**UNIVERSIDADE FEDERAL DE SANTA CATARINA**
**DEPARTAMENTO DE ENGENHARIA ELÉTRICA E ELETRÔNICA**

**PROCESSAMENTO DIGITAL DE SINAIS**

# PROJETO DE FILTRO IIR

*Prof. José Carlos Bermudez*
*Lucas Pereira Luiz - 13101258*
*Pedro Henrique Kappler Fornari - 13104320*

**FLORIANÓPOLIS, 08 DE JULHO DE 2016**



## 1 - Introdução

Este relatório irá apresentar o desenvolvimento do projeto de um filtro IIR, resposta ao impulso infinita, passa faixa, com requerimentos especificados pelo professor da disciplina. Serão apresentados os passos que foram seguidos para a obtenção dos resultados do projeto, quais foram as maiores dificuldades no desenvolvimento e os pontos críticos do desenvolvimento, onde o leitor poderá identificar possíveis equívocos, se, por ventura, vier a desenvolver um filtro próprio. O trabalho possui alguns breves trechos da teoria envolvida no projeto de filtros, portanto, para leitores que já conhecem essa teoria, os parágrafos que forem iniciados por [Teoria] não precisam ser lidos para o entendimento do projeto. Além disso, é altamente recomendável que o leitor possua conhecimento da teoria de sinais discretos e sistemas lineares, além de ter realizado uma pré leitura de [1], também é interessante que o leitor, que não conhece a teoria de projeto de filtros, leia [4], pois, ali, uma introdução ao projeto de filtros "não ideais" é feita.

[Teoria] Filtros digitais  com resposta ao impulso infinita podem ser desenvolvidos por meio de protótipos em tempo contínuo. Diferente de filtros FIR, os filtros IIR geralmente são desenvolvidos em três etapas: transformamos as especificações de tempo discreto para tempo contínuo, com isso pode-se utilizar todo o ferramental matemático pronto para projeto de filtros analógicos, passa baixas, para projetar um filtro que respeite as especificações do protótipo analógico e então é feito um mapeamento de frequências para fazer a conversão para tempo discreto. e então transformá-lo em um filtro passa faixas, representado nesse relatório, ou passa altas, rejeita faixa ou filtros multibanda[1]. As Figuras 1.1 e 1.2 ilustram melhor essas etapas.

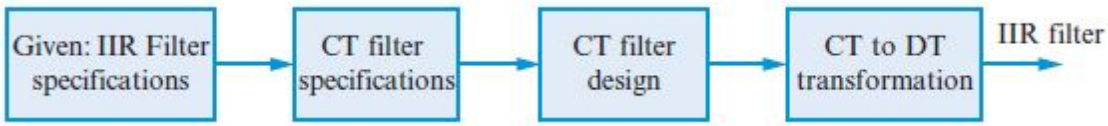



**Figura 1.1 - Processo para desenvolver filtros IIR em tempo contínuo - [1]**

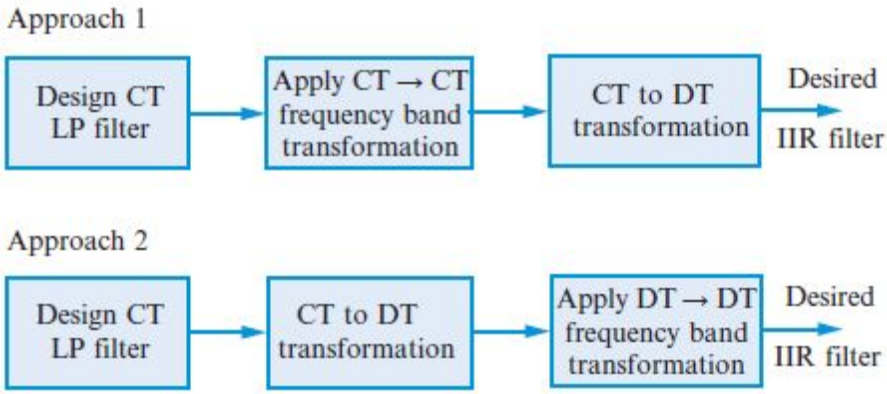



**Figura 1.2 - Transformações em frequência para filtros passa baixa - [1]**

## 2 - Estrutura do Projeto

Seguindo os passos de [3] e utilizando as ferramentas e teorias apresentadas em [1] e [2], é possível verificar quais são os passos para desenvolver o filtro IIR e como implementá-los nas interfaces de desenvolvimento, como o MATLAB nesse caso. Dessa forma, o filtro desenvolvido nesse projeto seguiu o fluxograma apresentado na Figura 2.1.

[Teoria] Inicialmente é feita uma pré distorção, visto que a transformação bilinear gera um efeito conhecido como "frequency wraping", que ocorre porque a conversão das frequências contínuas para discretas da transformação bilinear comprime o eixo das frequências contínuas, de forma não linear, conforme apresentado na página 664 de [1].

[Teoria] Feito isso, as especificações digitais são transformadas em especificações de frequências contínuas que satisfaçam as condições necessárias para se fazer um protótipo de filtro analógico. Para isso, deve-se estabelecer a média geométrica das frequências da especificação em torno da frequência central do filtro, conforme explicado em [3]. Após isso, as especificações do filtro passa faixa com média geométrica estabelecida podem ser normalizadas e transladadas para banda base, fazendo o processo inverso do que é explicado em na seção 2 de [3]. Por fim, com essas novas especificações, é possível, utilizando as aproximações da seção 10.2 de [1], realizar o projeto do protótipo analógico de um filtro de tempo continuo.

[Teoria] O próximo passo é fazer o mapeamento do protótipo passa baixas para as frequências desejadas do filtro passa faixas. Nesse projeto, como solicitado pelo professor, foi feita a transformação bilinear, que, como explicado anteriormente, faz a conversão das frequências contínuas para frequências discretas, mapeando H(s) -> H(z) conforme seção 10.3 de [1]. Da transformação bilinear são tirados os coeficientes do filtro passa faixas, de ordem duas vezes maior que a ordem do protótipo passa baixas, porém, para fins de facilitar a implementação e possíveis correções necessárias quando o filtro for implementado

fisicamente, é interessante dividir esses coeficientes em filtros de segunda ordem em cascata. Além disso, para evitar overflow na saída do filtro, deve-se fazer um escalonamento dos coeficientes conforme seção 9.7 de [2]. Por fim é feita a quantização dos coeficientes de cada estrutura biquadrada, visto que uma implementação real de filtros contará com esse tipo de efeito.

[Teoria] Com isso é possível realizar a verificação do filtro, sendo que, para isso, não é mais possível utilizar a teoria de sistemas lineares, visto os efeitos não lineares introduzidos pela quantização. Logo, o que deve ser feito é uma analise da saída com entradas sinusoidais, pois estas possuem harmônicas principais na frequência de oscilação da onda. Assim, aplicando diversas senoides com frequências entre os limites de banda do filtro, podemos avaliar o ganho aplicado em cada frequência testada, caracterizando assim, de forma indireta, uma resposta em frequência do filtro e comparando essa resposta com as especificações pode ser validada ou não a funcionalidade do filtro. Caso essa resposta não respeite as especificações desejadas, deve-se aumentar a ordem do filtro e reprojetá-lo, até que o mesmo atenda as especificações.

O filtro desenvolvido nesse trabalho, como ilustrado na Figura 2.1, é controlado por uma interface gráfica, chamada *IIR_interface.m*, onde o usuário escolhe qual aproximação utilizar e qual a precisão do DSP onde o filtro será implementado. Feito isso, a função *filtro_IIR.m* é chamada, onde fica a estrutura principal do código. Então é feita a pré distorção das frequências especificadas pelo professor da matéria em [3], pela função *pre_distorcao.m*. Assim, com as frequências previamente distorcidas, é feito o cálculo da média geométrica e o deslocamento das frequências discretas para contínuas. Com isso, é possível fazer o protótipo analógico do filtro, sendo que para isso é feito inicialmente o cálculo da ordem do filtro com uma otimização, que restringe as especificações até que a ordem ultrapasse o menor valor inteiro da estimação, isso porque a ordem estimada pode não ser inteira e é então aproximada para o próximo  número inteiro, assim já é feita uma breve prevenção para as distorções geradas pela quantização dos coeficientes. Isso tudo é feito pela função *CalculaOrdem.m*, e logo após isso, é feito o projeto do protótipo analógico passa baixas por funções do próprio escopo do MATLAB.

Com o protótipo analógico, utilizamos a função do MATLAB *lp2bp.m*, onde os coeficientes do filtro analógico passa baixas são transformados nos coeficientes do filtro analógico passa altas e então a função *bilinear.m* é utilizada para fazer a conversão dos coeficientes do filtro passa faixa analógico para digital. Então, utilizando a função *tf2sos.m* foi feito a separação do filtro em biquadradas escalonadas.

Após isso, é feita a quantização dos coeficientes, com a função *quantizador.m* onde os coeficientes são quantizados segundo o número de bits escolhido pelo usuário e um nível de saturação, não necessariamente igual a 1, dado pelo valor do maior coeficiente do filtro. Também são aplicados dois ganhos nos coeficientes, um, chamado de *deslocamento*, para aplicar uma leve atenuação no ganho do filtro para banda passante, criando assim uma leve folga com a restrição superior do ripple da banda passante, visto que as aproximações sempre tinham leve oscilação de ganho muito próximo a 0dB, e qualquer valor estritamente maior que isso já faz com que o filtro não respeite o gabarito. Dessa forma, o filtro que antes estava

bastante rente ao ganho unitário tem uma folga maior para oscilações de ganho maior do que um. O outro ganho, chamado de *g*, é aplicado devido a distorção introduzida pela separação em biquadradas e escalonamento. Toda quantização e aplicação de ganhos ocorre dentro da função *filtragem_quantizada.m*, apesar disso os ganhos são apenas calculados na função principal *filtro_IIR.m*.

Assim o filtro está implementado, agora é iniciada a fase de testes do filtro. Na função *teste_filtro.m* as senoides de teste são criadas ao longo do espectro de frequências do filtro. Essas senoides são então aplicadas no filtro, por meio da função *filtragem_quantizada.m* e então a harmônica principal da saída, correspondente ao ponto de teste é armazenada, criando um vetor com todas as harmônicas testadas e o ganho aplicado em cada senoide pelo filtro. Após isso, esse vetor é adaptado para dois vetores, um com os valores de ganho do filtro na banda passante e outro com os valores de ganho na banda de rejeição. Estes ganhos são então testados com as especificações. Caso as especificações sejam obedecidas o teste retorna filter_ok =1, caso contrário retorna filter_ok = 1.

Assim, caso o filtro não obedeça as especificações, a ordem é incrementada e o filtro é reprojetado a parir da função *CalculaOrdem.m* onde, nesse caso é feita apenas uma otimização. Porém quando a ordem do filtro é incrementada mais de 5 vezes o filtro é considerado não implementável e o código é abortado.

Por fim, com o filtro pronto ou não, é feito o calculo da distorção harmônica aplicada pelo filtro nas senoides, com a função *filter_thd.m*, e são apresentados os gráficos de magnitude e fase do ganho do filtro, com a função *filter_plot.m*. O resultado da implementação do filtro é apresentado no campo *filtro implementado?* de forma que o usuário esteja ciente do que ocorreu, assim como a ordem e a distorção harmônica.

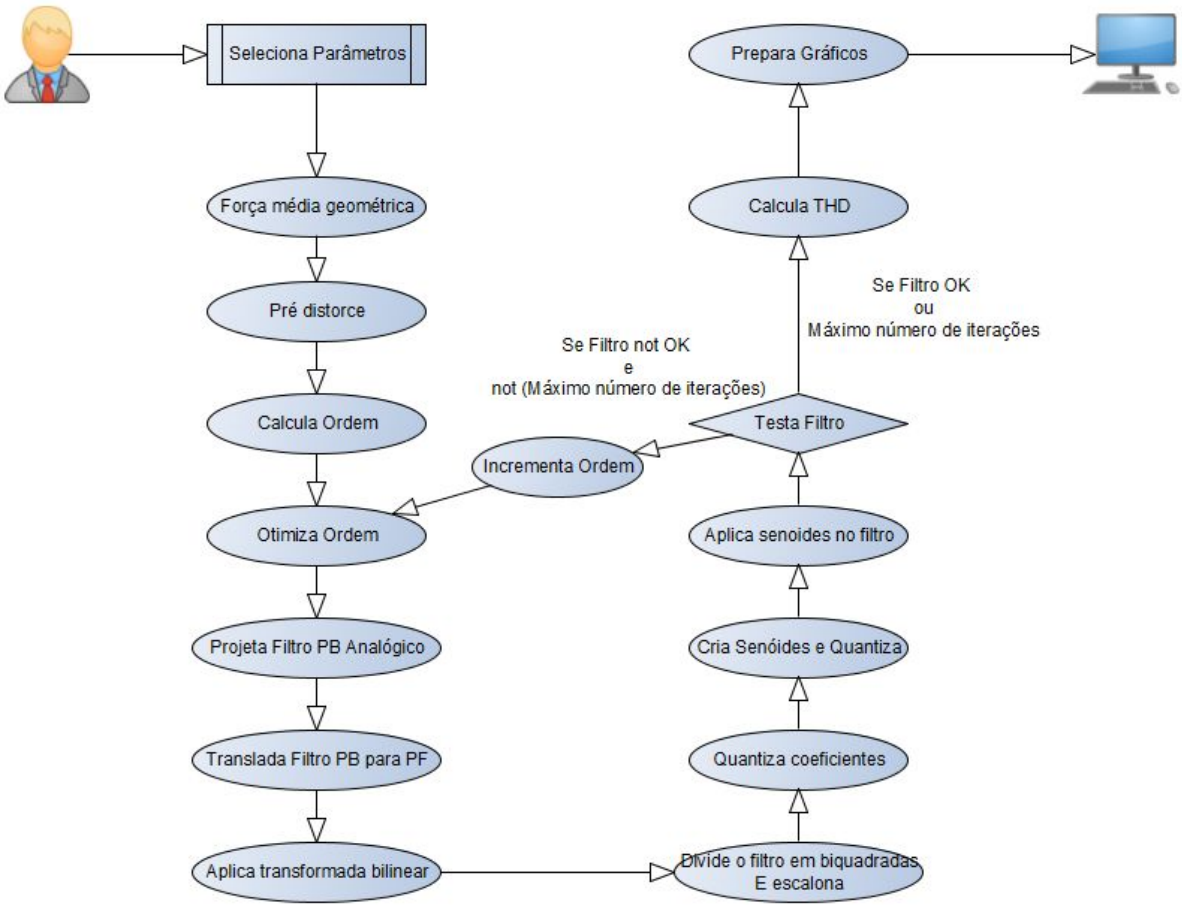



**Figura 2.1 - Fluxograma do código desenvolvido no projeto.**

## 3 - Metodologia

Para analisar as características gerais do programa, o código foi feito de uma maneira bastante modular, com diversas funções separadas, facilitando a visualização e o debug do código. Também foram salvas variáveis de controle, para que fosse possível apresentar resultados a quem estivesse rodando o código principal. Foram criadas algumas variáveis para controle do projeto, decididas pelos alunos, como o número de pontos das transformadas de fourrier chamada Nfft, nesse caso 20000; o número de senoides a serem testadas chamada de pontos_teste, que nesse caso é de 40 pontos ao longo de todo o espectro; na função *CalculaOrdem.m* o quinto argumento é um passo utilizado na otimização da ordem, nesse caso vale 0.001; o deslocamento aplicado na banda passante foi escolhido de forma que a metade da resposta ideal na banda passante fique na metade das especificações da banda passante, porém é mais fácil de visualizar isso observando os gráficos da resposta do filtro; também, como citado anteriormente nesse relatório foi implementada a variável *inc_order* que controla um numero máximo de incrementos de ordem para tentar implementar o filtro, nesse projeto o número máximo foi de 5 incrementos. Variando estes parâmetros, principalmente o número de pontos de teste e o número máximo de incrementos, é possível obter resultados diferentes dos apresentados na seção de resultados deste relatório, visto que o

teste com senoides pode deixar alguns pontos sem serem testados, podendo assim existir algum ponto do filtro que não foi testado e que não respeita as especificações.

Com isso, variando a precisão entre 12 e 16 bits, como instruído pelo professor, cada aproximação do filtro, Butterworth; Chebyshev; Chebyshev inverso e Elíptica, foi avaliada a resposta. Essas respostas serão apresentadas na próxima seção desse relatório com gráficos e conclusões sobre cada implementação.

## 4 - Resultados

Ao executar-se o programa *IIR_Interface.m* a interface da Figura 4 aparece para o usuário selecionar a aproximação e a precisão desejada para o filtro. Todos os gráficos obtidos estão presentes no anexo, nas Figuras 6.1 à 6.8.

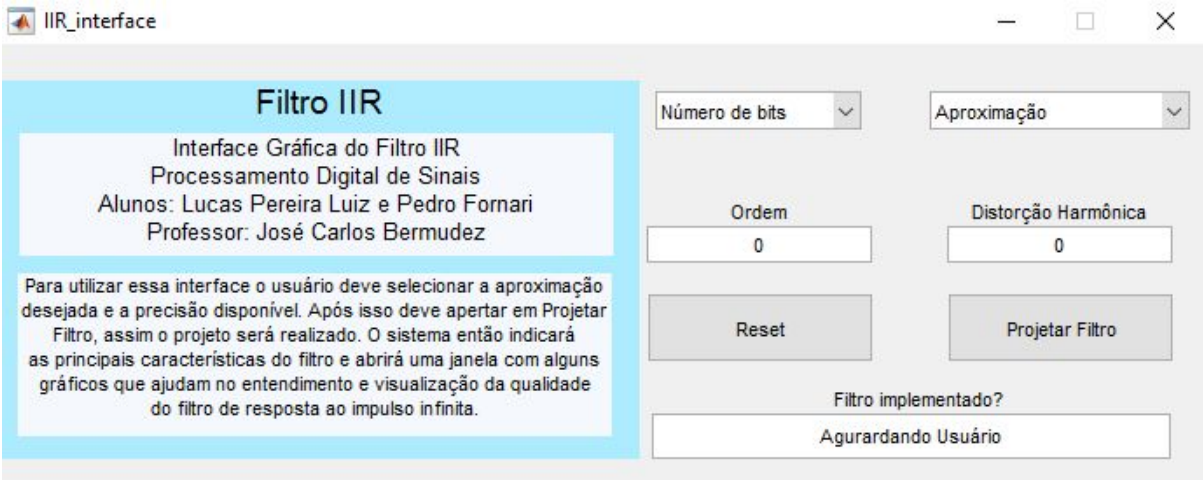



**Figura 4.1 - Interface gráfica para o projeto de filtro IIR**

Executando-se o programa para todas as combinações de ordem e janela, foram obtidos os valores dos parâmetros *ordem*, *THD*, *folga mínima na banda passante* e *folga mínima na banda de rejeição*, que são discutidos a seguir e apresentados nas tabelas 4.1 à 4.4.

É possível observar que todas as combinações foram implementadas, como pode ser visto no anexo, e todas, menos a Butterworth 12 bits, apresentaram um THD menor que 1%.

A aproximação elíptica apresentou-se como a melhor no quesito ordem mínima para as precisões testadas.

Avaliando-se apenas o THD, nota-se que a Chebyshev apresenta o melhor para resolução de 12 bits, porém vale lembrar que ele possui ordem maior que a da Elíptica, por isso uma avaliação de apenas um parâmetro não pode ser feita.

Todos apresentaram folgas espectrais relativamente grandes tanto da banda passante quanto na de rejeição, exceto o Butterworth 12 bits, que apresentou o pior desempenho.

| Aproximação\Precisão | 12 Bits | 16 Bits | Infinita |
|---|---|---|---|
| Butterworth | 12 | 12 | 10 |
| Chebyshev | 10 | 10 | 8 |
| Chebyshev Inverso | 8 | 8 | 8 |
| Elíptica | 6 | 6 | 6 |

**Tabela 4.1 - Ordem dos filtros**

| Aproximação\Precisão | 12 Bits | 16 Bits |
|---|---|---|
| Butterworth | 2.27507% | 0.147692% |
| Chebyshev | 0.0449028% | 0.00398818 |
| Chebyshev Inverso | 0.123655% | 0.00914143% |
| Elíptica | 0.0748021% | 0.00254853% |

**Tabela 4.2 - THD**

| Aproximação\Precisão | 12 Bits | 16 Bits | Infinita |
|---|---|---|---|
| Butterworth | 0.008dB | 0.35dB | 0.37985dB |
| Chebyshev | 0,1915dB | 0.1938dB | 0.4806dB |
| Chebyshev Inverso | 0.2821dB | 0.3765dB | 0.3857dB |
| Elíptica | 0.3276dB | 0.3359dB | 0.3399dB |

**Tabela 4.3 - Folga mínima banda passante**

| Aproximação\Precisão | 12 Bits | 16 Bits | Infinita |
|---|---|---|---|
| Butterworth | 0.8125dB | 3.6376dB | 1.5470dB |
| Chebyshev | 4.5487dB | 4.6077dB | 2.4056dB |
| Chebyshev Inverso | 1.5370dB | 1.5956dB | 1.5843dB |
| Elíptica | 1.2793dB | 1.4108dB | 1.3989dB |

**Tabela 4.4 - Folga mínima banda rejeição**

## 5 - Conclusão

Este trabalho, assim como o projeto do Filtro FIR, auxiliou muito no compreendimento da teoria de projeto de filtros e no uso da ferramenta de software para auxilio a simulações e projetos como o MATLAB. Com ele foi possível melhorar nossas habilidades e lógicas para implementação de testes e funções de otimização, além de organização de estrutura de código. Ainda foi bastante interessante verificar as diferenças entre filtros FIR e IIR, inclusive na implementação, como a separação do filtro em pequenos filtros de ordem menor, a implementação de estruturas recursivas e as transformações de filtros digitais para analógico e analógico para digital, a fim de aproveitar o ferramental matemático existente.

Novamente avaliamos os efeitos da não linearidade e como esse problema afeta ainda mais os filtros IIR, visto que o erro aumenta devido a recursividade. Outros pontos interessantes avaliados foi a forma como as aproximações implementam os filtros bastante rente às especificações de ganho, e que por isso tivemos que aplicar um deslocamento, afim de auxiliar no respeito das especificações.

Os resultados obtidos foram bastante interessantes, já que todos os filtros puderam ser implementados. O THD se manteve razoavelmente baixo, exceto para o Butterworth 12 bits, como já comentado. Notou-se também que tanto o Chebyshev inverso quanto o Elíptico não incrementaram a ordem para uma menor resolução de bits, sem perder muito em folga espectral. As folgas ficaram bastante justas após a quantização, devido ao fato da não linearidade, principalmente para 12 bits, visto que quanto menor a precisão finita, maior a distorção introduzida. Porém esses resultados ainda podem ser contestados, visto que o número de pontos que utilizamos para testar todo o espectro de frequências foi apenas de 40 senoides, antes da implementação de um filtro real deve-se analisar ainda mais pontos, a fim de obter uma maior segurança.

Algumas outras formas de projetar filtros IIR com funções mais prontas do MATLAB seriam interessantes para trabalhos futuros, como filterdesign ou yulewalk.m que fazem diretamente o projeto do filtro. Assim poderíamos analisar as diferenças entre os filtros projetados utilizando essas ferramentas e os que desenvolvemos, porém assim a teoria de filtros ficou mais clara e poderíamos avaliar quais efeitos positivos e negativos são introduzidos por essas funções. Ainda existem também outras formas para a otimização e correção do filtro, que não foram abordadas nesse projeto, mas que com o estudo mais aprofundado na teoria de filtros seria mais facilmente aplicado, como por exemplo, estudar como quantizar os coeficientes do filtro entre -1 e 1, independendo da saturação máxima, o que evitaria overflow em dispositivos de processamento.

## 5.1 - Considerações Finais

Este projeto implementa o filtro com as especificações passadas pelo professor orientador em [3], e por isso não permite que o usuário da interface altere estes valores,

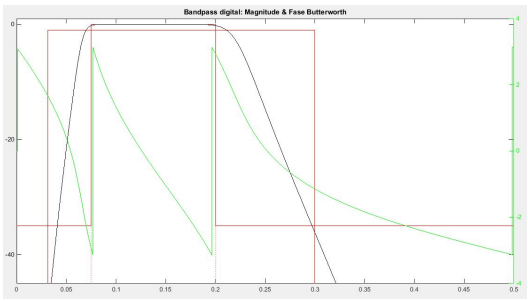
porém, caso o leitor deseje pode alterar as especificações no início do arquivo filtro_IIR, tomando cuidado com a média geométrica.

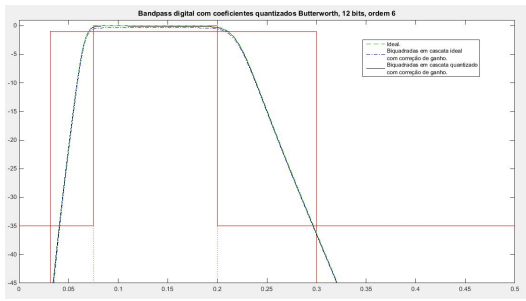
Ainda, como citado anteriormente, parâmetros foram utilizados como decisão de projeto, como o número de pontos de teste por exemplo, o que traz uma confiabilidade melhor do filtro, porém aumenta o tempo de execução e possivelmente a ordem do filtro. O valor do incremento máximo da ordem também é importante de ser verificado, para evitar erros devido a esse parâmetro, logo se aumentar o número de pontos de teste provavelmente o usuário deverá aumentar o número de incremento máximo.

É válido salientar também que a ordem apresentada na interface indica a ordem do protótipo analógico do filtro passa baixas, ou seja, a ordem do filtro passa faixas digital é o dobro da ordem apresentada, visto que ao transformar um passa baixas em passa faixa a ordem dobra pois o número de coeficientes também dobra.

Por fim é válido lembrar que antes da execução do projeto, o usuário deve selecionar uma das aproximações e uma precisão, caso contrário o programa apontará um erro, devido a não seleção dos parâmetros necessários e não rodará. Também é recomendado que o usuário reinicie a interface, apertando em *Reset* cada vez que projetar um novo filtro, a fim de evitar que qualquer variável mantida do projeto anterior interfira no novo projeto.

# 6 - Anexos



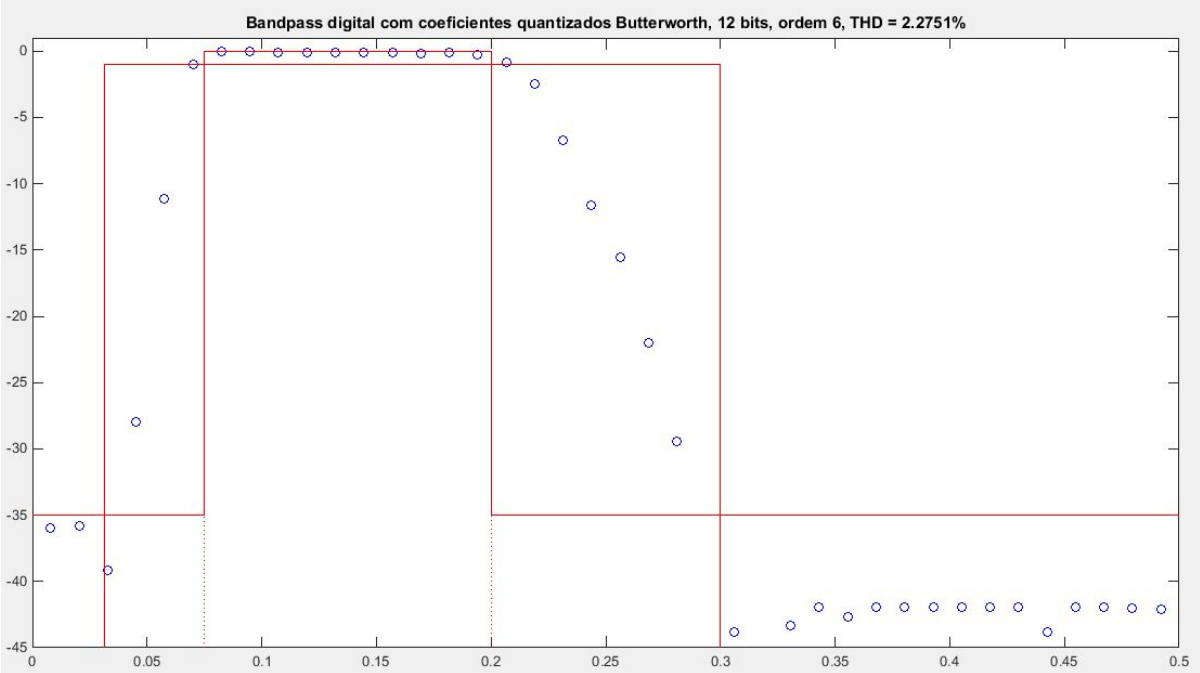
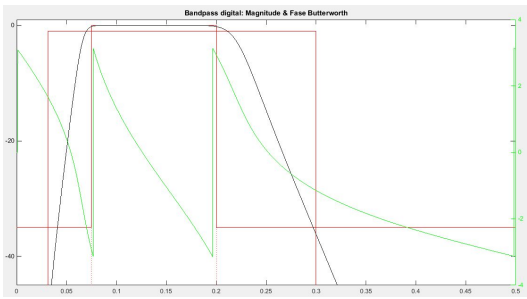
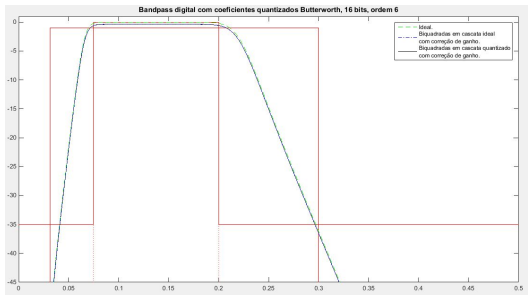




**Figura 6.1 - Butterworth 12 bits**



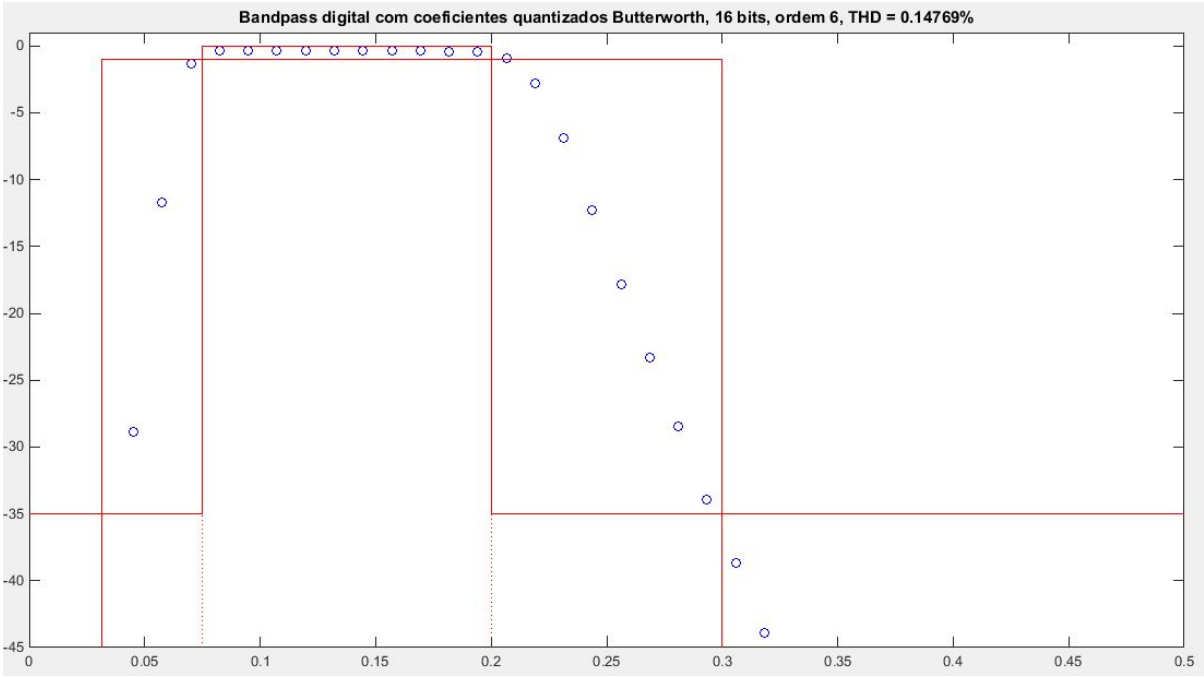
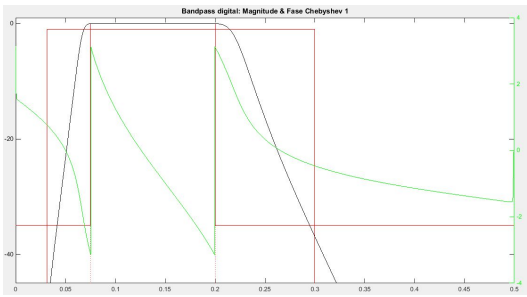
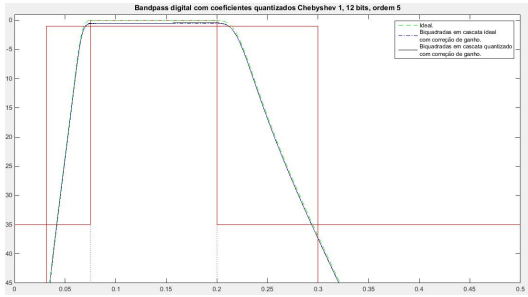




**Figura 6.2 - Butterworth 16 bits**



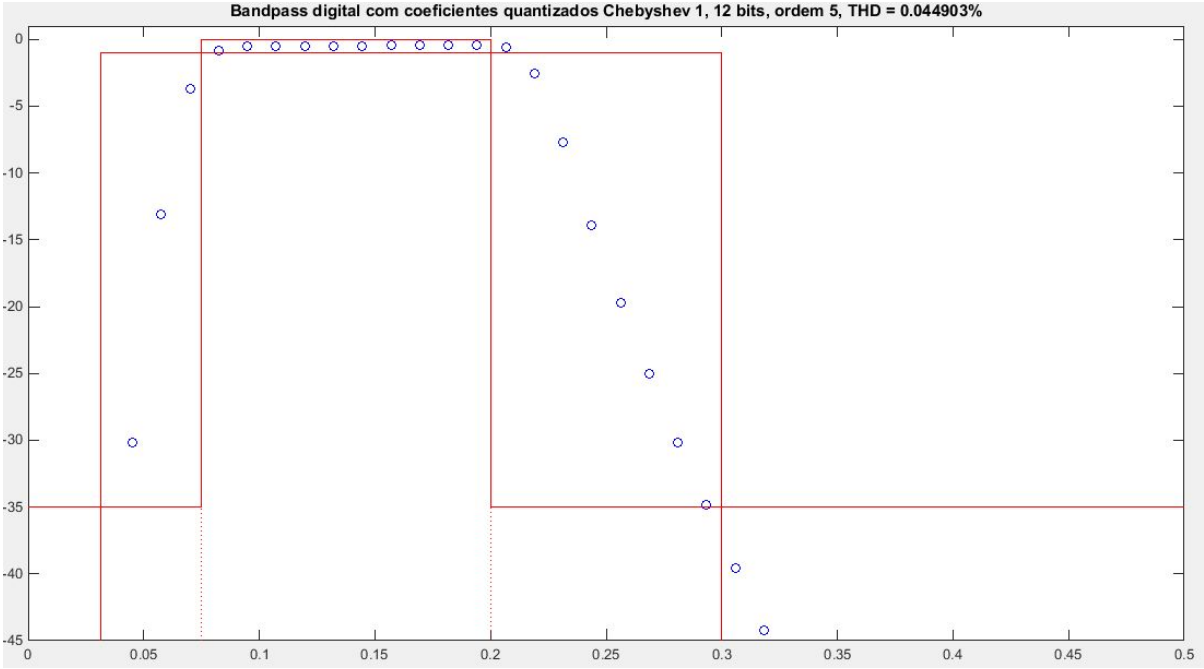
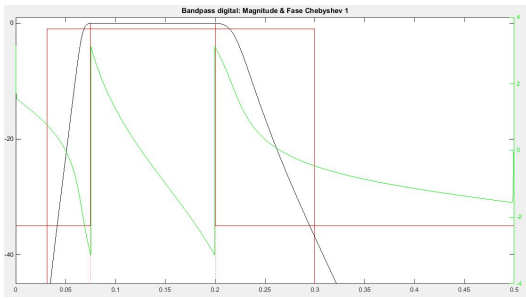
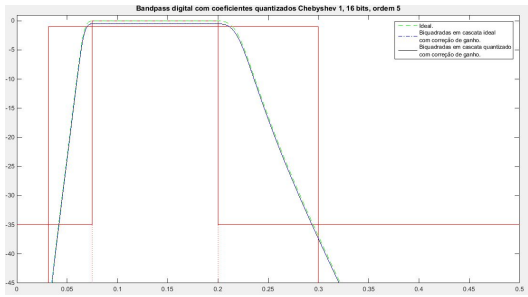




**Figura 6.3 - Chebyshev 12 bits**



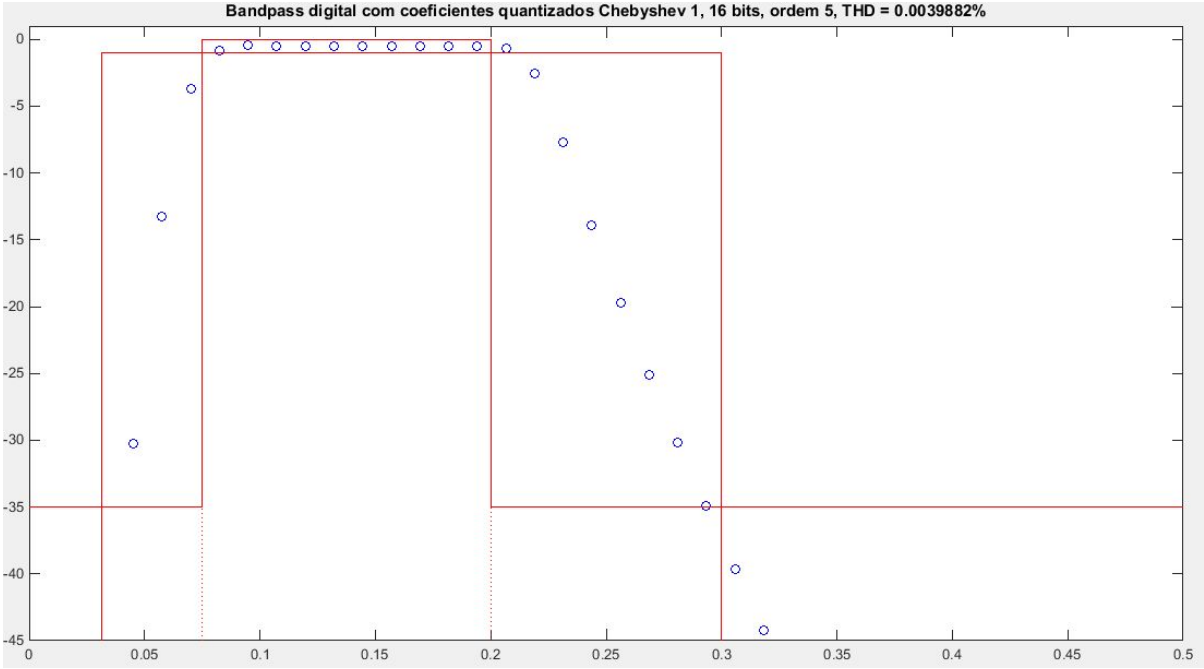
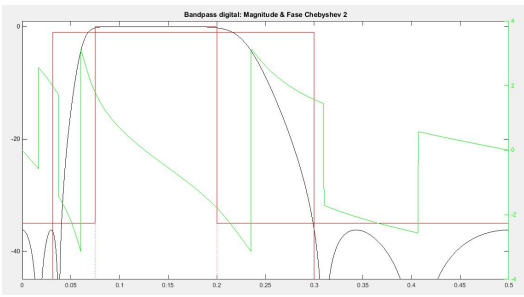
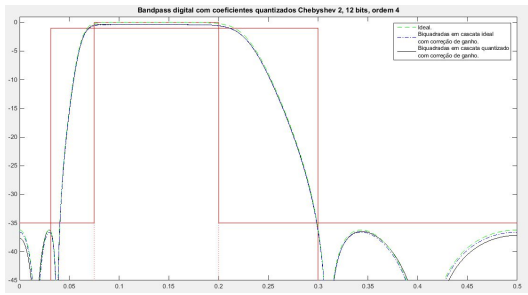




**Figura 6.4 - Chebyshev 16 bits**



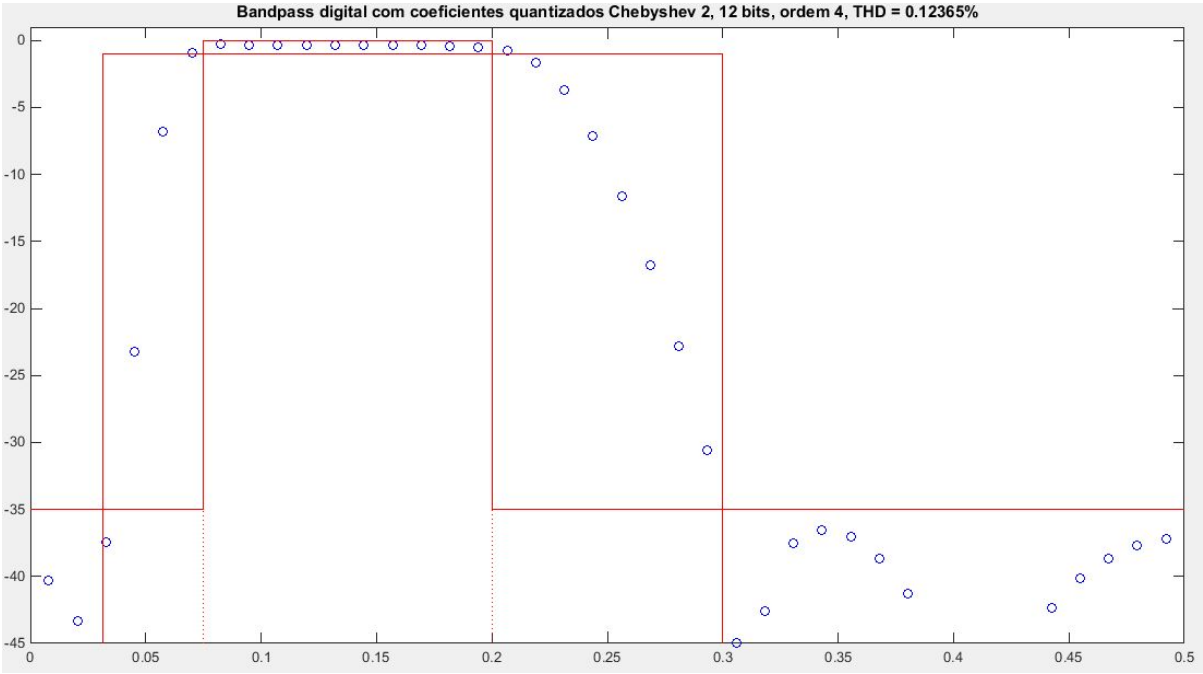
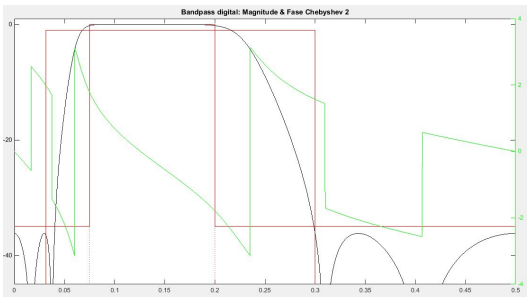
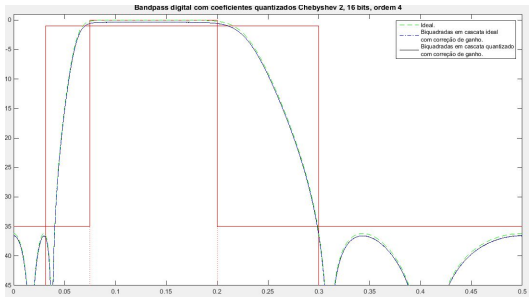




**Figura 6.5 - Chebyshev Inverso 12 bits**



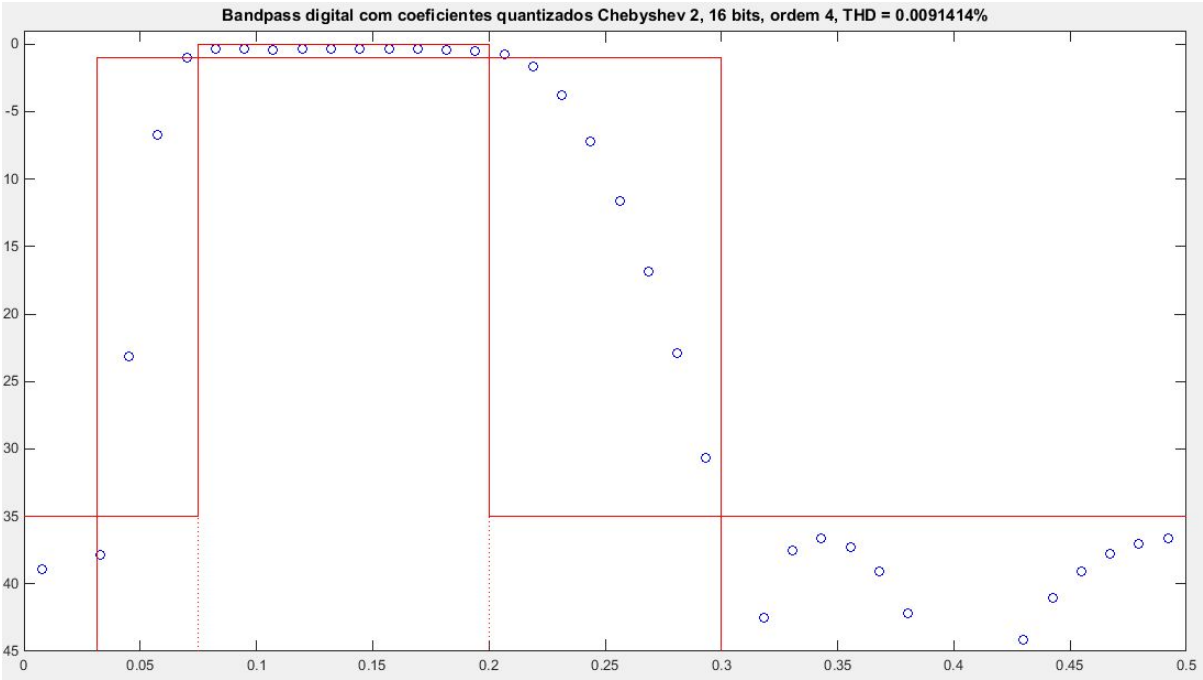
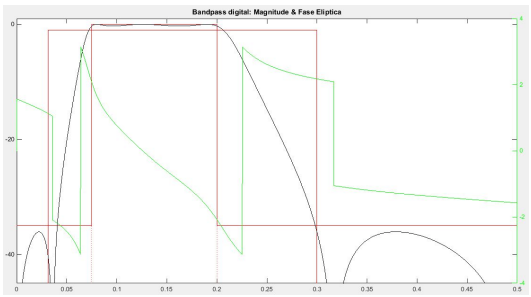
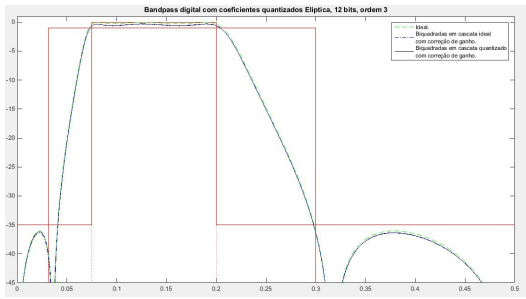




**Figura 6.6 - Chebyshev Inverso 16 bits**



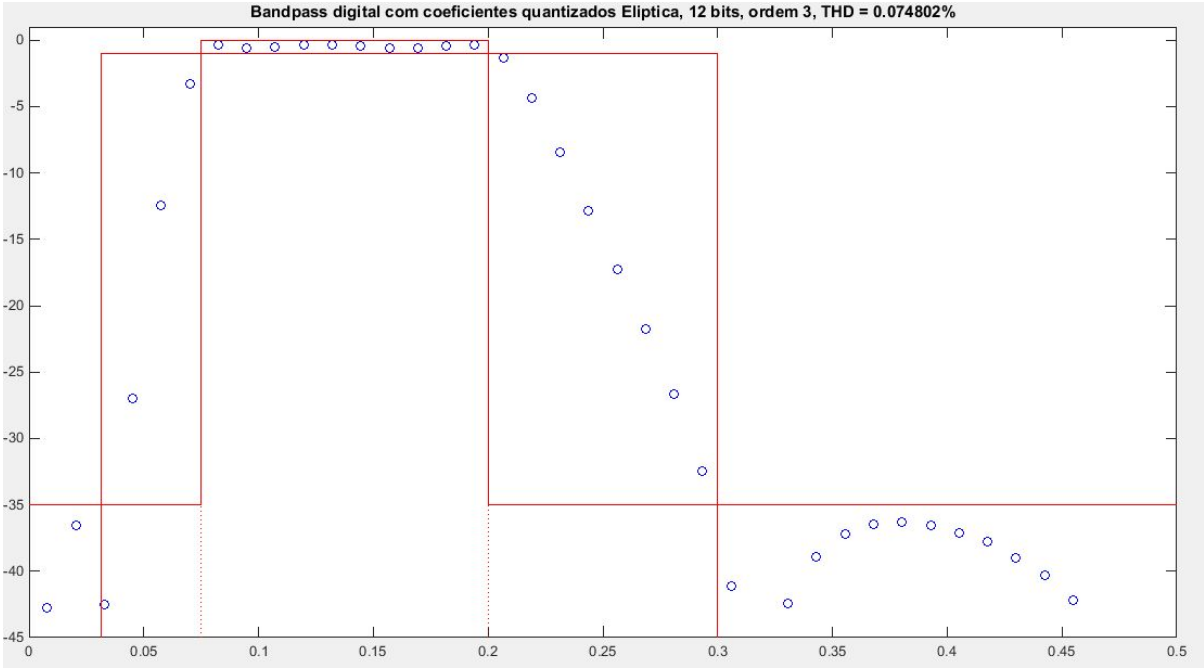




**Figura 6.7 - Elíptico 12 bits**

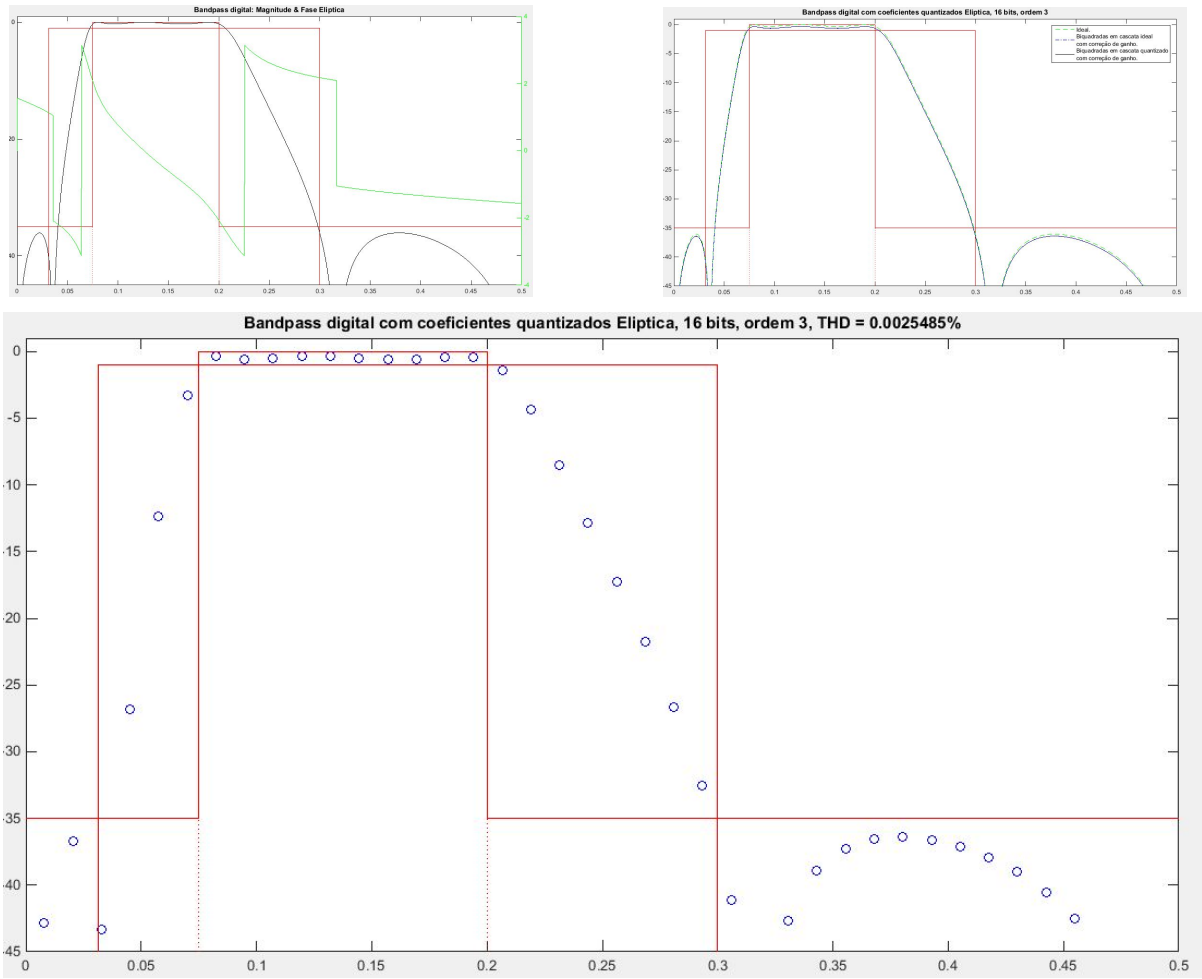



**Figura 6.8 - Elíptico 16 bits**

# 7 - Referências

[1] - Capítulo 11 Seção 1, D. G. Manolakis and V. K. Ingle, Applied Digital Signal Processing. Cambridge University Press, 2011.
[2] - Capítulo 10 Seções 1 a 3, D. G. Manolakis and V. K. Ingle, Applied Digital Signal Processing. Cambridge University Press, 2011.
[3] - Etapas do Projeto PDF, José Carlos Bermudez 2016.
[4] - Introdução, L. P. Luiz and P. H. K. Fornari, Relatório FIR, 2016