos. Para aumentar o nível de detalhe do gráfico na área mais importante, este conjunto foi excluído.

Na Figura 32, que mostra o desempenho dos diferentes conjuntos de parâmetros para o filtro mais difícil, o melhor resultado foi obtido utilizando-se o conjunto $c_g = 2,2$, $c_l = 2,2$ e $w = 0,6$. Em seguida tem-se $c_g = 2,2$, $c_l = 2,2$ e $w = 0,5$ com resultado semelhante, e $c_g = 1,8$,

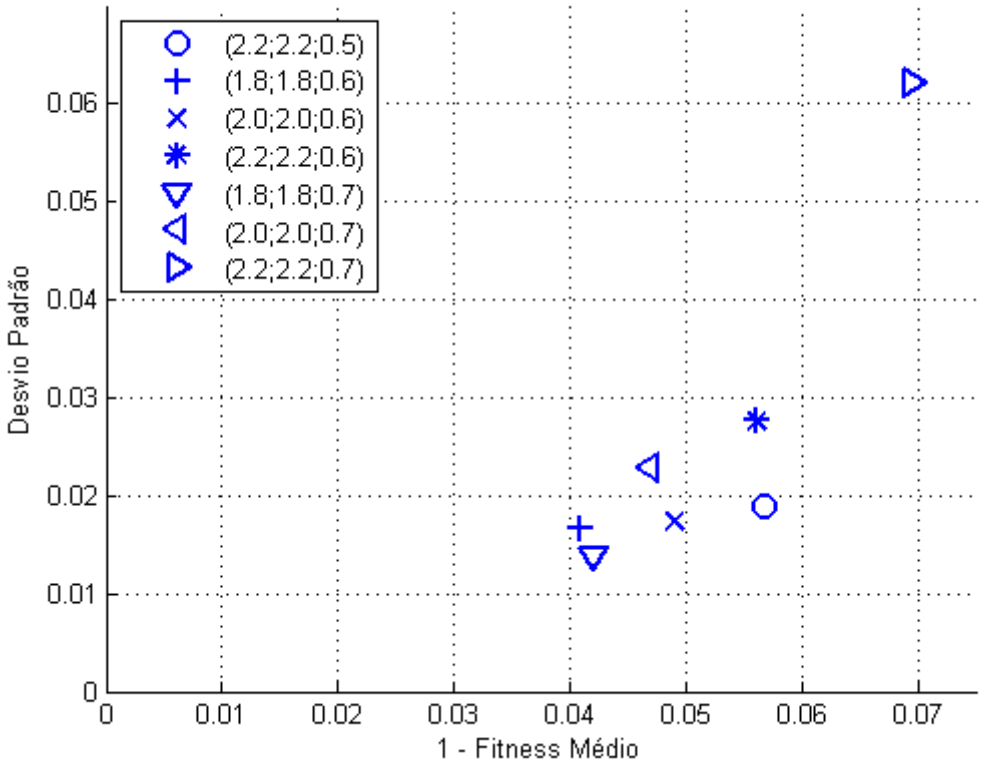


Figura 31: Gráfico de Pareto para o filtro 2

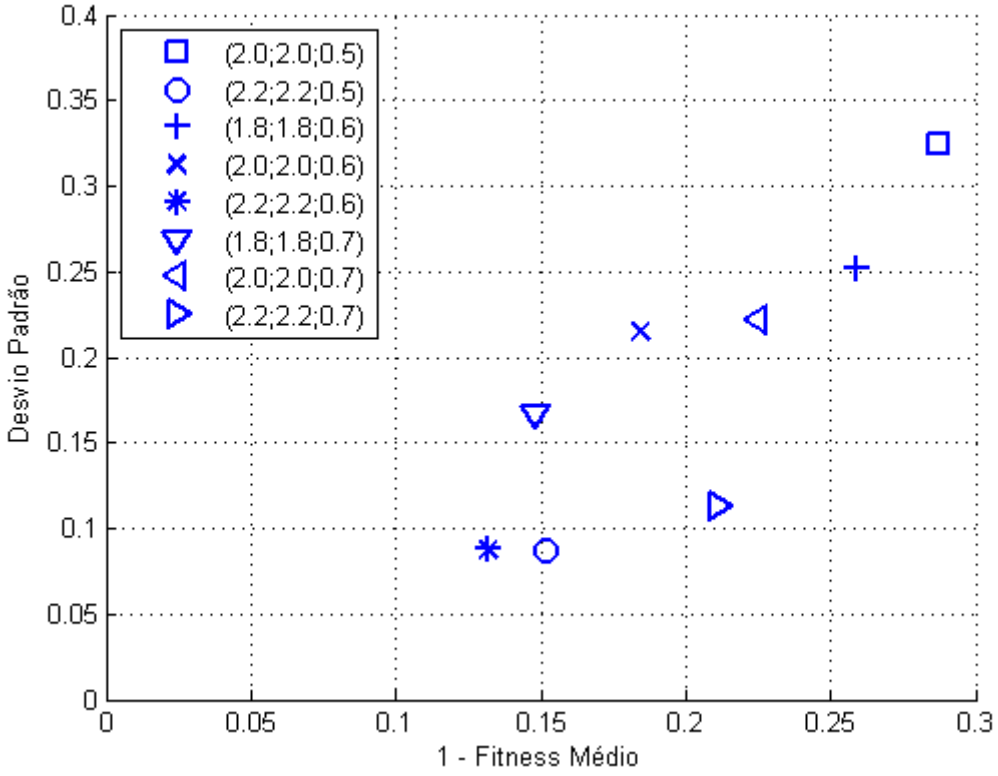


Figura 32: Gráfico de Pareto para o filtro 3

$c_l = 1,8$ e $w = 0,7$ com *fitness* ligeiramente maior, mas desvio padrão pior. O conjunto que obteve o melhor resultado nos outros filtros, $c_g = 1,8$, $c_l = 1,8$ e $w = 0,6$, foi o segundo pior para este caso.

4.4.2 Discussão

O conjunto de parâmetros que obteve os melhores resultados nos dois primeiros filtros foi $c_g = 1,8$, $c_l = 1,8$ e $w = 0,6$, e no terceiro foi $c_g = 2,2$, $c_l = 2,2$ e $w = 0,6$. Por ter atingido melhores resultados mais vezes, o primeiro foi o escolhido como conjunto padrão de parâmetros para o conjunto de codificação e função de *fitness* proposto. Porém, o segundo conjunto também pode ser utilizado em filtros mais difíceis.

Outro conjunto recomendado é $c_g = 1,8$, $c_l = 1,8$ e $w = 0,7$, que alcançou o segundo melhor *fitness* em todos os filtros. É, portanto, uma configuração que mostrou desempenho razoável em filtros de dificuldade diferente, do mais fácil ao mais difícil.

4.5 EQUALIZAÇÃO

Como a equalização é sempre feita para um filtro específico, não faz sentido especificar bandas e ganhos. É necessário passar um filtro já pronto para o cálculo do equalizador. Por isto, foram escolhidos filtros calculados nos testes de *fitness*, e então equalizados. É necessário testar tanto equalizadores inversíveis quanto não-inversíveis.

Foram escolhidos aleatoriamente 2 filtros diferentes de cada especificação, dentro do conjunto de filtros obtidos nos experimentos de ajuste de parâmetros que obtiveram melhores resultados. Ou seja, dos experimentos 34, 43 e 54, dois filtros de cada. Estes 6 filtros foram utilizados para avaliar o desempenho da equalização proposta. A Tabela 11 mostra as diferentes configurações do algoritmo para este problema.

Os parâmetros padrão do PSO foram utilizados, mas o tamanho da população e o número de iterações foram estabelecidos de forma empírica. Estes valores estão próximos dos comumente utilizados para o PSO. Deve-se ressaltar que, como citado na Seção 3.3, este algoritmo é separado do estudado até agora. O objetivo deste é encontrar o equalizador para um filtro específico, a codificação utilizada é a por coeficientes, e a função de *fitness* é baseada na variância do atraso de fase. Portanto, não faz sentido utilizar os parâmetros encontrados na Seção 4.4, que trata do ajuste de parâmetros, já que são casos completamente diferentes de aplicação do PSO.

Exp. N^o	Filtro N^o	Ordem	Inversível?	Rodadas	N	M	c_g	c_l	w
58	1a	6	S	20	50	100	2,0	2,0	0,5
59	1a	6	N	20	50	100	2,0	2,0	0,5
60	1a	4	S	20	50	100	2,0	2,0	0,5
61	1a	4	N	20	50	100	2,0	2,0	0,5
62	1b	6	S	20	50	100	2,0	2,0	0,5
63	1b	6	N	20	50	100	2,0	2,0	0,5
64	1b	4	S	20	50	100	2,0	2,0	0,5
65	1b	4	N	20	50	100	2,0	2,0	0,5
66	2a	4	S	20	50	100	2,0	2,0	0,5
67	2a	4	N	20	50	100	2,0	2,0	0,5
68	2a	2	S	20	50	100	2,0	2,0	0,5
69	2a	2	N	20	50	100	2,0	2,0	0,5
70	2b	4	S	20	50	100	2,0	2,0	0,5
71	2b	4	N	20	50	100	2,0	2,0	0,5
72	2b	2	S	20	50	100	2,0	2,0	0,5
73	2b	2	N	20	50	100	2,0	2,0	0,5
74	3a	6	S	20	50	100	2,0	2,0	0,5
75	3a	6	N	20	50	100	2,0	2,0	0,5
76	3a	4	S	20	50	100	2,0	2,0	0,5
77	3a	4	N	20	50	100	2,0	2,0	0,5
78	3b	6	S	20	50	100	2,0	2,0	0,5
79	3b	6	N	20	50	100	2,0	2,0	0,5
80	3b	4	S	20	50	100	2,0	2,0	0,5
81	3b	4	N	20	50	100	2,0	2,0	0,5

Tabela 11: Experimentos de equalização

Os filtros são denominados por um número e uma letra, onde o número indica a especificação do filtro (conforme as Tabelas 1 e 3) e a letra mostra se é o primeiro ou segundo filtro selecionado dentre os experimentos anteriores.

4.5.1 Resultados

Os resultados dos experimentos de equalização são mostrados na Tabela 12. As colunas Estável e Inversível mostram as taxas de acerto destas condições para cada experimento. É possível notar que não foi alcançado nenhum equalizador inversível. A taxa de estabilidade foi mais elevada nos casos em que não se buscou a inversibilidade, o que indica que esta tentativa levou o algoritmo para espaços de busca muito ruins.

Além da estabilidade, é possível perceber o impacto da consideração da inversibilidade do equalizador no *fitness* médio encontrado. Em todos os experimentos, quando não se buscou um equalizador inversível, o *fitness* obtido foi muito maior. Isto é uma consequência óbvia

da falha em obter equalizadores inversíveis, já que as penalidades advindas desta condição diminuem o *fitness* das soluções. As Figuras 33 a 35 mostram as resposta em fase obtidas em cada experimento, pelo melhor equalizador encontrado. Os experimentos com equalizadores inversíveis foram excluídos, pois a instabilidade foi muito alta e a qualidade muito baixa.

Exp N^o.	*Fitness* máximo	*Fitness* mínimo	*Fitness Médio*	Estável?	Inversível?
58	0,1385	0,0263	0,0921 $\pm$ 0,0319	40,0%	0,0%
59	0,6446	0,2332	0,4772 $\pm$ 0,1194	70,0%	-
60	0,3335	0,0813	0,1922 $\pm$ 0,0689	45,0%	0,0%
61	0,5754	0,2650	0,4129 $\pm$ 0,0772	85,0%	-
62	0,1391	0,0547	0,1035 $\pm$ 0,0271	30,0%	0,0%
63	0,7035	0,0633	0,5117 $\pm$ 0,1457	65,0%	-
64	0,3586	0,0769	0,1736 $\pm$ 0,0616	50,0%	0,0%
65	0,4489	0,2695	0,3835 $\pm$ 0,0380	70,0%	-
66	0,1408	0,0121	0,0931 $\pm$ 0,0464	25,0%	0,0%
67	0,8816	0,4360	0,7787 $\pm$ 0,1205	70,0%	-
68	0,3832	0,0321	0,1774 $\pm$ 0,0756	40,0%	0,0%
69	0,7343	0,4039	0,6965 $\pm$ 0,0786	55,0%	-
70	0,1362	0,0097	0,0798 $\pm$ 0,0446	20,0%	0,0%
71	0,8744	0,0834	0,7328 $\pm$ 0,1812	80,0%	-
72	0,4029	0,0716	0,1533 $\pm$ 0,0642	75,0%	0,0%
73	0,8085	0,3943	0,6717 $\pm$ 0,1020	100,0%	-
74	0,0659	0,0132	0,0361 $\pm$ 0,0170	95,0%	0,0%
75	0,0720	0,0660	0,0673 $\pm$ 0,0024	90,0%	-
76	0,0654	0,0220	0,0423 $\pm$ 0,0174	45,0%	0,0%
77	0,0661	0,0661	0,0661 $\pm$ 0,0000	90,0%	-
78	0,0658	0,0132	0,0375 $\pm$ 0,0204	45,0%	0,0%
79	0,0740	0,0661	0,0669 $\pm$ 0,0019	90,0%	-
80	0,0662	0,0221	0,0400 $\pm$ 0,0176	85,0%	0,0%
81	0,0663	0,0662	0,0662 $\pm$ 0,0000	75,0%	-

Tabela 12: Resultados dos experimentos de equalização

4.5.2 Discussão

O método proposto de equalização de fase foi incapaz de fornecer equalizadores inversíveis. A consideração desta característica em forma de penalidade (como explicado na Seção 3.3) não apenas não surtiu efeito, como também prejudicou a evolução das soluções nos outros critérios, que são a estabilidade e a linearidade da fase do filtro.

Nos casos em que apenas a estabilidade do equalizador penalizava o *fitness* das soluções, os resultados foram melhores. Foram encontrados equalizadores estáveis em cerca de 75% das execuções. Este número pode tornar o uso do algoritmo em sistemas *online* inviável. Portanto, ajustes e modificações no método devem ser efetuadas para que este tipo de aplicação também

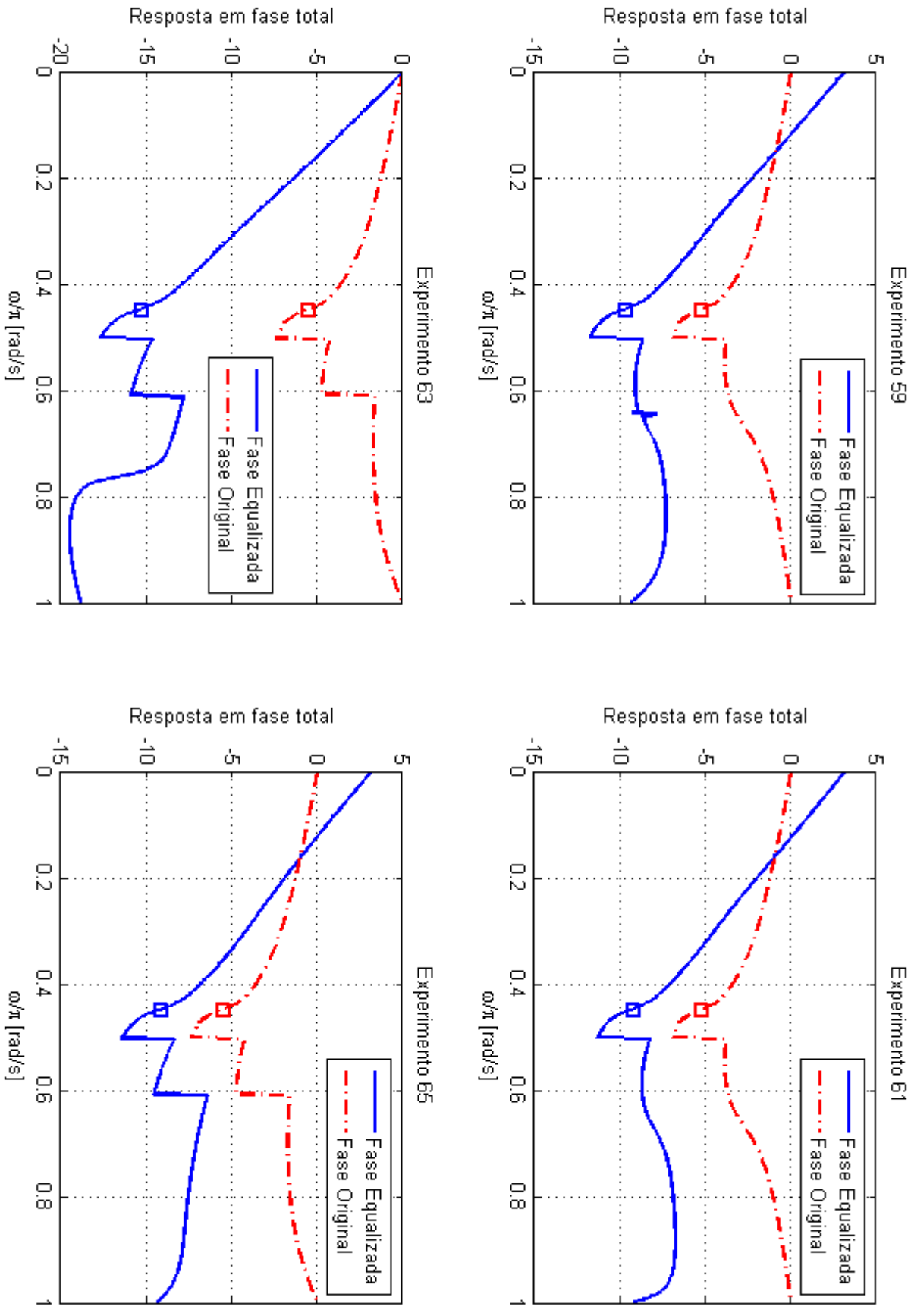


Figura 33: Resposta em fase obtida nos experimentos 59 a 65

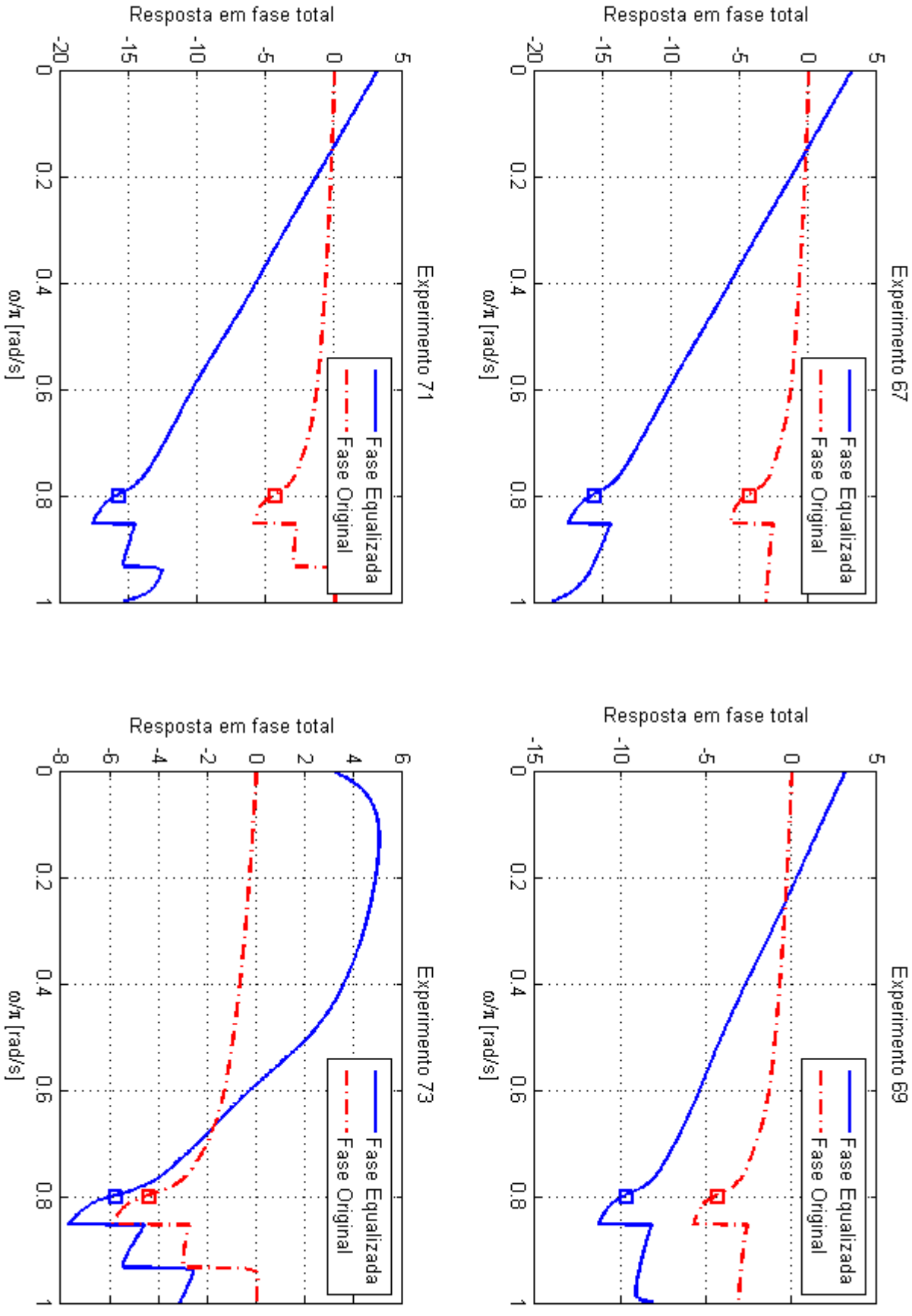


Figura 34: Resposta em fase obtida nos experimentos 67 a 73

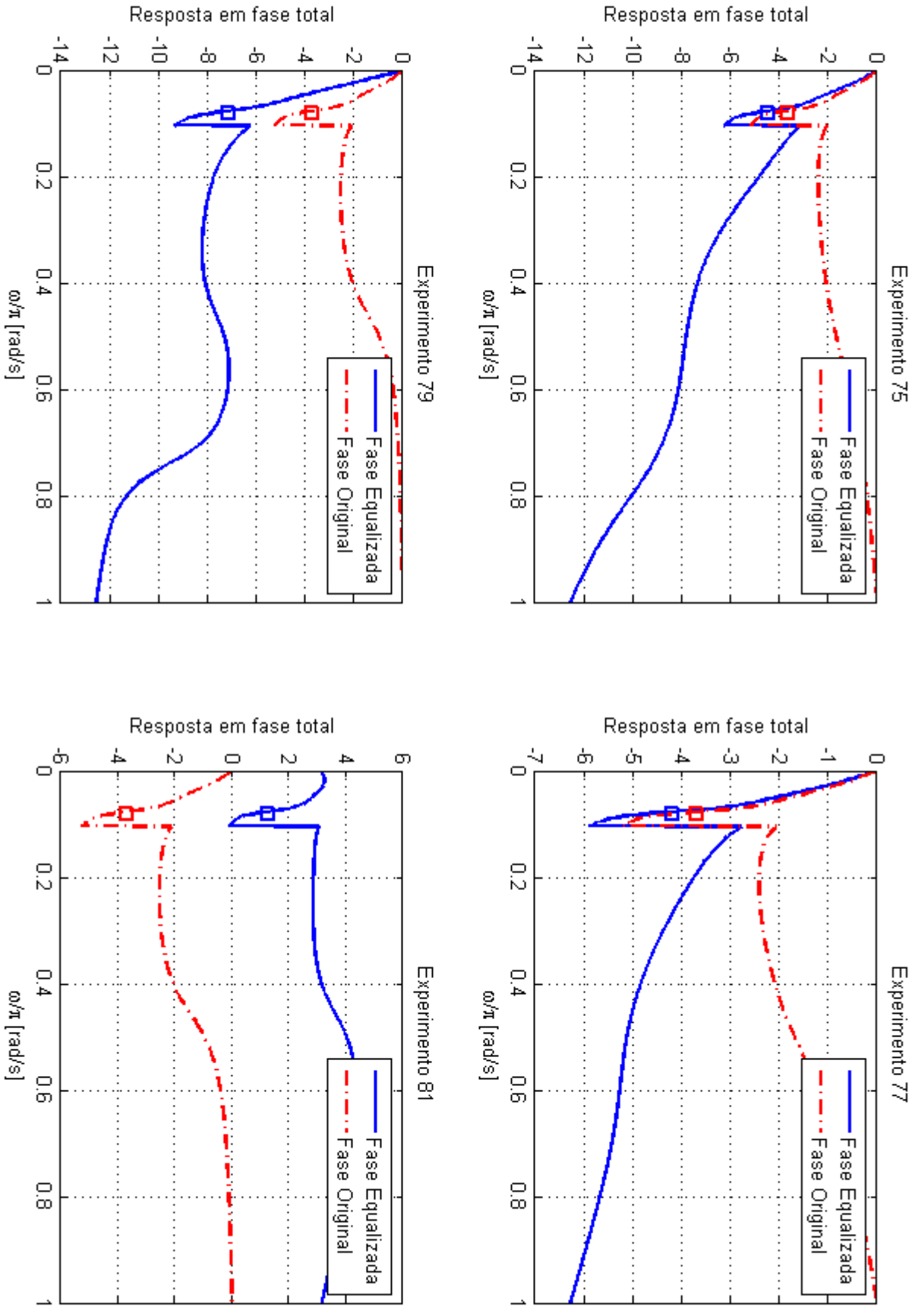


Figura 35: Resposta em fase obtida nos experimentos 75 a 81

seja possível.

O *fitness* obtido nos experimentos foi em média maior no filtro 2, seguido pelo filtro 1 e por último pelo filtro 3. Esta diferença é atribuída à largura das bandas de passagem, pois esta ordem é decrescente de largura. Em geral, a resposta em fase muda de forma abrupta próximo da frequência de corte (NOCETI FILHO, 1998), sendo este trecho responsável por uma parcela considerável da variância desta resposta. Deste modo, bandas de passagem mais largas tem proporcionalmente mais amostras na faixa que varia de forma mais suave, diminuindo a variância como um todo.

Para o restante do trabalho, será utilizado o equalizador não-inversível. Apesar de seu desempenho não ter sido muito satisfatório, foi muito melhor do que o obtido com os equalizadores inversíveis.

4.6 COMPARAÇÃO COM MÉTODOS ESPECIALISTAS

Após a realização dos experimentos descritos nas Seções 4.1 a 4.4, obteve-se uma implementação de um sistema para cálculo de filtros IIR, combinando as melhores codificação, função de *fitness* e algoritmo de CE, sintonizado com os parâmetros que melhores resultados apresentaram. O passo final é comparar o desempenho deste sistema com métodos de cálculo convencionais da literatura. Como descrito na Seção 2.3.2, dos quatro métodos mais utilizados para este fim, o elíptico é o único que não apresenta monotonicidade em nenhuma das bandas. Apesar desta característica ser uma desvantagem, ela torna o *roll-off* deste tipo de filtro mais acentuado. Como os filtros calculados pela abordagem aqui apresentada também não apresentam monotonicidade (para isto teriam que obedecer aos critérios de Butterworth ou Chebyshev (DINIZ et al., 2004)), a comparação mais justa é com o método elíptico.

O método elíptico não permite a especificação de todos os parâmetros de um filtro ao mesmo tempo. Se a ordem do filtro por especificada, não é possível fixar as frequências de limiar de ambas as bandas, apenas de uma. Se o desejo for fixar tanto o limite da banda de passagem quanto da banda de rejeição, não é possível especificar a ordem do filtro.

Comparar filtros de ordem diferente é injusto, pois a taxa de *roll-off* é limitada matematicamente pelo número de pólos e zeros do filtro. Além disto, o esforço computacional exigido na implementação de filtros de ordem mais elevada é maior, sendo, portanto, uma das maiores restrições para o uso de um filtro em um sistema real (HAMMING, 1997).

Portanto, foi escolhido especificar a frequência da banda de passagem, que é a mais importante, e a ordem do filtro. A Tabela 13 mostra as diferentes especificações de filtro uti-

lizadas para a comparação entre a abordagem por CE apresentada neste trabalho, e o método elíptico. Porém, deve-se destacar que a coluna ω_r só é utilizada no cálculo por CE e para analisar a qualidade das respostas obtidas por ambos os métodos. Cada filtro foi calculado pelo método proposto 5 vezes, sendo o melhor resultado utilizado para a comparação.

Estes filtros tem as mesmas especificações de banda utilizadas em todo trabalho, mas foram inseridos novos ganhos, que tornam o problema mais difícil. Desta forma, espera-se obter taxas de acerto inferiores.

Exp N^o	Ordem	ω_p	ω_r	A_p [dB]	A_r [dB]
82	6	0,45	0,5	1	-30
83	6	0,45	0,5	0,1	-30
84	6	0,45	0,5	1	-50
85	6	0,45	0,5	0,1	-50
86	6	0,8	0,85	1	-30
87	6	0,8	0,85	0,1	-30
88	6	0,8	0,85	1	-50
89	6	0,8	0,85	0,1	-50
90	4	0,08	0,1	1	-30
91	4	0,08	0,1	0,1	-30
92	4	0,08	0,1	1	-50
93	4	0,08	0,1	0,1	-50

Tabela 13: Filtros para a comparação com o método elíptico

4.6.1 Resultados

Os resultados obtidos pelos experimentos de comparação com o método elíptico são mostrados na Tabela 14. A coluna MSE de ambos os métodos mostra o erro quadrático médio obtido. A única diferença é que como o projeto de um filtro elíptico é não-estocástico, o resultado é sempre o mesmo. Portanto, não faz sentido calcular o valor médio, e a taxa de acerto será sempre 100% ou 0%.

No primeiro filtro (experimentos 82-85), o algoritmo proposto obteve MSE menor em todos os casos, mas não conseguiu atender às especificações nenhuma vez no experimento 83, onde o filtro elíptico mostrou que era possível. No segundo filtro, (experimentos 86-89), a taxa de acerto também deixa a desejar, pois o filtro elíptico atendeu três especificações, enquanto o método proposto alcançou 66,7% em apenas um deles. A MSE obtida pelo método de CE foi menor nos experimentos 86 e 88. No terceiro filtro (experimentos 90-93), nenhum método conseguiu atender às especificações de nenhum filtro, mas a MSE mostra uma diferença de qualidade tendendo para o algoritmo proposto em todos os experimentos, exceto o 92.

	PSO					Elíptico	
Exp N^o	*Fit.* Máximo	*Fit.* mínimo	*Fitness* médio	Acerto	MSE	Acerto	MSE
82	0,9885	0,9705	0,9847 $\pm$ 0,0039	100,0%	0,0531	100,0%	0,6130
83	0,9876	0,9389	0,9788 $\pm$ 0,0039	0,0%	0,0538	100,0%	0,0707
84	0,9867	0,9530	0,9821 $\pm$ 0,0065	0,0%	0,0634	0,0%	0,5428
85	0,9849	0,9481	0,9784 $\pm$ 0,0079	0,0%	0,0546	0,0%	0,0712
86	0,9924	0,9332	0,9576 $\pm$ 0,0160	66.7%	0,2081	100,0%	1,1642
87	0,9920	0,7366	0,9457 $\pm$ 0,0444	0,0%	0,1832	100,0%	0,0309
88	0,9929	0,3995	0,9255 $\pm$ 0,1058	0,0%	0,4960	100,0%	1,0519
89	0,9896	0,8483	0,9379 $\pm$ 0,0356	0,0%	0,2020	0,0%	0,0111
90	0,9146	0,2034	0,8377 $\pm$ 0,1440	0,0%	0,2308	0,0%	0,2626
91	0,8969	0,1837	0,7882 $\pm$ 0,1469	0,0%	0,2235	0,0%	0,5412
92	0,8837	0,1800	0,7683 $\pm$ 0,1945	0,0%	0,3429	0,0%	0,1872
93	0,8629	0,6688	0,7812 $\pm$ 0,0862	0,0%	0,1578	0,0%	1,1555

Tabela 14: Resultados dos experimentos de comparação com o método elíptico

As Figuras 36, 37 e 38 comparam os filtros médios (çom *fitness* mais próximo da média obtida pelo algoritmo) obtidos nos experimentos 82 a 93 com os filtros obtidos pelo método elíptico. Por razões de espaço e clareza, a legenda foi excluída das figuras. A linha contínua representa o filtro obtido pelo PSO com as configurações propostas, e a linha tracejada representa o filtro elíptico obtido.

O resultado obtido nos experimentos 82 a 85 é exibido na Figura 36. O desempenho foi semelhante, mas nota-se uma diferença no experimento 85. Nele, o *roll-off* obtido pelo PSO foi mais acentuado do que o do filtro elíptico. Isto ocorre porque o método elíptico força um ganho de ondulação constante nas duas bandas, e como a ordem fornecida é insuficiente para atender todas as especificações, o resultado é um filtro com uma violação mais acentuada próximo da fronteira da banda, e atenuação dentro do desejado apenas em frequências mais altas.

Na Figura 37 tem-se o segundo grupo de filtros, projetados nos experimentos 86 a 89. Aqui, o desempenho foi semelhante aos experimentos 86 e 87, mas o PSO não conseguiu alcançar as atenuações desejadas nos experimentos 88 e 89. Outra caraterística notável é que os filtros obtidos pelo PSO nos 4 experimentos foram bem parecidos. Isto pode indicar que caso o algoritmo não consiga atender às especificações, a distância das mesmas não influencia muito. Esta característica deve ser corrigida, pois indica falta de resolução na medição de desobediência ao ganho da banda de rejeição. Desta forma, atenuações muito elevadas são difíceis de ser atigidas.

As tendências observadas nos experimentos 82 a 89 são novamente observads nos experimentos 90 a 93, mostrados na Figura 38. Nestes quatro experimentos, a atenuação na banda de rejeição tem valores altos, já que é impossível atendê-los. Nos experimentos 90 e 91,

que tem a atenuação menos acentuada, esta característica não é problemática, já que a resposta fica bem próxima da especificação. Já em 92 e 93, a diferença é muito grande. A outra tendência é a de buscar um *roll-off* que possibilite o atendimento da especificação da banda de rejeição em sua borda. Nos experimentos 91 a 93, o *roll-off* obtido pelo PSO foi mais rápido que o obtido pelo filtro elíptico, que busca o atendimento do ganho com ondulação constante.

4.6.2 Discussão

Os filtros obtidos pelo PSO com a codificação e função de *fitness* propostos obtiveram resultados comparáveis ao do obtido pelo método elíptico, que é um dos mais poderosos métodos especialistas com esta finalidade. Embora o resultado tenha sido pior em alguns casos, o método proposto permite a especificação de todas as variáveis importantes, o que não ocorre no método especialista.

Porém, atenuações elevadas não foram alcançadas com os parâmetros utilizados. O motivo deste fenômeno deve ser investigado, já que limita a flexibilidade e a gama de aplicações em que este pode ser utilizado. Uma das possíveis causas é o valor numericamente baixo que os erros na banda de rejeição causam no valor final da função de *fitness*.

Quando se calcula um filtro elítpico com sua ordem especificada, ele obedece aos critérios da banda de passagem de forma exata, mas a banda de rejeição é apenas um subproduto do processo. O algoritmo que utiliza o PSO busca a obediência de ambos de forma equilibrada, e ainda pode ser calibrada para casos específicos utilizando os pesos existentes na função de *fitness*.

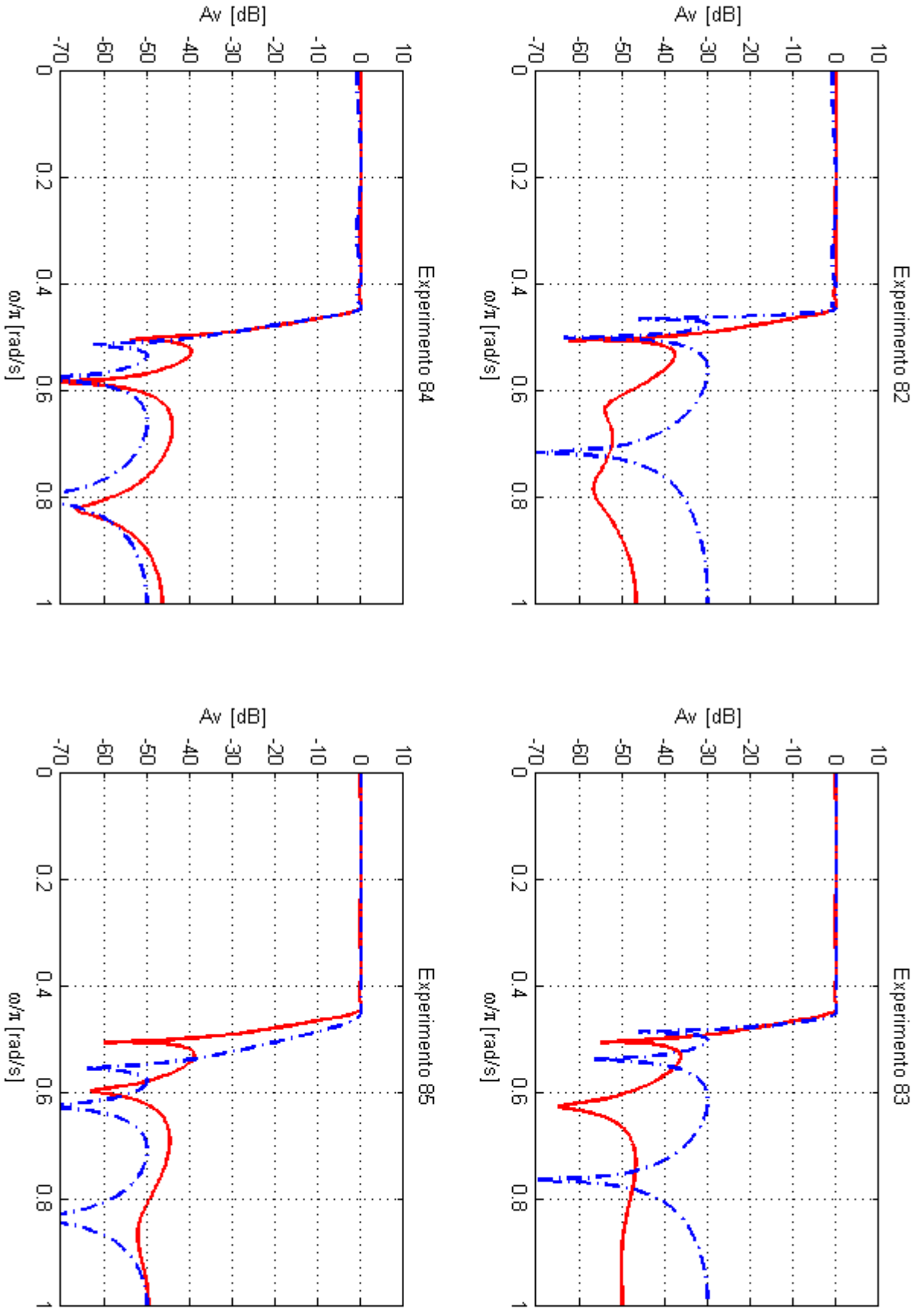


Figura 36: Filtros obtidos nos experimentos 82 a 85

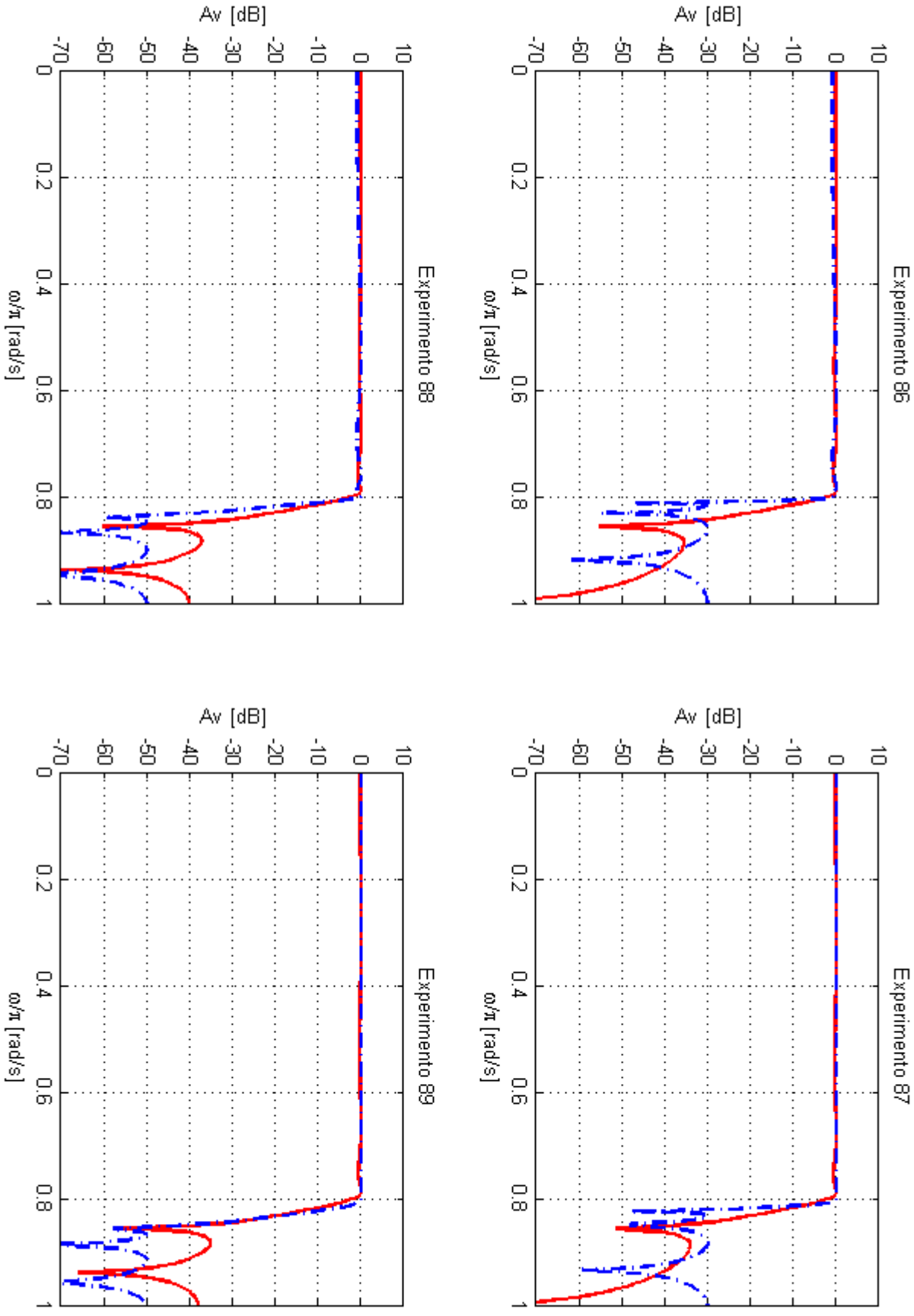


Figura 37: Filtros obtidos nos experimentos 86 a 89

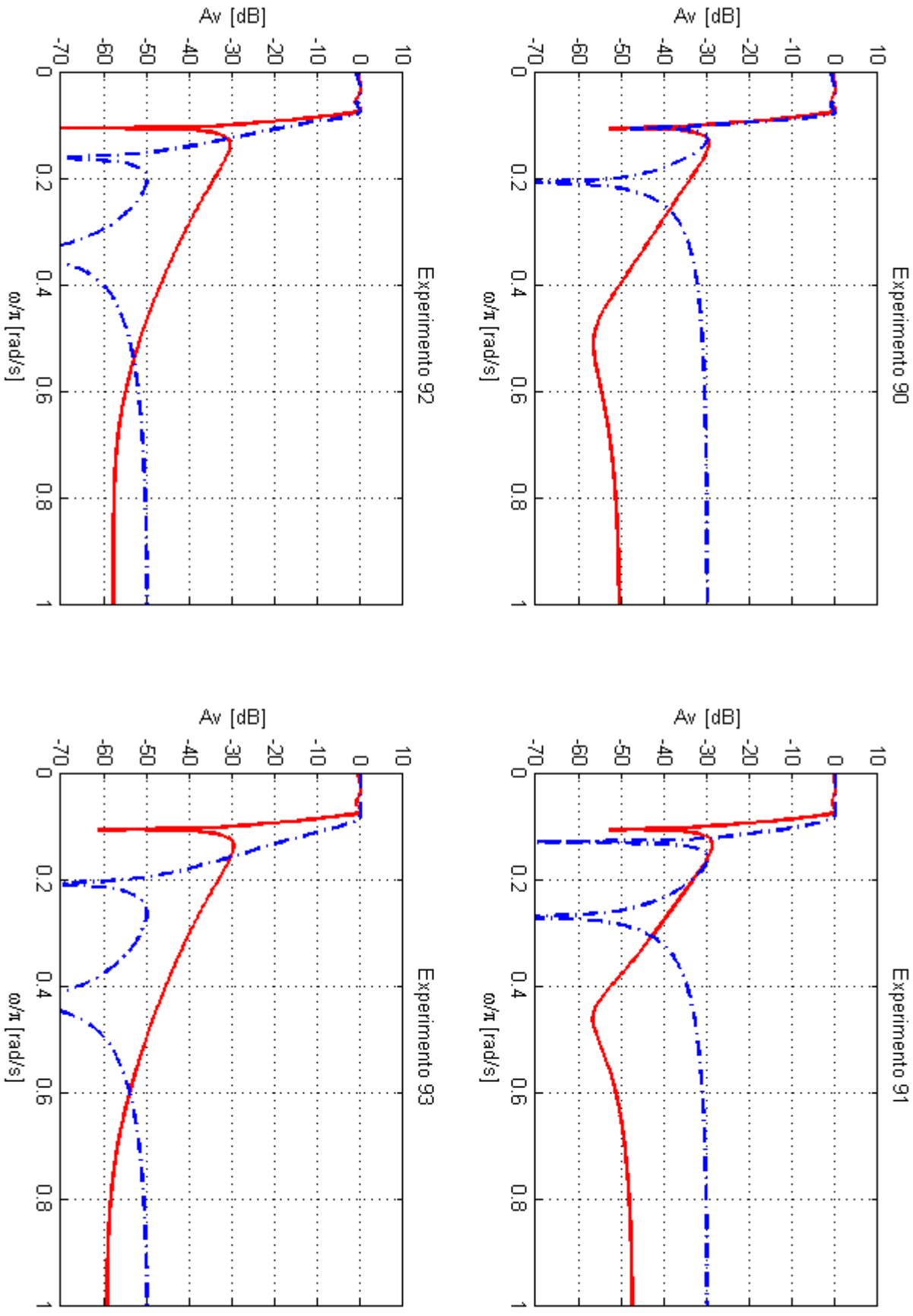


Figura 38: Filtros obtidos nos experimentos 90 a 93

5 DISCUSSÃO E CONCLUSÕES

5.1 DISCUSSÃO DOS RESULTADOS

O primeiro conjunto de testes visou validar a codificação por pólos e zeros proposta como uma alternativa viável à codificação por coeficientes utilizada na literatura. Os resultados mostraram que a qualidade dos filtros obtidos com a nova codificação foi muito melhor. Além disto, não há o risco de se obter uma solução instável ao final da evolução, já que estas sequer podem ser representadas. Apesar do uso de penalidades, este risco está sempre presente na codificação tradicional, já que nada garante que a solução final não desobedecerá estes critérios.

O segundo conjunto de testes comparou os filtros obtidos com a função de *fitness* tradicional, que utiliza o erro quadrático médio (MSE) em relação à resposta ideal do filtro, com a função proposta, que também busca a proximidade com a resposta ideal, mas penaliza filtros que não atendam às especificações, normaliza o erro em relação à largura da banda, e corrige o baixo peso de violações próximas à fronteira das bandas. Os resultados mostraram que a nova função obtem resultados melhores, especialmente no quesito de atendimento às especificações. Em alguns casos, o erro quadrático médio alcançado também foi menor, mostrando que a modificação permitiu que fossem encontradas soluções antes isoladas no espaço de busca.

O terceiro grupo de experimentos comparou a diferença entre os dois algoritmos de CE mais utilizados, o AG e o PSO. Foram testados três diferentes variações de parâmetros em torno dos mais comumente utilizados, para evitar que um eventual baixo desempenho de um deles fosse creditado unicamente ao conjunto padrão de parâmetros. Estes experimentos mostraram que o PSO é mais adequado à resolução deste problema, embora em um dos filtros o AG tenha alcançado melhor resultado. Uma característica observada é a rápida convergência do AG, que pode ser desejada em alguma aplicação específica.

Tendo sido definidos o algoritmo, a codificação e a função de *fitness*, o quarto conjunto de experimentos teve o objetivo de encontrar o melhor conjunto de parâmetros do PSO para este problema. Foram estabelecidos nove diferentes conjuntos, com base na variação positiva e negativa dos parâmetros padrão, e em experimentos empíricos para achar as melhores faixas. Cada

conjunto então foi testado com os filtros estabelecidos no início do Capítulo 4, e o desempenho de cada um foi analisado buscando o melhor *fitness* e o menor desvio padrão. Esta combinação foi escolhida para que os parâmetros padrão forneçam bons resultados, com baixa chance de filtros ruins devido à estagnação precoce da evolução. Dois conjuntos obtiveram os melhores resultados, mas o escolhido foi $c_g = 1,8$, $c_l = 1,8$, e $w = 0,6$.

Após o projeto do filtro, o passo seguinte é a equalização do filtro. Foi proposta a minimização da dispersão do atraso de fase na banda de passagem, medida através da variância da mesma, como citado na Seção 3.3. Esta medida garante a linearidade da resposta em fase (NOCETI FILHO, 1998). Porém, os resultados alcançados não foram bons. Não foi possível o projeto de equalizadores inversíveis, e em muitas execuções, o algoritmo convergiu de forma prematura e não conseguiu sair de regiões ruins do espaço de busca. Uma nova codificação, que ao mesmo tempo garanta a estabilidade e a inversibilidade do equalizador, deve ser estudada. Caso isto não seja possível, pode-se tentar uma solução intermediária.

Quando comparado com os filtros elípticos, o método proposto obteve desempenho semelhante. A taxa de acerto foi inferior, já que o projeto elíptico conseguiu atender às especificações em casos onde a abordagem por CE não conseguiu atender nenhuma vez. Como vantagem, o método proposto permite a especificação simultânea de todos os parâmetros do filtro, e obteve resultados mais equilibrados quando era impossível atender às especificações de banda com a ordem dada.

5.2 CONCLUSÕES

Este trabalho propôs uma nova abordagem para o projeto de filtros digitais com computação evolucionária, enfatizando a produção de bons filtros ao invés de melhoras no desempenho de um algoritmo de CE. Enquanto a maior parte dos trabalhos encontrados na literatura que tratam deste assunto propõem modificações nos algoritmos e utiliza os filtros como *benchmark*, as abordagens aqui estudadas utilizam as propriedades matemáticas dos filtros para modificar a forma como o problema é tratado, sem focar o algoritmo evolucionário em si. Portanto, os avanços aqui obtidos podem ser aplicados facilmente em qualquer outro algoritmo utilizado para o projeto de filtros digitais IIR.

Além de garantir a estabilidade e a inversibilidade de qualquer filtro obtido, o método proposto elevou a qualidade dos filtros calculados por técnicas de CE ao mesmo patamar obtido com técnicas especialistas. Quase nenhum dos trabalhos encontrados na literatura executa essa comparação, especula-se que por causa do foco em melhorias do algoritmo e não na busca por

filtros de boa qualidade.

O método proposto de equalização não obteve resultados satisfatórios. A estratégia escolhida não parece promissora, e estudos com novas abordagens tanto de codificação quanto de *fitness* devem ser realizados, já que as abordagens clássicas já utilizam técnicas de otimização numérica e há diversas aplicações em que os requisitos mudam em tempo real (NOCETI FILHO, 1998).

Uma fraqueza detectada no método proposto é a incapacidade de atingir atenuações acentuadas na banda de rejeição. Uma causa possível é o baixíssimo valor numérico destes ganhos, que gera erros insignificantes perto dos erros da banda de passagem. Uma possível solução é algum tipo de normalização, como usar o erro em dB para a banda de rejeição.

O método proposto alcançou bons resultados com um único conjunto de parâmetros, mas é possível ajustar os mesmos para um filtro específico. A maior sensibilidade observada foi ao parâmetro w, mas como todos os melhores conjuntos de parâmetros possuíam $w = 0,6$, o ajuste fino para cada filtro deve alterar apenas os valores de c_l e c_g.

5.3 TRABALHOS FUTUROS

Diversas melhorias podem ser realizadas no método proposto neste trabalho. Além da correção dos problemas encontrados no Capítulo 4, filtros mais específicos como diferenciadores, multibanda, *notching*, etc. podem exigir ajustes ou diferentes abordagens. Portanto, um dos trabalhos derivados pode investigar o comportamento deste método com filtros cuja resposta em frequência tenha características diferentes.

Uma das aplicações mais promissoras é na de projeto de filtros *online*, isto é, em tempo de operação de um sistema. Em vários casos, as especificações do filtro mudam em tempo real, como em canais de comunicação com resposta não-estacionária. Nestes casos, a utilização de filtros adaptativos é necessária, e a utilização de técnicas evolucionárias deve ser investigada. Como a codificação proposta neste trabalho garante a estabilidade do filtro, ela é mais segura de ser utilizada do que a codificação por coeficientes, que pode gerar filtros instáveis e comprometer o funcionamento do sistema. Serão também estudados novos mecanismos a ser incorporados no PSO para impedir a convergência prematura do algoritmo, pois este fenômeno pode comprometer um sistema de forma crítica.

Na equalização de fase, uma nova abordagem deve ser estudada. Como explicado na Seção 3.3, não é possível aplicar a codificação por pólos e zeros no problema dos equalizadores, pois o requisito de ganho unitário em todas as frequências impõe uma certa relação entre os

polinômios característicos e, portanto, entre os pólos e os zeros do sistema. Porém, alguma solução de compromisso pode ser encontrada. A escolha da dispersão de atraso de fase, apesar de ser uma medida segura da linearidade da resposta, mostrou maus resultados. Uma abordagem diferente é a maximização da simetria da resposta ao impulso (NOCETI FILHO, 1998), que exige um pré-processamento maior.

O projeto simultâneo da resposta em módulo e da resposta em fase, apesar de pouco usual na literatura, também será investigado. Experimentos empíricos mostraram que a convergência para os equalizadores é muito mais rápida do que para o projeto do filtro, mas em um sistema *online*, o número de iterações não faz sentido, pois ele tende a infinito. O importante é o comportamento dos algoritmos evolucionários em diferentes condições de mutação da especificação, como mudanças lentas ou rápidas, grandes ou pequenas, de módulo e fase, etc. Serão estudados tanto duas evoluções em paralelo quanto na mesma população de soluções, cujo produto seria um filtro já equalizado.

REFERÊNCIAS

CHEN, H.-C.; CHEN, O. T.-C. Particle swarm optimization incorporating a preferential velocity-updating mechanism and its applications in iir filter design. **IEEE International Conference on Systems, Man, and Cybernetics**, v. 1, p. 1190–1195, 2006.

DINIZ, P.; SILVA, E. da; NETTO, S. **Processamento Digital de Sinais – Projeto e Análise de Sistemas**. [S.l.]: Bookman, 2004.

DORIGO, M.; STÜTZLE, T. **Ant Colony Optimization**. [S.l.]: MIT Press, 2004.

GAO, Y.; LI, Y.; QIAN, H. The design of iir digital filter based on chaos particle swarm optimization algorithm. **Second International Conference on Genetic and Evolutionary Computing**, v. 1, p. 303–306, 2008.

GEEM, Z.; KIM, J.; LOGANATHAN, G. A new heuristic optimization algorithm: Harmony search. **Simulation**, v. 76, n. 2, p. 60–68, 2001.

GOLDBERG, D. **Genetic Algorithms in Search, Optimization and Machine Learning**. Reading, MA: Addison-Wesley, 1989.

HAMMING, R. **Digital Filters**. 3^{rd}. ed. Mineola, NY: Dover Publications, 1997.

HOLLAND, J. **Adaptation in Natural and Artificial Systems**. [S.l.]: University of Michigan Press, 1975.

JIA, D.; JIAO, Y.; ZHANG, J. Satisfactory design of iir digital filter based on chaotic mutation particle swarm optimization. **Third International Conference on Genetic and Evolutionary Computing**, v. 1, p. 48–51, 2009.

KENNEDY, J.; EBERHART, R. Particle swarm optimization. **Proc. of IEEE International Conference on Neural Networks**, v. 4, p. 1942–1948, 1995.

KOZA, J. **Genetic Programming: on the programming of computers by means of natural selection**. [S.l.]: Cambridge: MIT Press, 1992.

KRUSIENSKI, D. J.; JENKINS, W. K. The application of particle swarm optimization to adaptive iir phase equalization. **Acoustics, Speech, and Signal Processing. Proceedings of IEEE International Conference on**, v. 2, p. 693–696, 2004.

LATHI, B. **Modern Digital and Analog Communication**. 3^{rd}. ed. New York: Oxford University Press, 1998.

LEONG W. F. ; YEN, G. . Impact of tuning parameters on dynamic swarms in pso-based multiobjective optimization. **IEEE World Congress on Computational Intelligence**, v. 1, p. 1317 – 1324, 2008.

LUITE, B.; VENAYAGAMOORTHY, G. K. Particle swarm optimization with quantum infusion for the design. **IEEE Swarm Intelligence Symposium**, v. 1, p. 1–8, 2008.

NOCETI FILHO, S. **Filtros Seletores de Sinais**. [S.l.]: DAUFSC, 1998.

OLIVEIRA, D. R. de; PARPINELLI, R. S.; LOPES, H. S. Evolutionary algorithms. In: ____. InTech, 2011. cap. Bioluminescent Swarm Optimization Algorithm, p. 69–84. Disponível em: <http://www.intechopen.com/articles/show/title/bioluminescent-swarm-optimization-algorithm>.

OLIVEIRA, D. R. de et al. Computação evolucionária em problemas de engenharia. In: ____. [S.l.]: Omnipax, 2011. cap. Comparação de técnicas de computação evolucionária para o projetos de filtros digitais de resposta finita ao impulso, p. 107–128.

OPPENHEIM, A.; SCHAFER, R. **Digital Signal Processing**. [S.l.]: Prentice-Hall, 1975.

OPPENHEIM, A.; SCHAFER, R. **Discrete-Time Signal Processing**. [S.l.]: Prentice-Hall, 1989.

PAN, S.-T. Evolutionary computation on programmable robust iir filter pole-placement design. **IEEE Transactions on Instrumentation and Measurement**, v. 60, p. 1469–1479, 2011.

PANAHI, I. M.; VENKAT, K. A new class of invertible fir filters for spectral shaping. **Signal Processing**, v. 89, p. 1271–1287, 2009.

PICEK, S.; GOLUB, M. Comparison of a crossover operator in binary-coded genetic algorithms. **WSEAS TRANSACTIONS on COMPUTERS**, v. 9, p. 1064–1073, 2010. Disponível em: <http://www.wseas.us/e-library/transactions/computers/2010/88-211.pdf>.

TANG, J.; ZHAO, X. A fine tuning hybrid particle swarm optimization algorithm. **International Conference on Future BioMedical Information Engineering**, v. 1, p. 296 – 299, 2009.

TEWOLDE, G.; HANNA, D.; HASKELL, R. Enhancing performance of pso with automatic parameter tuning technique. **Swarm Intelligence Symposium, 2009. IEEE**, v. 1, p. 67 – 73, 2009.

YAMAGUCHI, T.; YASUDA, K. Adaptive particle swarm optimization; self-coordinating mechanism with updating information. **IEEE International Conference on Systems, Man and Cybernetics**, v. 3, p. 2303 – 2308, 2006.

YU, X.; LIU, J.; LI, H. An adaptive inertia weight particle swarm optimization algorithm for iir digital filter. **Artificial Intelligence and Computational Intelligence. International Conference on**, v. 1, p. 114–118, 2009.

ZHANG, X.; JIA, P.; GUO, J. An improved particle swarm optimizer for iir digital filter design. **Intelligent Systems and Knowledge Engineering (ISKE), International Conference on**, v. 1, p. 191–196, 2010.