UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA E ELETRÔNICA

PROCESSAMENTO DIGITAL DE SINAIS
PROJETO DE FILTRO FIR USANDO JANELAS

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

FLORIANÓPOLIS, 06 DE JUNHO DE 2016

1 - Introdução

Este relatório irá apresentar o desenvolvimento do projeto de um filtro FIR, resposta ao impulso finita, por janelamento e passa baixas, 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 [2].

[Teoria] Filtros digitais seletivos em frequência, quando ideais, não são realizáveis devido à transição abrupta da frequência de passagem para a de rejeição. Isso não é implementável, visto que não existe maneira de computar uma resposta ao impulso de infinitos números[1]. Por isso projetos de filtros levam em conta especificações pré definidas, que satisfaçam a aplicação na qual o filtro será inserido[2]. Estas especificações fazem com que não exista nenhuma frequência rejeitada completamente, e que exista uma banda de transição, não imediata, entre a banda de passagem e a banda de rejeição[2]. As especificações definem então qual será o tamanho da banda de transição, qual a oscilação máxima no ganho de banda passante e qual a rejeição mínima para frequências contidas na banda de rejeição do filtro[2]. Além disso, esse projeto aborda os efeitos da não linearidade dos sinais discretos quantizados, fator importante quando é desenvolvido um filtro real, visto que a memória dos dispositivos de processamento tem limitação de bits.

O desenvolvimento desse projeto seguirá o fluxo sugerido pelo livro texto da disciplina[2] e os requerimentos de projeto[3], disponibilizados pelo professor, passando pelos seguintes tópicos: especificação, aproximação, quantização, verificação e implementação[2].

2 - Estrutura do Projeto

Como apresentado anteriormente, o desenvolvimento do filtro deve passar pelas cinco etapas citadas no último parágrafo da introdução desse relatório. As especificações do projeto foram fornecidas pelo professor[3].

[Teoria] O projeto deve, inicialmente, estimar a frequência de corte ideal e a ordem inicial do filtro, que corresponde ao número de amostras da resposta ao impulso do filtro, com isso, a etapa de aproximação pode ser realizada truncando a resposta ao impulso do filtro ideal com o número de coeficientes referente a ordem do filtro[2]. Feito isso, deve-se aplicar o janelamento para obter os coeficientes do filtro "ideal" e verificar se este obedece as especificações[2], caso contrário, deve-se voltar a estimativa de ordem e a seleção da frequência de corte, fazendo uma nova aproximação, até que o filtro "ideal" obedeça os requisitos[3]. Após isso, tem-se os coeficientes do filtro "ideal", porém, este, não necessariamente, é implementável em DSP's, microcontroladores ou até mesmo computadores, visto que esses dispositivos possuem precisão finita, por isso quantiza-se os coeficientes do filtro ideal, obtendo um filtro digital realizável. Por fim, para a implementação do filtro, deve ser feita inicialmente a verificação dos parâmetros. Para isso, aplica-se diversas senoides com frequências diferentes pelo filtro e observa-se a resposta[3], pois o filtro tem efeitos não lineares, logo não é possível utilizar as teorias de sistemas lineares para esse teste. Analisa-se a frequência da primeira harmônica de cada senoide à saída em comparação com sua respectiva entrada para se levantar a resposta em frequência do filtro. Com diversas senoides, é possível fazer uma boa verificação do filtro[3]. O levantamento da resposta em frequência deve ser repetido, juntamente com o aumento de ordem e variação de wc, até que o filtro obedeça os requisitos. Então, finalmente o filtro é implementado.

A implementação desse filtro seguiu o fluxograma apresentado na Figura 2.1 deste relatório. Inicialmente, o usuário executa o arquivo *Fir_Interface.m*, onde abrirá uma interface para selecionar os parâmetros possíveis para a implementação, sendo, nesse caso, a janela utilizada e o número de bits. Feito isso, o programa *FIR_filter.m* estima a ordem inicial com a função *estimativaordem.m*, de acordo com a tabela 10.3 de [2], e um valor inicial para a frequência de corte do filtro "ideal". com esses valores, é feito o truncamento da resposta ao impulso do filtro "ideal", com a função *respostatruncada.m*. Após isso são calculados os coeficientes da janela escolhida, com a função *coefjanelas.m*, que faz uso das funções do matlab fornecidas pela toolbox do livro, apresentadas em [2], e então esses coeficientes são multiplicados, termo a termo, pelas amostras coletadas da resposta ao impulso. Com isso obtém-se filtro não quantizado, então, para verificar seu funcionamento, utiliza-se a função *filter_improvment.m*, onde é calculada a transformada de fourier do filtro e compara-se o módulo do filtro com os pré-requisitos, verificando se, na banda passante, o ripple não ultrapassa os limites e se, na banda de rejeição, a atenuação mínima é alcançada. Caso isso não seja verdade, o teste varia wc entre as frequências limites (wp e ws), tentando encontrar algum wc que satisfaça os requisitos, caso isso não ocorra, o programa aumenta a ordem, sendo que, cada vez que o teste é realizado, os coeficientes do filtro são recalculados.

Esses coeficientes são então quantizados, com a função *quantizador.m*, com precisão relativa ao número de bits escolhido pelo usuário. A função *filter_improvment.m* é chamada novamente, agora fazendo a otimização do filtro com coeficientes quantizados, mas ainda de maneira linear. O teste é similar ao já descrito para coeficientes não quantizados.

Na função *filter_improvment_quant.m* são criadas diversas senoides para realizar o teste do filtro de precisão finita, que simularão entradas do filtro, a fim de verificar os efeitos da não linearidade. A função *MAC.m* realiza a convolução dos coeficientes do filtro com os coeficientes de cada senoide. A saída do filtro é então quantizada e transformada, assim é possível verificar se a amplitude do lóbulo da harmônica principal de cada senoide de saída do filtro quantizado respeita os pré requisitos, caso contrario, a função *filter_improvment_quant.m* recalcula os coeficientes quantizados com diferentes wc's e ordens, repetindo o teste a cada vez. Assim, quando obtém um resultado satisfatório, a função *filter_thd.m* calcula a distorção harmônica da saída, de forma que seja possível comparar os resultados de diferentes números de bits e janelas.

Por fim são plotados gráficos e valores que interessam o projetista para identificar o resultado de seu trabalho. Esses gráficos são: magnitude e fase da resposta "ideal", magnitude e fase da resposta com coeficientes quantizados, e a magnitude da análise não linear do filtro (usando senoides para pegar os pontos). Os valores plotados na interface são: ordem do filtro ideal, ordem final do filtro prático, frequência de corte ideal, frequência de corte final do filtro prático, distorção harmônica, em porcentagem e se o filtro prático foi implementado de fato ou não.

Também é válido informar que as funções em anexo desse trabalho estão detalhadamente comentadas, para facilitar o leitor no entendimento de como são realizadas as operações descritas acima. Além disso a notação de ordem nesse projeto é M, onde a resposta ao impulso é truncada de -M/2 até M/2.

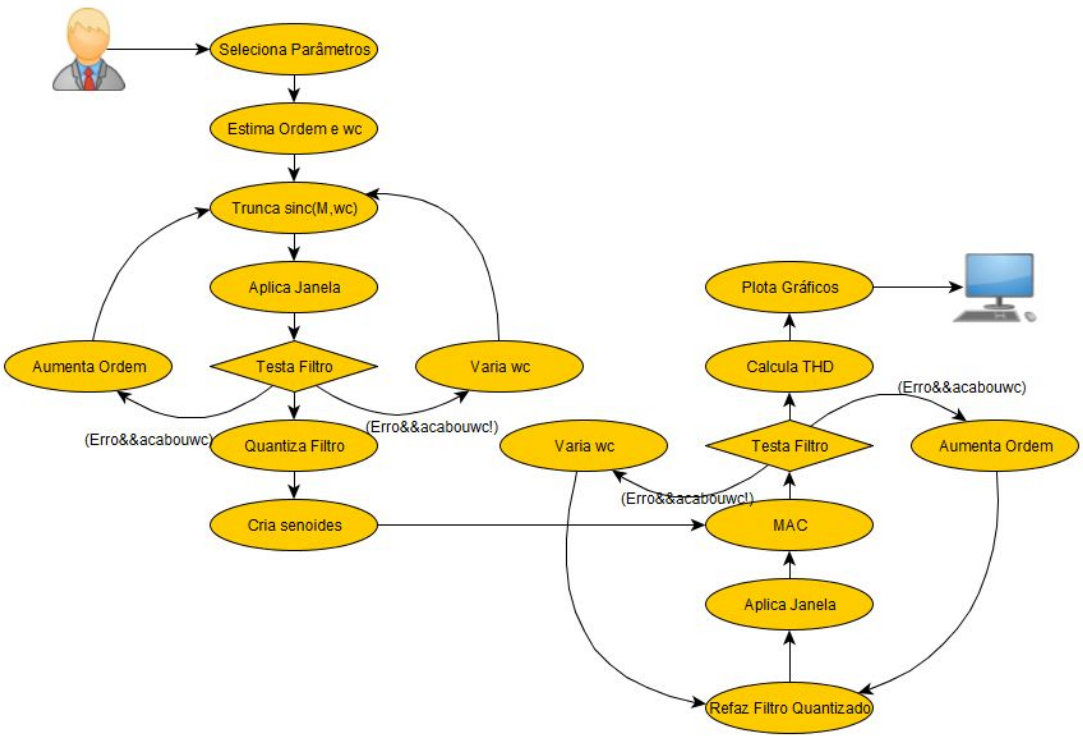

Figura 2.1 - Fluxograma do Funcionamento do Programa

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. Foi desenvolvido um controle de máximo número de tentativas, de forma que, ao alcançar determinado número de tentativas no incremento da ordem o código trava e apresenta resultados incorretos, indicando que é impossível implementar o filtro com o número de bits e janela respectivos a escolha do usuário.

Por fim, foram analisados testes com as quatro janelas pré determinadas em [3] utilizando precisão de 8, 10 e 32 bits para cada janela diferente. Os resultados desses testes serão apresentados na próxima seção deste relatório, apresentando a ordem de cada filtro, a distorção harmônica da saída, e gráficos de magnitude e fase da resposta em frequência dos filtros.

4 - Resultados

Ao executar-se o programa *Fir_Interface.m* a interface da imagem 4.1 aparece para o usuário selecionar a janela e a precisão desejada para o filtro. Todos os gráficos obtidos estão presentes no anexo, nas imagens 6.1 à 6.16.

Imagem 4.1 - Interface Gráfica

Executando-se o programa para todas as combinações de ordem e janela, foram obtidos os valores para ordem presentes na tabela 4.1.

Como pode ser visto, nenhum filtro consegue atender às especificações com precisão de apenas 8 bits, devido aos efeitos não lineares da quantização, mas mesmo assim o programa projeta um filtro com a maior ordem que ele consegue chegar antes de declarar como não implementável para fim de observação do resultado. Esses filtros projetados podem ser vistos no anexo.

Para precisão de 10 bits, apenas a janela retangular não consegue ser implementada, já as outras janelas conseguem. A que apresentou menor ordem foi a Kaiser

Todos os filtros podem ser implementados com precisão de 16 bits, porém a ordem da janela Retangular é muito maior que as outras, a melhor continua sendo a Kaiser.

Para precisão infinita, todas as janelas são implementadas e a Kaiser apresenta a menor ordem.

JANELA/BITS	8 BITS	10 BITS	16 BITS	INFINITO
RETANGULAR	NÃO IMPLEMENTÁVEL	NÃO IMPLEMENTÁVEL	226	226
HAMMING	NÃO IMPLEMENTÁVEL	28	28	28
BLACKMAN	NÃO IMPLEMENTÁVEL	38	38	38
KAISER	NÃO IMPLEMENTÁVEL	26	26	26

Tabela 4.1 - Ordem dos Filtros

A partir deste momento, serão apresentados dados apenas para implementações de 10 bits, 16 bits, e precisão infinita pois 8 bits não é implementável.

Para análise das folgas espectrais, foi pego o maior ripple para cada configuração de filtro (tanto em banda passante quanto em banda de rejeição) e foi comparado com os valores da especificação, obtendo-se os valores das tabelas 4.2 (para banda passante) e 4.3 (para banda de rejeição). A janela retangular com 10 bits não foi analisada por não ser implementável.

Percebe-se que a janela Kaiser apresenta as melhores folgas em banda passante tanto para precisão infinita quanto para 16 bits. Para 10 bits a janela Blackman apresentou a maoir folga de todas em banda passante, porém a pior em precisão infinita.

JANELA/BITS	10 BITS	16 BITS	INFINITO
RETANGULAR	NÃO IMPLEMENTÁVEL	0,0031826 dB	0,0040547 dB
HAMMING	0,0263866 dB	0,0042996 dB	0,0045880 dB
BLACKMAN	0,0483761 dB	0,0033945 dB	0,0000799 dB
KAISER	0,0121832 dB	0,0170752 dB	0,0174817 dB

Tabela 4.2 - Folga Espectral em Banda Passante dos Filtros

Analisando-se a folga na banda de rejeição, nota-se que é muito maior que em banda passante. A janela Hamming apresenta os melhores resultados nas precisões apresentadas, e a janela retangular apresenta as piores.

JANELA/BITS	10 BITS	16 BITS	INFINITO
RETANGULAR	NÃO IMPLEMENTÁVEL	0,1585663 dB	0,0907185 dB
HAMMING	6,9492716 dB	7,9193690 dB	7,9481665 dB
BLACKMAN	6,2923318 dB	4,5231014 dB	4,8039245 dB
KAISER	6,3009395 dB	6,0970235 dB	6,0777671 dB

Tabela 4.2 - Folga Espectral em Banda de Rejeição dos Filtros

Analisando-se o THD das implementações pela tabela 4.2, nota-se que houve um aumento de cerca de 70 vezes ao passar de 16 bits para 10 bits para todas as janelas, exceto é claro, para a retangular, que não é implementável. Como 8 bits não é implementável também, não foi feita análise de THD.

Pode-se observar que o THD não muda muito de uma janela para outra, porém deve-se lembrar que as ordens são diferentes. A janela Kaiser apresenta uma boa relação entre ordem e THD comparando-se com as outras janelas, porém, analisando-se preferencialmente o THD, a janela Hamming (com ordem 28 e 16 bits) apresenta o melhor resultado.

JANELA/BITS	10 BITS	16 BITS
RETANGULAR	NÃO IMPLEMENTÁVEL	0.000671791%
HAMMING	0.0480499%	0.00067292%
BLACKMAN	0.0495753%	0.000718877%
KAISER	0.050603%	0.000701994%

Tabela 4.4 - THD dos Filtros

5 - Conclusão

Desenvolver este projeto trouxe um entendimento muito satisfatório da teoria de projeto de filtros FIR por janelamento aos autores, auxiliando a compreensão do que foi lecionado em sala de aula. O código implementado fez com que os autores ainda aprendessem a utilizar diversas funções do MATLAB, além de aprender e conhecer diversos algorítimos de otimização de filtros. Com relação a parte teórica foi possível verificar como trabalhar com as especificações para fazer um janelamento corretamente e escolher a ordem e frequência de corte mais adequada ao projeto para minimizar custo e respeitar as especificações. Além disso, foi feita a analise da não linearidade prática de filtros implementáveis na prática, pois estes devem ser quantizados com precisão finita, que depende da memória do processador dos sinais. Foram conhecidas formas de analisar e corrigir o filtro quando não era mais possível utilizar as teorias de sistemas lineares, devido a não linearidade da quantização. Foi interessante perceber que, em alguns casos, as não linearidades introduzidas no filtro podem fazer com que ele não seja implementável, como foi apresentado nas tentativas de implementar os filtros com precisão de 8 bits.

Os resultados obtidos foram bastante satisfatórios, quando respeitavam as especificações, como foi apresentado na seção anterior. A análise das distorções harmônicas apresentou um resultado surpreendente, fazendo com que os autores inclusive revisassem a implementação de algumas funções para validar o resultado obtido, porém, como não foram encontrados erros claros, foi concluído que, a princípio, os resultados foram condizentes, pois o filtro estava introduzindo uma distorção mínima, o que é muito bom.

Os autores ainda desenvolveram um algorítimo para convolução, apresentada na função *MAC.m*, fazendo as operações de multiplicação e acumulação dos sinais por matrizes, de uma forma bastante rápida e otimizada, fazendo com que evoluíssem bastante seu raciocínio lógico para programação matricial.

Para futuros trabalhos ainda podem ser testadas outras janelas, como a Barlett ou a Hann, e avaliar seus resultados comparando para quais aplicações cada uma se comporta melhor, levantando os trade-offs de cada uma. Além disso os algorítimos de otimização dos filtros podem ser otimizados, criando formas de aumentar a velocidade do projeto do filtro, mantendo sempre o respeito as especificações. Para isso deve-se buscar outros tipos de algorítimos mais sofisticados de verificação e correção, visto que o implementado nesse projeto simplesmente testa se o filtro respeita as especificações até encontrar uma ordem e uma frequência de corte que respeite as especificações, dentro de um laço que pode ser bastante demorado se a ordem for muito grande.

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 *FIR_Filter.m*. Também foram setados alguns parâmetros para o funcionamento do projeto que podem ser alterados nesse arquivo, como o número de senoides testadas para corrigir o filtro não linear, quantidade de amostras das transformadas de Fourier *N_fft* assim como resoluções de divisões de frequência utilizadas em alguns testes, setados com valores da decisão dos autores. Também foi implementado um limite de incremento de ordem, para que o programa não fique muito tempo sem conseguir implementar o filtro, logo, após 200 incrementos na ordem esperada (para operação linear) e/ou 15 incrementos na ordem esperada (para operação não linear) para cada janela, o programa é interrompido e as respostas plotadas são respostas de um filtro incoerente com as especificações, e a interface avisa que o filtro não foi implementado corretamente.

Por fim é válido lembrar que antes da execução do projeto, o usuário deve selecionar uma das janelas 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

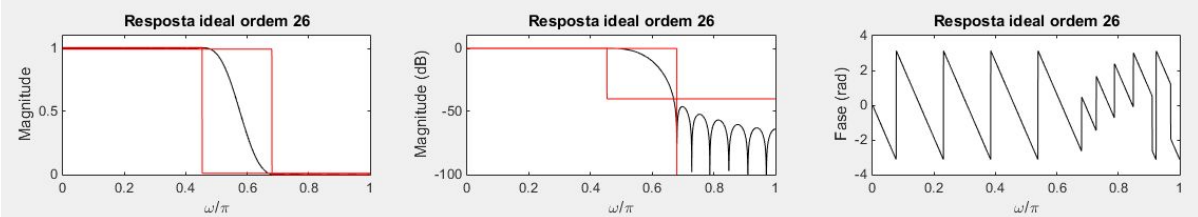

Imagem 6.1 - Janela Kaiser com precisão infinita

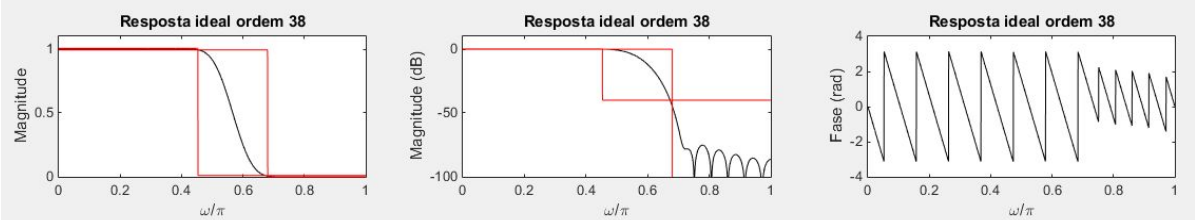

Imagem 6.2 - Janela Blackman com precisão infinita

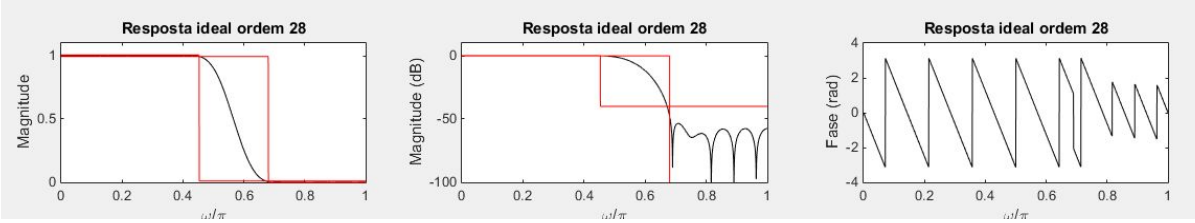

Imagem 6.3 - Janela Hamming com precisão infinita

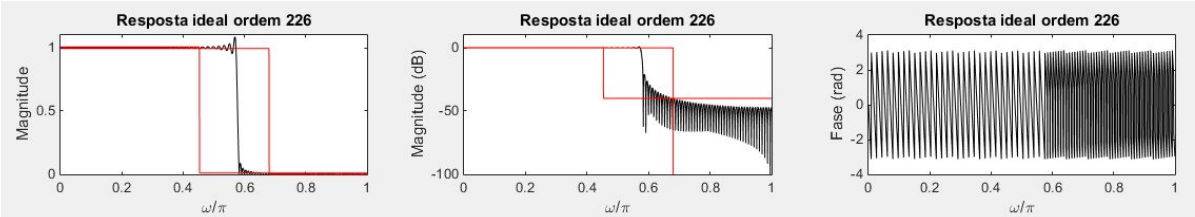

Imagem 6.4 - Janela Retangular com precisão infinita

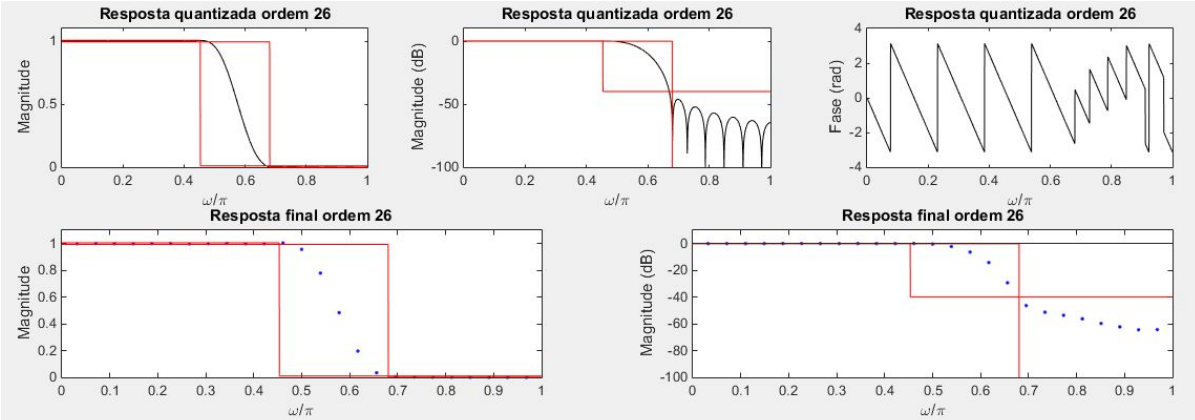

Imagem 6.5 - Janela Kaiser 16 bits

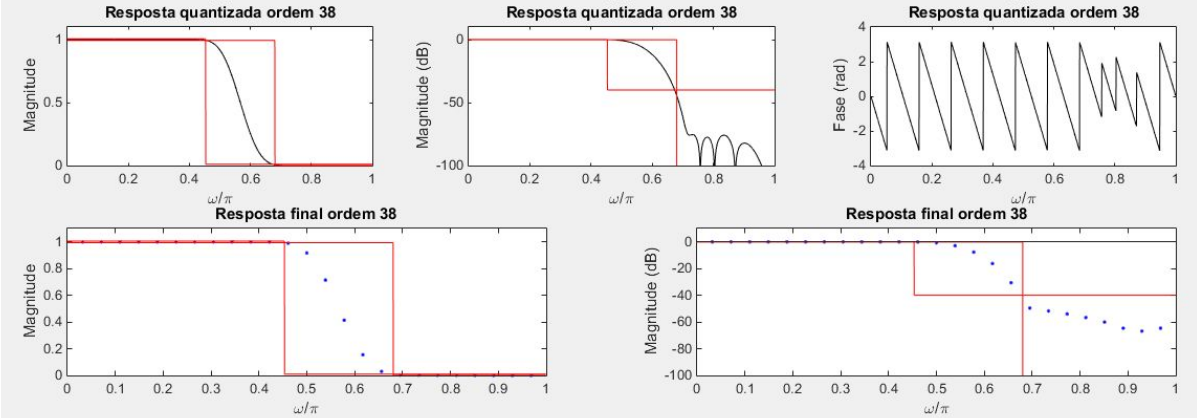

Imagem 6.6 - Janela Blackman 16 bits

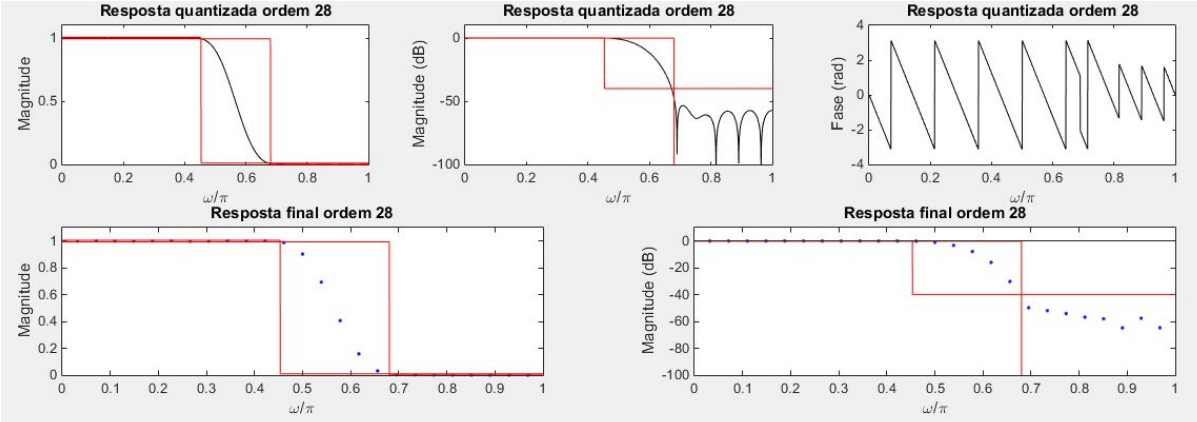

Imagem 6.7 - Janela Hamming 16 bits

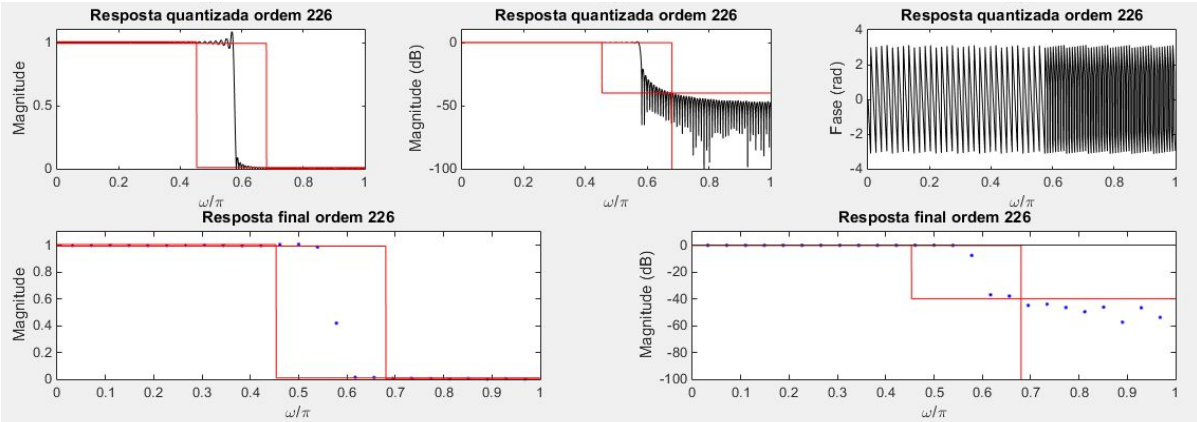

Imagem 6.8 - Janela Retangular 16 bits

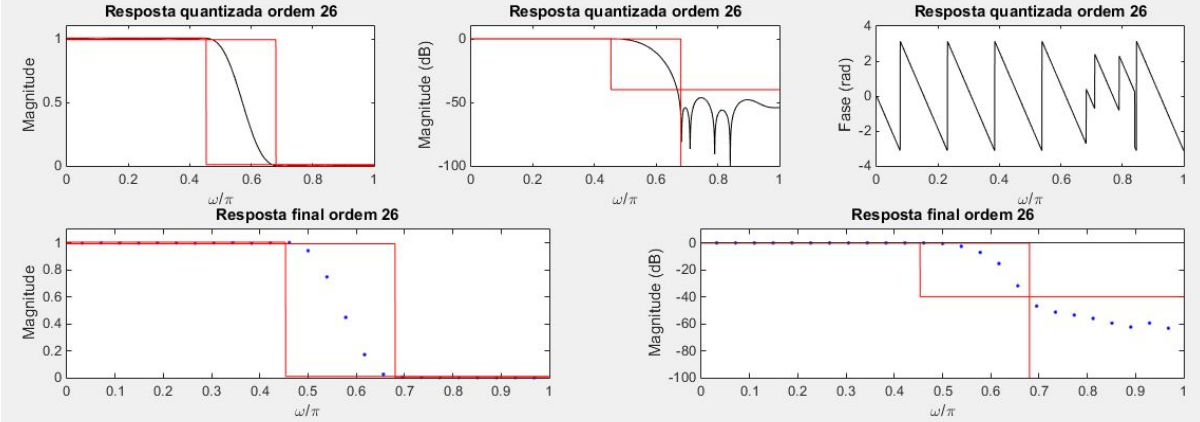

Imagem 6.9 - Janela Kaiser 10 bits

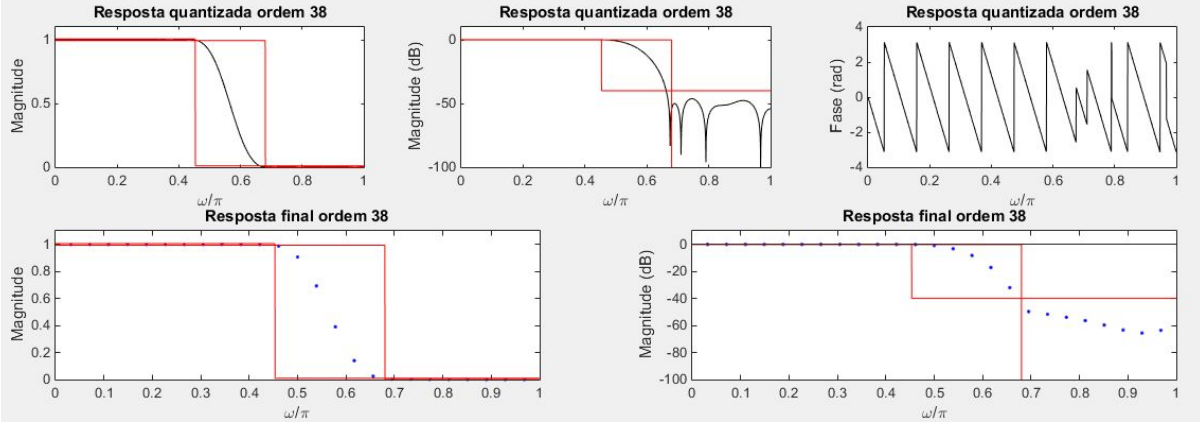

Imagem 6.10 - Janela Blackman 10 bits

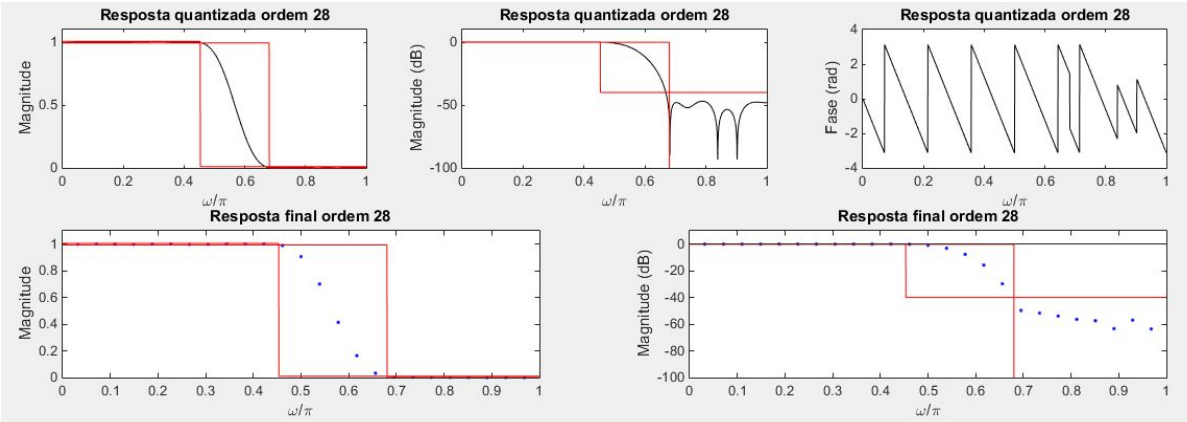

Imagem 6.11 - Janela Hamming 10 bits

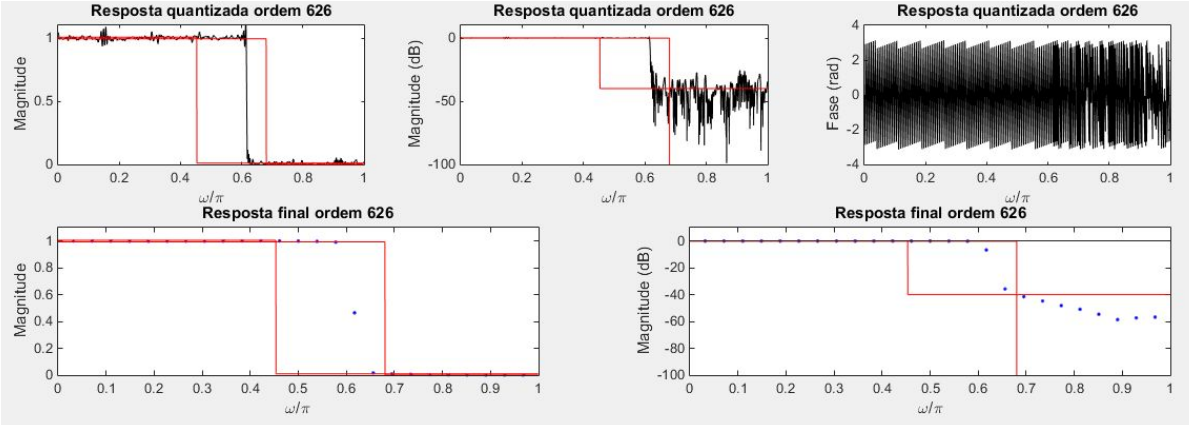

Imagem 6.12 - Janela Retangular 10 bits (não implementável)

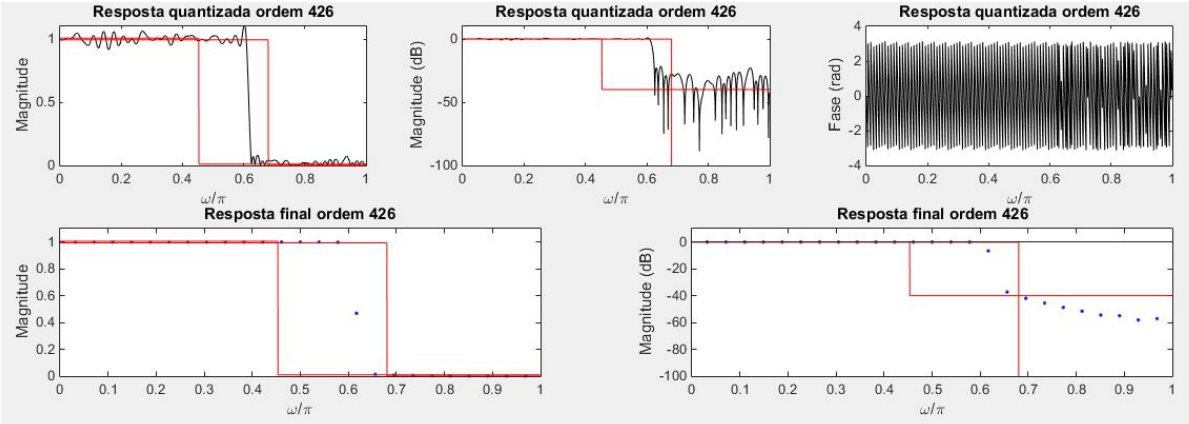

Imagem 6.13 - Janela Kaiser 8 bits (não implementável)

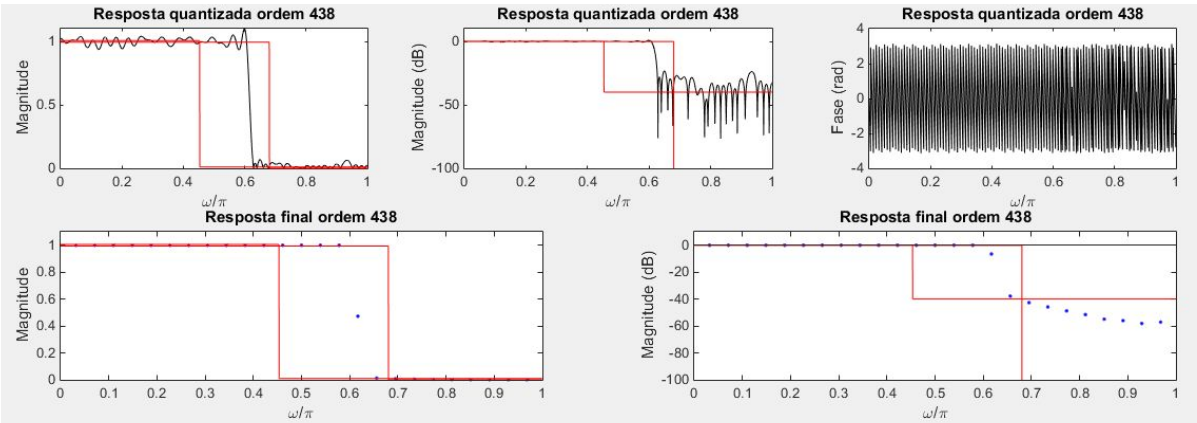

Imagem 6.14 - Janela Blackman 8 bits (não implementável)

rapidamente em seu documento.

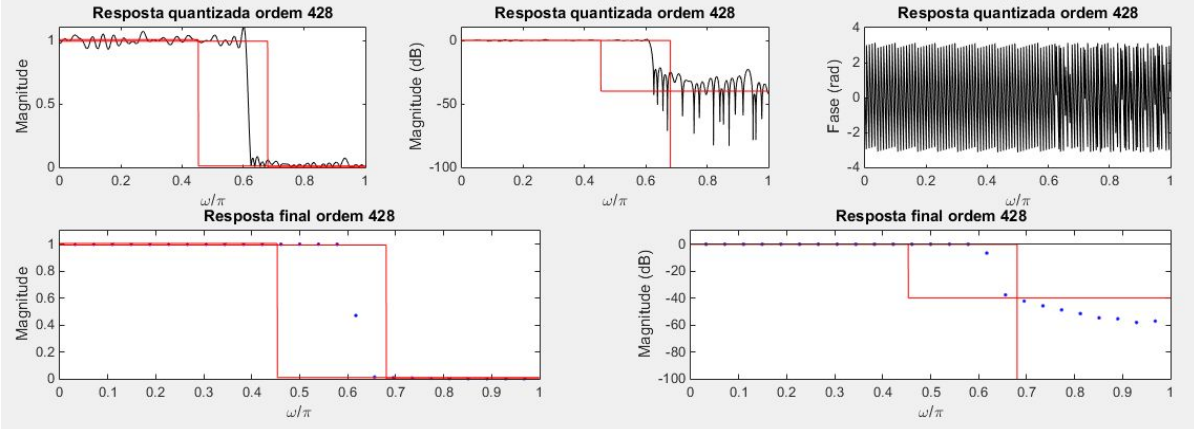

Imagem 6.15 - Janela Hamming 8 bits (não implementável)

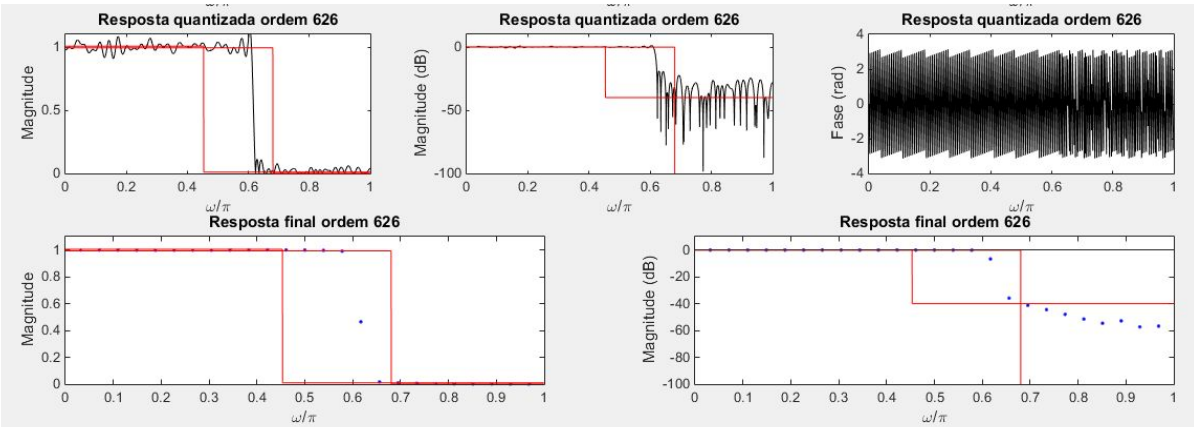

Imagem 6.16 - Janela Retangular 8 bits (não implementável)

7 - Correções

Após as correções feitas pelo monitor e pelo professor, fomos aconselhados a alterar um ponto do projeto, para obter um resultado melhor. A correção a ser feita era eliminar as amostras iniciais da saída do teste do filtro quantizado, de forma a eliminar a resposta transitória do filtro. Para isso, todos os arquivos que utilizam a função MAC.m, que calcula a saída do filtro quantizado, foram alteradas. no arquivo MAC.m, adicionamos uma linha, número 65, onde as primeiras 200 amostras da saída quantizada para memória do filtro são jogadas fora. No arquivo filter_improvment_quant.m a linha 96 foi alterada, pois devemos retirar o mesmo número de amostras da entrada antes de calcular a transformada de fourrier para encontrar os pontos de ganho máximo de cada frequência testada. Por fim, no arquivo filter_plots.m, linha 45, fizemos a mesma alteração para plotar o ganho real de cada frequência testada com as senóides.

Com essas alterações foi possível observar uma melhora na qualidade do sinal de saída, visto que o thd para os sinais de saída teve uma leve melhora. Já os gráficos de magnitude e ganho ficaram bastante similares aos apresentados anteriormente.

8 - Referências

[1] - Capítulo 5 Seção 4, 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] - PDF Requerimentos do Projeto, José Carlos Bermudez 2016.

[4] - Introdução do capítulo 15, D. G. Manolakis and V. K. Ingle, Applied Digital Signal Processing. Cambridge University Press, 2011.