Aluno: Ana Luiza Morais Oliveira Matrícula: 14.1.8312
Relatório: Transformada Rápida de Fourier

1 Introdução

A transformada de Fourier discreta (TFD) desempenha um papel importante na análise, projeto e implementação de algoritmos e sistemas de processamento em tempo discreto de sinais. A TFD é um componente importante em muitas aplicações práticas dos sistemas de tempo discreto.

Existem vários métodos para o cálculo de valores da TFD. O foco principal é analisar uma classe de algoritmos particularmente eficiente para a implementação computacional da TFD de N pontos. Esses algoritmos eficientes são conhecidos como *Fast Fourier Transform* (FFT). Para alcançar a máxima eficiência, os algoritmos de FFT precisam calcular todos os N valores da TFD. Há muitas maneiras de medir o custo e a eficiência de uma implementação, e uma avaliação final depende tanto da tecnologia de implementação disponível quanto da aplicação pretendida. Será usado o número de multiplicações e adições aritméticas como uma medida do custo computacional e essa medida é simples de aplicar.

Neste trabalho é implementado os algoritmos que representam a TFD (cálculo direto e dizimação no tempo), sendo posteriormente comparado o tempo de execução e erro entre as funções, afim de mostrar a eficiência de tais implementações.

1.1 Objetivos

Os objetivos deste trabalho são:

- Implementar o algoritmo de TFD.
- Implementar o algoritmo de dizimação no tempo.
- Comparar as funções desenvolvidas com as nativas do MatLab.

2 A Transformada de Fourier Discreta - TFD

A TFD de uma sequência de comprimento finito de comprimento N é:

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn} \tag{1}$$

para $k = 0, ..N-1$, onde também $W_N^{kn} = e^{-j(2\pi/N)}$.

A transformada de Fourier discreta inversa é dada por:

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] W_N^{-kn} \tag{2}$$

A maioria das técnicas para melhorar a eficiência do cálculo da TFD explora as propriedades de simetria e periodicidade de W_N^{kn}. Especificamente:

- Simetria complexa conjugada:

$$W_N^{k(N-n)} = W_N^{-kn} = (W_N^{kn})^* \tag{3}$$

- Periodicidade:

$$W_N^{kn} = W_N^{k(N+n)} = W_N^{n(k+N)} \tag{4}$$

Como $W_N^{kn} = \cos(2\pi kn/N) - j\sin(2\pi kn/N)$, essas propriedades são uma consequência direta da simetria e da periodicidade das funções seno e cosseno subjacentes. Como os números complexos fazem o papel de coeficientes nas equações (1) e (2), uma redundância consequente dessas condições pode ser usada na redução dos cálculos.

2.1 Cálculo direto pela definição de DFT

Usando a equação (1) como referência, temos que x[n] pode ser complexo, então N multiplicações complexas e $(N1)$ adições complexas são requeridas para calcular cada valor da TFD. **Para calcular todos os N valores, portanto, é preciso um total de N^2 multiplicações complexas e $N(N$-1) adições complexas.**

Para melhor entendimento, vamos expandir em termos de operações sobre números reais:

$$X[k] = \sum_{n=0}^{N-1} [(Re(x[n])Re(W_N^{kn}) - Im(x[n])Im(W_N^{kn})) + jRe(x[n])Im(W_N^{kn}) + Im(x[n])Re(W_N^{kn})]] \tag{5}$$

Isso mostra que cada multiplicação completa $x[n] * W_N^{kn}$ requer quatro multiplicações reais e duas adições reais, e cada adição complexa requer duas adições reais. Portanto, para cada valor de k, o cálculo direto de X[k] requer $4N$ multiplicações reais e $(4N\ -2)$ adições reais. Como X[k] precisa ser calculado para N diferentes valores de k, **o cálculo direto da TFD de uma sequência x[n] requer $4N^2$ multiplicações reais e $N(4N$ - 2) adições reais**.

Como a quantidade de cálculos (e, portanto, o tempo de computação) é aproximadamente proporcional a N^2, é evidente que o número de operações aritméticas requeridas para calcular a TFD pelo método direto se torna muito grande para valores grandes de N. Por esse motivo, estaremos interessados em procedimentos de cálculo que reduzem o número de multiplicações e adições.

Usando as propriedades é possível diminuir o custo de operação da TFD. Pela propriedade de simetria (equação 3) é possível agrupar as parcelas do somatório da equação (5) e assim, diminuir o custo de operação. A segunda propriedade, a de periodicidade (equação 4) pode ser explorada por recursão (por exemplo, algoritmo de Goertzel) e assim, diminuir ainda mais a complexidade do cálculo de DFT.

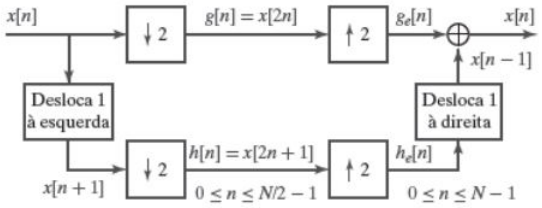
2.2 Algoritmos de FFT com dizimação no tempo

A decomposição e o cálculo da TFD em TFDs sucessivamente menores, explorando-se tanto a simetria quanto a periodicidade da exponencial complexa resulta em um aumento drástico na eficiência do cálculo da TFD. Os algoritmos em que a decomposição é baseada em decompor a sequência x[n] em sub-sequências sucessivamente menores são chamados de algoritmos de dizimação no tempo.

O princípio da dizimação no tempo é ilustrado, na Figura 1, de modo conveniente considerando-se o caso especial de N sendo uma potência inteira de 2, isto é, $N = 2^v$. Como N é divisível por dois, pode-se considerar a realização do cálculo de X[k] separando x[n] em duas sequências de $N/2$ pontos que consistam em pontos de índice par $g[n] = x[2n]$ e pontos de índice ímpar $h[n] = x[2n + 1]$.

Na Figura 1 são mostrados essa decomposição e também o fato de que a sequência original pode ser recuperada simplesmente pelo entrelaçamento das duas sequências.

Figura 1: Exemplo do princípio básico da dizimação no tempo



Fonte: [1]

Assim, o algoritmo de dizimação no tempo divide uma sequência de comprimento N, em dois, sendo um par e outro ímpar. A equação (6) evidencia esse processo.

$$X[k] = \sum_{n=0}^{(N/2)-1} x[2n]W_{N/2}^{nk} + W_N^k \sum_{n=0}^{(N/2)-1} x[2n+1]W_{N/2}^{nk} \tag{6}$$

Portanto as etapas para a dizimação no tempo são:

- Dividir sucessivamente a sequência x[n] até N sequências de 1 amostra.
- Depois, combinam-se sucessivamente as DFT's geradas anteriormente até obter X[k].

Um subproduto interessante dessa dedução é que esse diagrama de fluxo, além de descrever um procedimento eficiente para o cálculo da TFD, também sugere um modo útil de armazenar os dados originais e os resultados do cálculo em matrizes intermediárias.

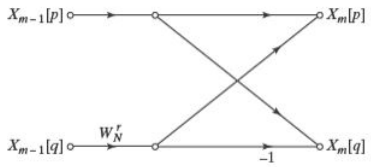
3 Considerações Práticas

Embora devemos levar em consideração os diagramas de fluxo, diversos detalhes precisam ser considerados na implementação de um dado algoritmo. Nesta seção serão descritos resumidamente alguns deles.

3.1 Indexação

No algoritmo representado na Figura 2 a entrada precisa estar em ordem bit-reversa, de modo que o cálculo possa ser realizado localmente. A TFD resultante está, então, em ordem normal. Geralmente, as sequências não se originam em ordem bit-reversa, de modo que o primeiro passo na implementação da Figura 2 é ordenar a sequência de entrada na ordem bit-reversa.

Figura 2: Exemplo de diagrama de fluxo



Fonte: [1]

A ordenação bit-reversa pode ser realizada localmente, já que as amostras são trocadas apenas em pares; isto é, uma amostra em um dado índice é trocada com a amostra na localização especificada pelo índice de bit reverso. Isso é convenientemente realizado localmente usando dois contadores, um na ordem normal e o outro na ordem bit-reversa. Os dados nas duas posições especificadas pelos dois contadores são simplesmente trocados. Uma vez que a entrada esteja na ordem bit-reversa, podemos prosseguir com o primeiro estágio do cálculo.

A escolha de determinado algoritmo depende de diversos fatores. Os algoritmos que utilizam um cálculo realizado localmente têm a vantagem de fazer uso eficiente da memória. Porém, duas desvantagens são que o tipo de memória exigido é de acesso aleatório em vez de sequencial e que a sequência de entrada ou a sequência da TFD de saída está na ordem bit-reversa. Além disso, dependendo se um algoritmo de dizimação no tempo ou de dizimação na frequência é escolhido, e se as entradas ou as saídas estão na ordem bit-reversa, requer-se que os coeficientes sejam acessados ou na ordem normal ou na ordem bit-reversa. Se é usada memória sequencial sem acesso aleatório, alguns algoritmos da transformada de Fourier rápida utilizam memória sequencial mas ou as entradas ou as saídas deverão estar na ordem bit-reversa. Embora o diagrama de fluxo para o algoritmo possa ser organizado para que as entradas, as saídas e os coeficientes estejam na ordem normal, a estrutura de indexação exigida para implementar esses algoritmos é complicada, e o dobro de memória de acesso aleatório é necessário. Consequentemente, o uso desses algoritmos parece não ser vantajoso.

Os algoritmos de FFT com cálculos realizados localmente estão entre os mais comumente utilizados. Se uma sequência tiver de ser transformada apenas uma vez, então a ordenação bit-reversa precisa ser implementada ou na entrada ou na saída. Porém, em algumas situações, uma sequência é transformada, o resultado é modificado de alguma forma e depois a TFD é calculada. Quando duas transformadas são colocadas em cascata, é possível, pela escolha apropriada dos algoritmos de FFT, evitar a necessidade de reversão de bits.

Se a forma do algoritmo por dizimação no tempo for escolhida para a transformada direta, então a forma do algoritmo por dizimação na frequência deverá ser escolhida para a transformada inversa, exigindo coeficientes na ordem bit-reversa. Da mesma forma, o algoritmo por dizimação na frequência para a transformada deverá ser emparelhado com o algoritmo por dizimação no tempo para a transformada inversa, que então utilizará coeficientes ordenados normalmente.

3.2 Coeficientes

Os coeficientes W_N^r podem ser requeridos na ordem bit-reversa ou na ordem normal, necessitando em ambos os casos o armazenamento de uma tabela suficiente para pesquisar todos os valores

requeridos ou então calcular os valores quando necessários.

A primeira alternativa (armazenamento em tabela) tem a vantagem da velocidade, porém exige mais armazenamento. O cálculo dos coeficientes à medida que são necessários economiza armazenamento, mas é menos eficiente do que armazenar uma tabela de pesquisa. Se os coeficientes têm de ser calculados, geralmente é mais eficiente usar uma fórmula de recursão.

4 Pseudocódigos

Nesta sessão será apresentadas as implementações do algoritmo da DFT pelo Cálculo Direto e pela Dizimação no Tempo. Serão apresentados também todos os pseudocódigos e o detalhamento de todas as funções realizadas e comparadas à função nativa do MatLab.

4.1 Algoritmo da Transformada Discreta de Fourier - MyDFT

A primeira implementação foi feita utilizando a fórmula direta da TDF:

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn} \tag{7}$$

O algoritmo implementado está no anexo mas, abaixo segue o pseudocódigo do **algoritmo MyDFT**.

Algorithm 1 DFT

```
1: function MyDFT(x)
2:     for k do1N
3:         soma ← 0
4:         for n do1N
5:             soma ← x(n)exp^{-j(2π/N)(k-1)(n-1)} + soma
6:         end for
7:         dft(k) ← soma
8:     end for
9: end function
```

Analisando o pseudocódigo nota-se que são realizadas N multiplicações complexas e $N-1$ adições complexas. Para calcular todos os N valores, é preciso de N^2 multiplicações complexas e $N(N-1)$ adições complexas.

Portanto, além do tempo necessário para implementação destas multiplicações e adições, ainda é necessário um tempo específico para requerer o armazenamento e o acesso a esses N valores. Espera-se portanto, comparado às outras implementações, que a *MyDFT* seja o algoritmo que demanda um maior tempo de implementação.

4.2 Algoritmo de FFT com dizimação no tempo - MyFFTDecTempo

O algoritmo sorteado para implementar a FFT foi o da dizimação no tempo. O pseudocódigo é apresentado abaixo:

Algorithm 2 Dizimação no tempo

```
function MyFFTDecTempo(x)
    tam1 ← size(x)
    if tam1(1,1) > tam1(1,2) then
        x ← x'
    end if
    N ← length(x)
    vet ← log2(N)
    vet ← ceil(vet)
    aux ← 2^vet
    if N ≠ aux then
        aux = aux − N
        x = [xzeros(1, aux)]
    end if
    n ← length(x)
    Wn ← e^(2π/n)
    w ← 1
    if n == 1 then
        FFTx ← x
    else
        xlpar ← zeros(1, length(x)/2)
        xlimpar ← zeros(1, length(x)/2)
        cont1 ← 1
        cont2 ← 1
        for a do1length(x)
            if mod(a, 2) == 0 then
                xlpar(1, cont1) ← x(1, a)
                cont1 ← cont1 + 1
            else
                xlimpar(1, cont1) ← x(1, a)
                cont2 ← cont2 + 1
            end if
        end for
        Ypar ← MyFFTDecTempo(xlpar)
        Yimpar ← MyFFTDecTempo(xlimpar)
        for k do1((n/2) − 1)
            FFTx(1, (k + 1)) ← Ypar(1, (k + 1)) + w ∗ Yimpar(1, k + 1)
            FFTx(1, (k + 1)) ← Ypar(1, (k + 1)) − w ∗ Yimpar(1, k + 1)
            w ← w ∗ Wn
        end for
```

Vimos que para a implementação da *MyDFT* anteriormente, requeria-se N^2 multiplicações e adições complexas. O cálculo para dizimação no tempo para todos os valores de k exige no máximo $N + 2(N/2)^2$ multiplicações e adições complexas. Então com isso, é possível observar que a função implementada de dizimação no tempo gasta menos cálculos que a função de DFT implementada.

5 Resultados e Discussões

Nessa seção trataremos dos resultados obtidos com as funções obtidas.

5.1 Teste

A atividade solicitada era criar um sinal aleatório e soma-ló com dois senos com frequência de 10 Hz e 25Hz. A Figura 3 mostra esse desenvolvimento solicitado.

Figura 3: Sinal criado para teste

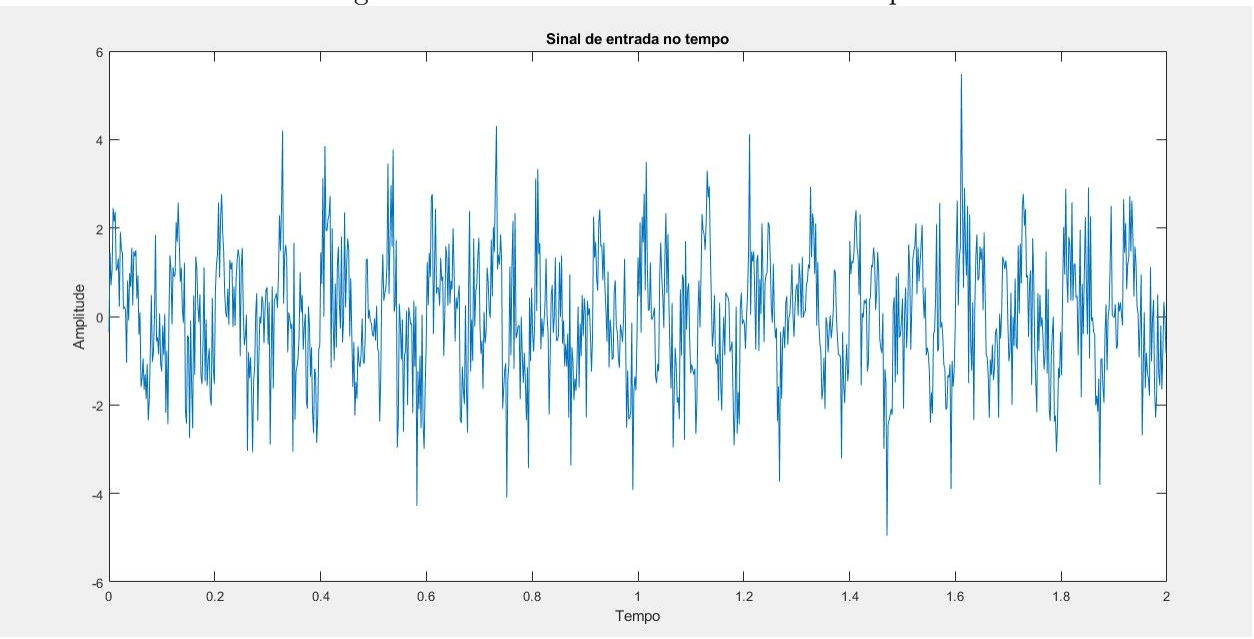
```
1
2       % teste: sinal aleatório + sen(2*pi*10*t) + sen(2*pi*25*t)
3       % com fs=512 e N=2,4,8,16,32,64,128,256,512,1024 e 2048)
4 -     clc
5 -     clear all
6 -     close all
7
8 -      N=1024; %AQUI VARIA-SE O N DESEJADO
9 -      fs =512;
10 -     T = 0: 1 /fs :((N/fs)-(1/fs ));
11 -     ws = 0:fs/N:fs-(fs/N);
12 -     x = randn (1,N);
13 -     x1 = sin(2*pi*10*T);
14 -     x2 = sin(2*pi*25*T);
15 -     X = x + x1 +x2;
```

Fonte: do autor

E a Figura 4 mostra esse sinal no domínio no tempo. O outro ponto de analise é observar os espectros de magnitude e fase na frequência encontrados pela função nativa do MatLab, da *MyDFT* e *MyFFTDecTempo* e compará-las. E também foi solicitado a variação da janela N de 2 até 2048.

Para simplificar a análise será apresentado os espectros de magnitude e fase das funções em um só gráfico para a variação de N.

Figura 4: Sinal aleatório no domínio do tempo



Fonte: do autor

5.1.1 Análise par N=2

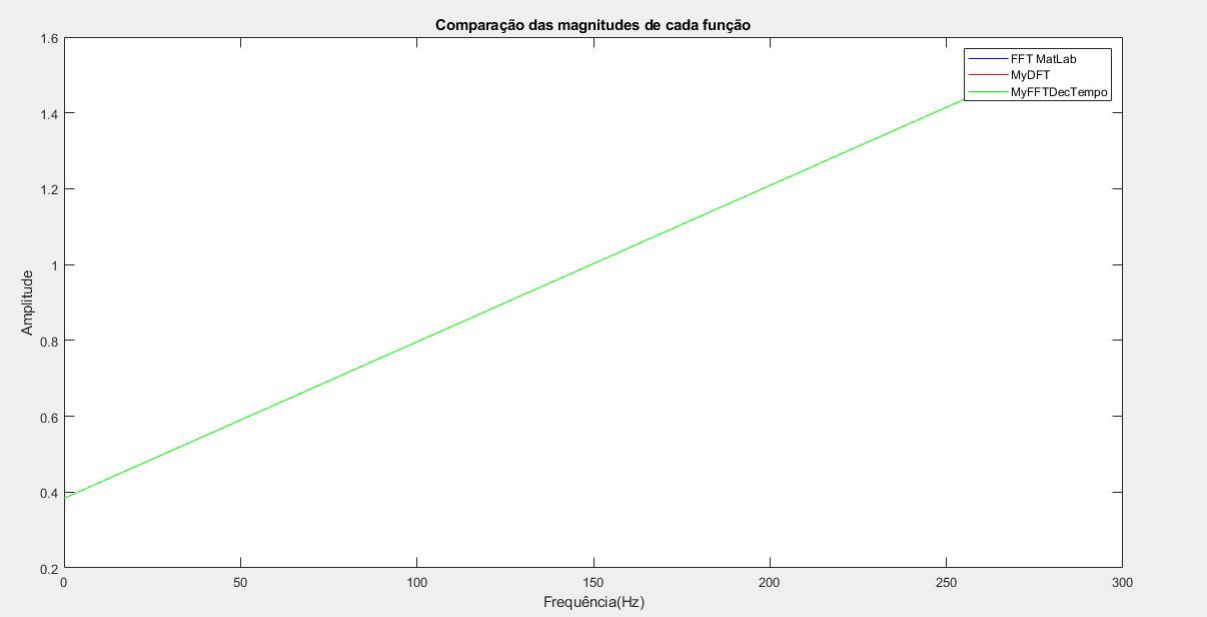
Aqui fizemos N=2 e copilamos o algoritmo para gerar os gráficos de magnitude e fase e pelas Figuras 6 e 7 podemos observar esses espectros. Como podemos observar a Figura 5 corrobora com o gráfico do espectro de magnitude, ou seja, para N=2 não houve diferenças entre as função implementadas (FFT, MyDFT e MyFFTDecTempo). Para o espectro de fases observa-se que há uma diferença entre as funções implementadas, tanto que o erro entre a FFT e as outras funções implementadas foi de 4.9348 s.

Figura 5: Erro entre os espectros para N=2

erro1F	9.8608e-32
erro1M	0
erro2F	4.9348
erro2M	0
erro3F	4.9348
erro3M	0

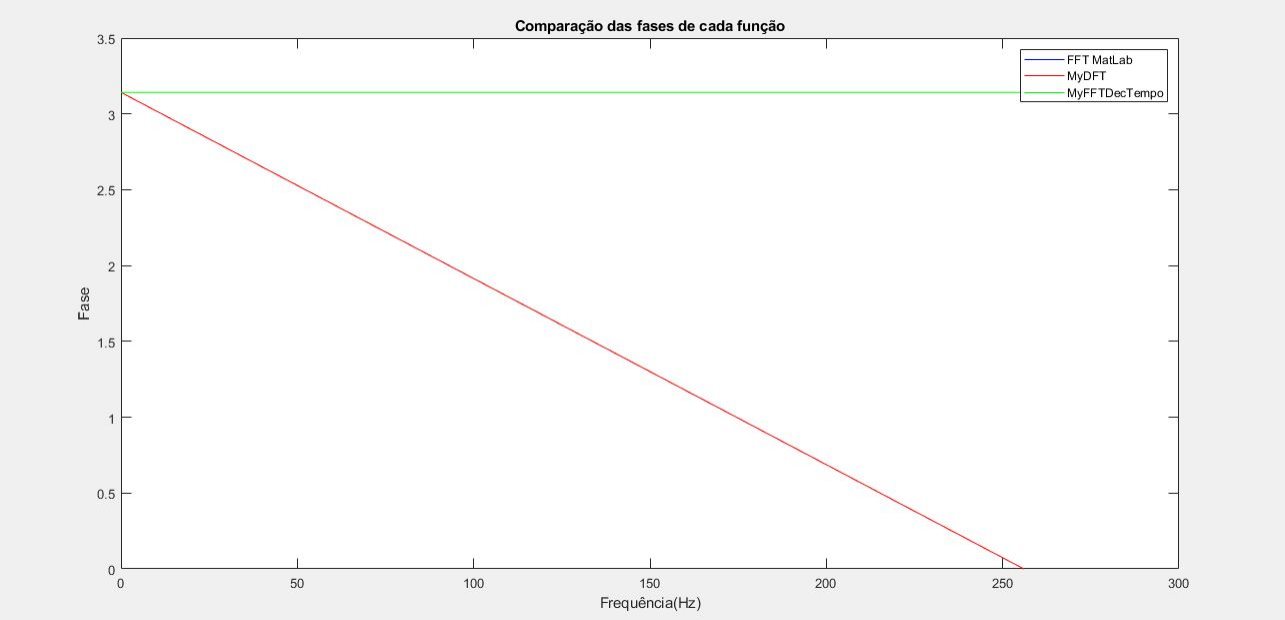
Fonte: do autor

Figura 6: Espectro de magnitude para N=2



Fonte: do autor

Figura 7: Espectro de fase para N=2



Fonte: do autor

5.1.2 Análise para N=4

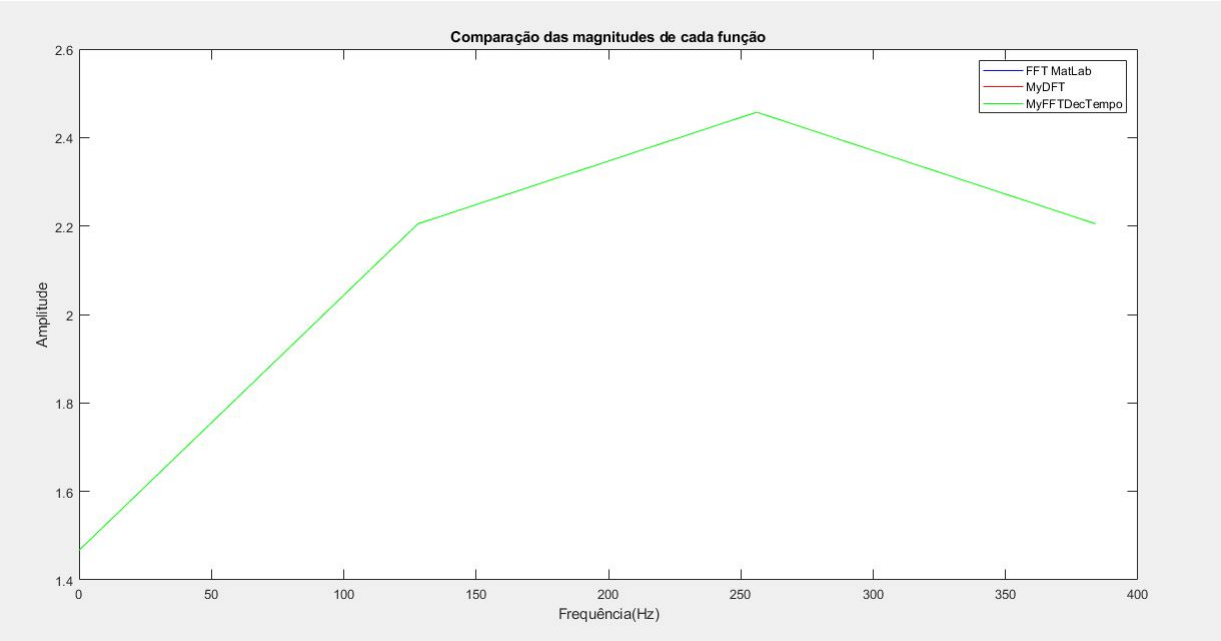
As Figuras 9 e 10 mostram os espectro de magnitude e fase para N=4, respectivamente. Como podemos observar, na Figura 9, os espectros de magnitudes das funções (FFT, MyDFT e MyFFT-DecTempo) se sobrepõem e ambas possuem o mesmo erro, que é de 4.9304^{-32}. Já, na Figura 10, vemos que há sobreposição das fases da FFT e da MyDFT, o que mostra um erro 0 entre essas duas funções. E na fase da função MyFFTDecTempo há um pequeno erro de 8.6359, em comparação com a função nativa do matlab.

Figura 8: Erro entre os espectros para N=4

erro1F	4.9304e-32
erro1M	4.9304e-32
erro2F	8.6359
erro2M	4.9304e-32
erro3F	8.6359
erro3M	0

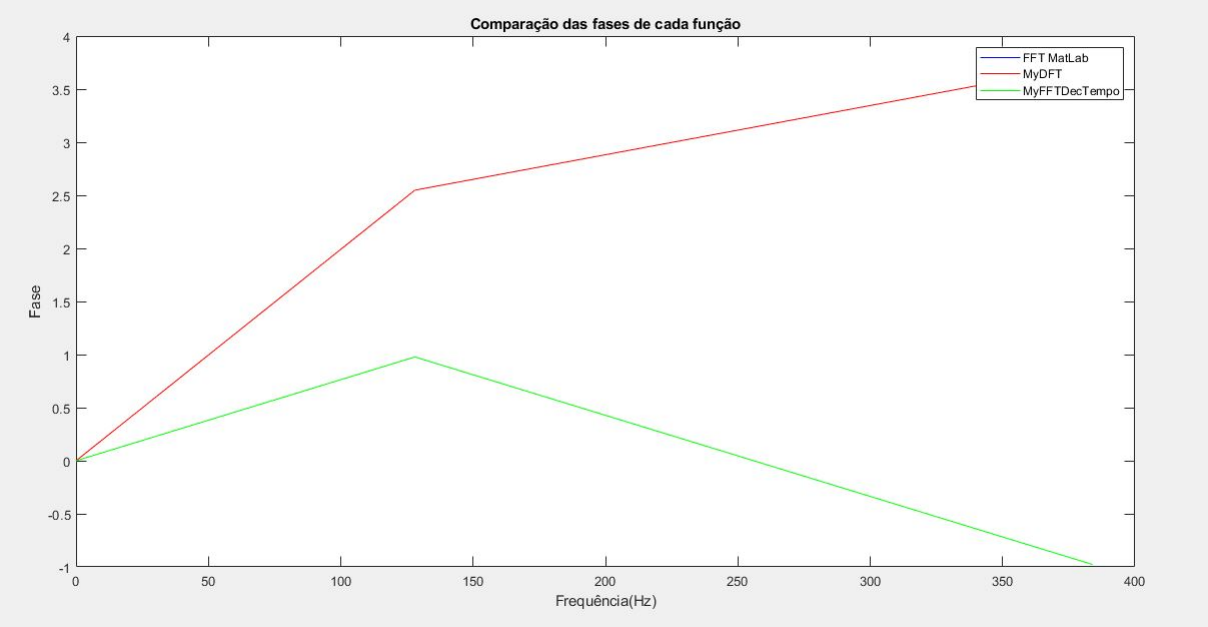
Fonte: do autor

Figura 9: Espectro de magnitude para N=4



Fonte: do autor

Figura 10: Espectro de fase para N=4

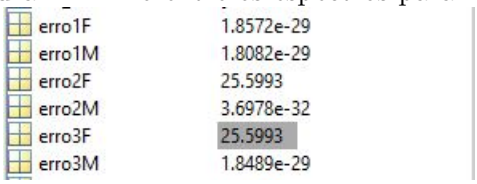


Fonte: do autor

5.1.3 Análise para N=8

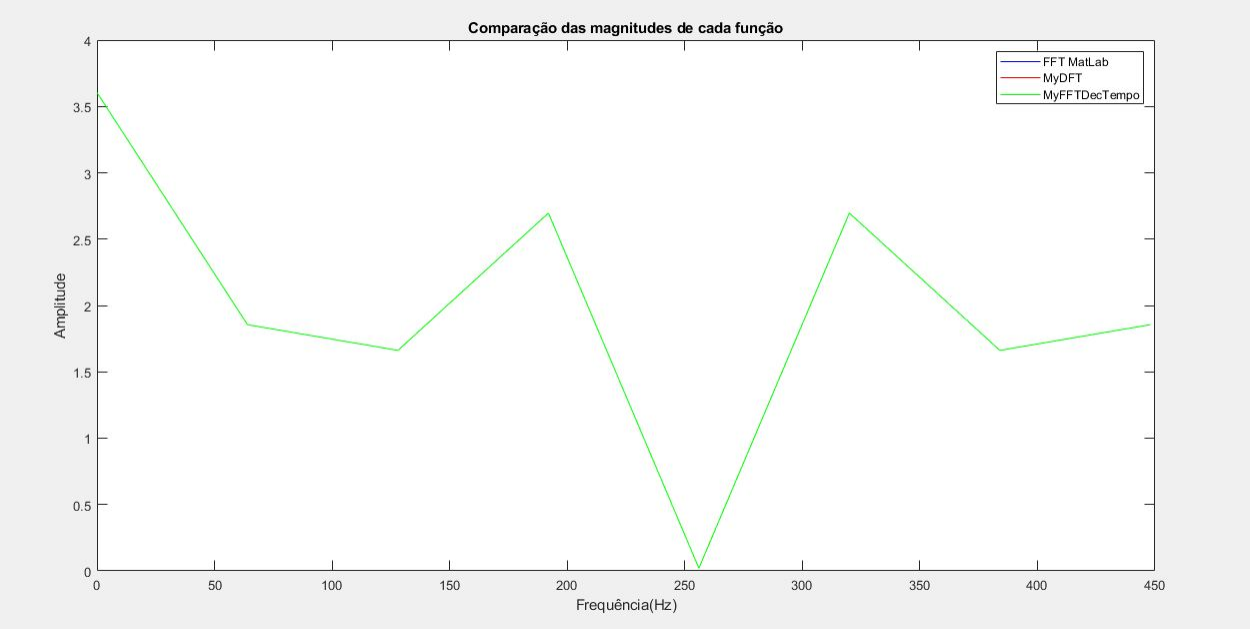
Os espectros de magnitude e fase podem ser observados nas Figuras 12 e 13. Novamente, vemos que os espectros de magnitude não há diferença, pois os espectros estão sobrepostos. Já para o espectro de fase vemos que há um erro entre a FFT e MyFFTDecTempo e o erro é de 10.7949.

Figura 11: Erro entre os espectros para N=8



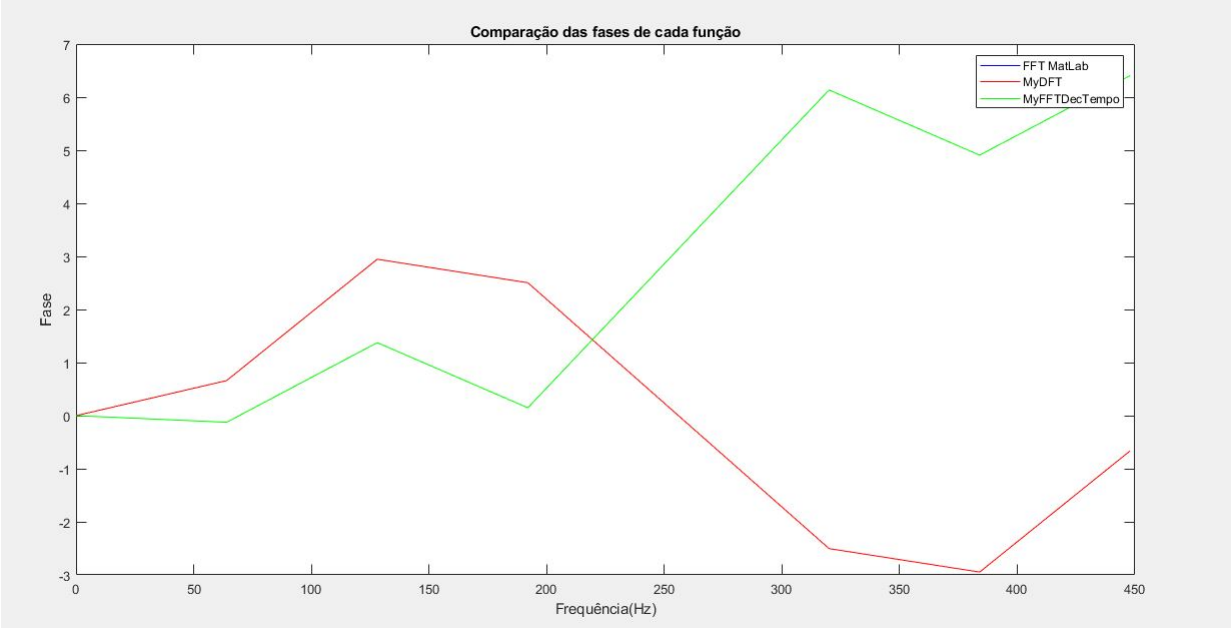
Fonte: do autor

Figura 12: Espectro de magnitude para N=8



Fonte: do autor

Figura 13: Espectro de fase para N=8



Fonte: do autor

5.1.4 Análise para N=16

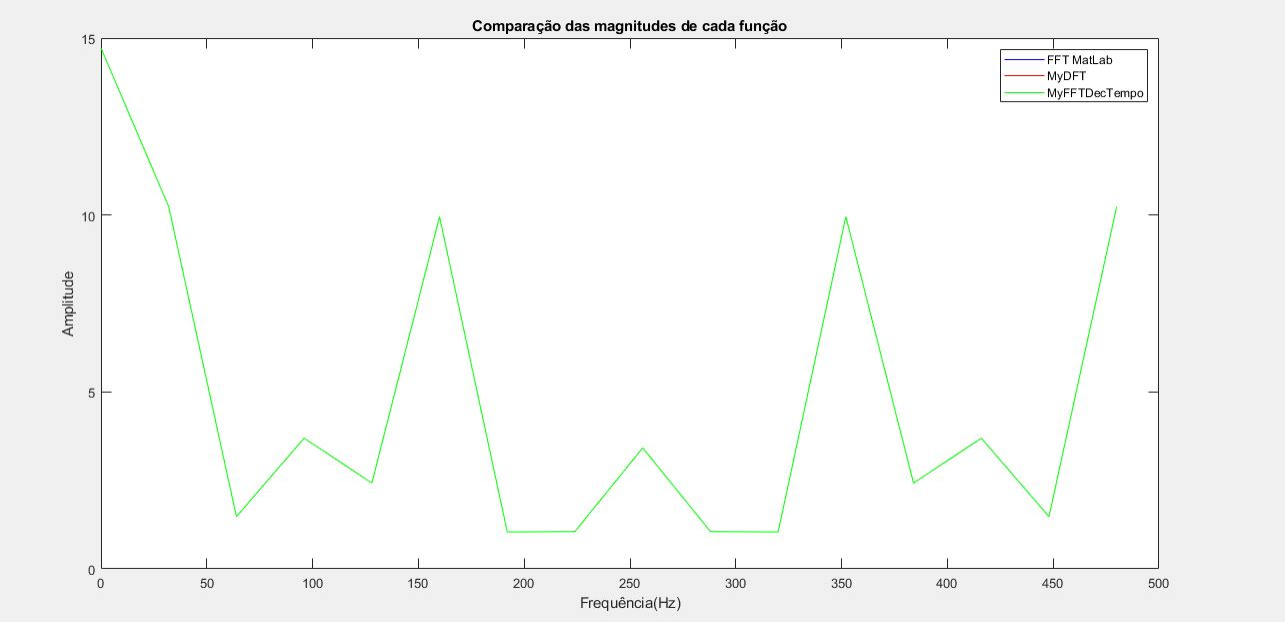
As Figuras 15 e 16 mostram os espectros de magnitude e fase. Como podemos observar, na Figura 15, as magnitudes das funções (FFT, MyDFT e MyFFTDecTempo) estão sobrepostas, indicando um erro muito pequeno que pode ser observado na Figura 14. Epor fim, vemos também que no gráfico de espectro de fases, há novamente em erro considerável entre a FFT e a MyFFTDecTempo.

Figura 14: Erro entre os espectros para N=16

erro1F	1.6182e-29
erro1M	6.3093e-29
erro2F	11.9515
erro2M	2.6501e-31
erro3F	11.9515
erro3M	6.2385e-29

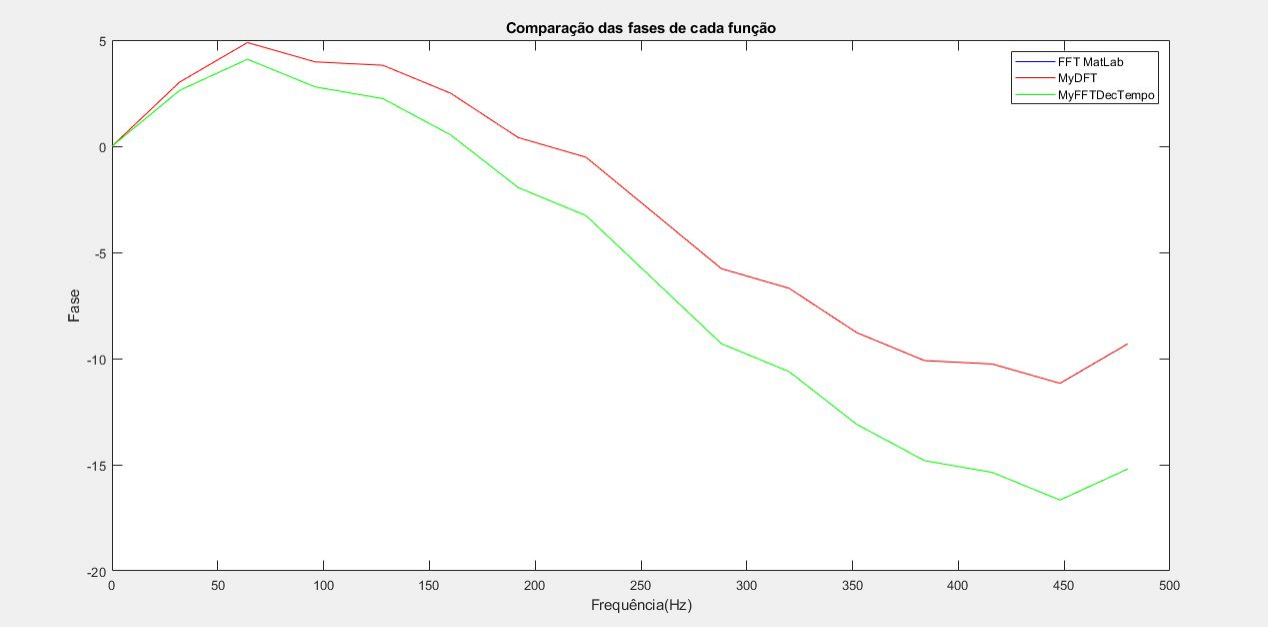
Fonte: do autor

Figura 15: Espectro de magnitude para N=16



Fonte: do autor

Figura 16: Espectro de fase para N=16

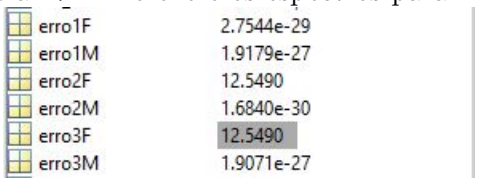


Fonte: do autor

5.1.5 Análise para N=32

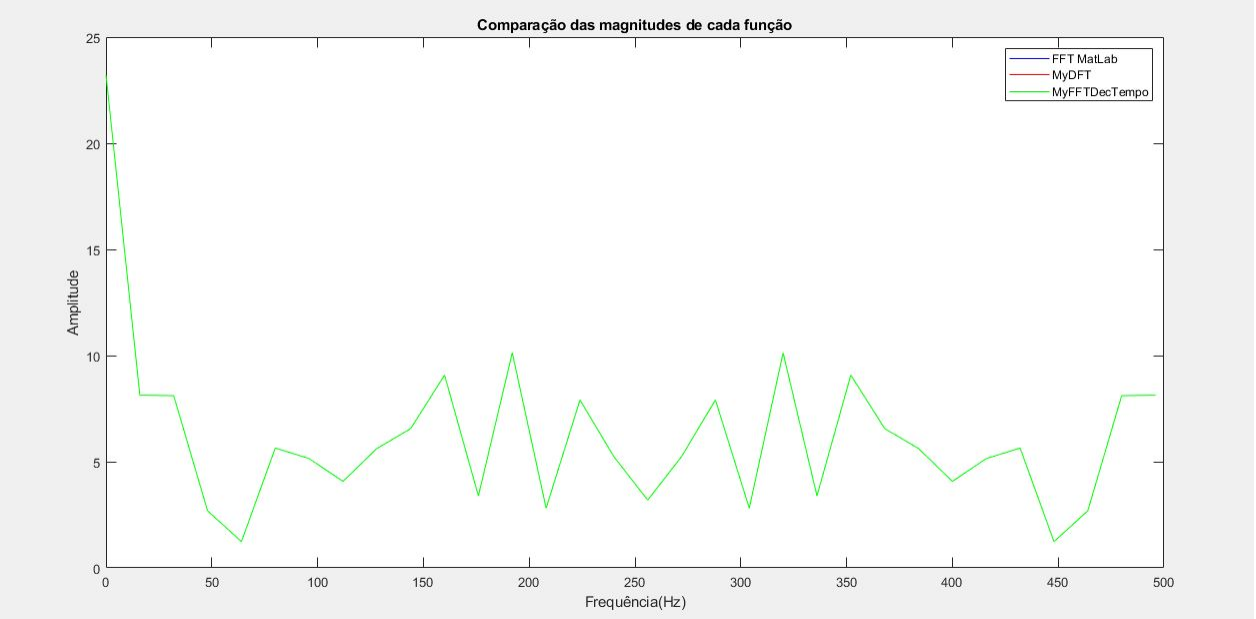
Os espectros de magnitude e fase para N=32 podem ser observados nas Figuras 18 e 19, respectivamente. Então, novamente, o espectro de magnitude das funções estão sobrepostas, indicando um erro muito baixo. Já no espectro de fase nota-se que há um erro considerável entre a FFT e a MyFFTDecTempo.

Figura 17: Erro entre os espectros para N=32



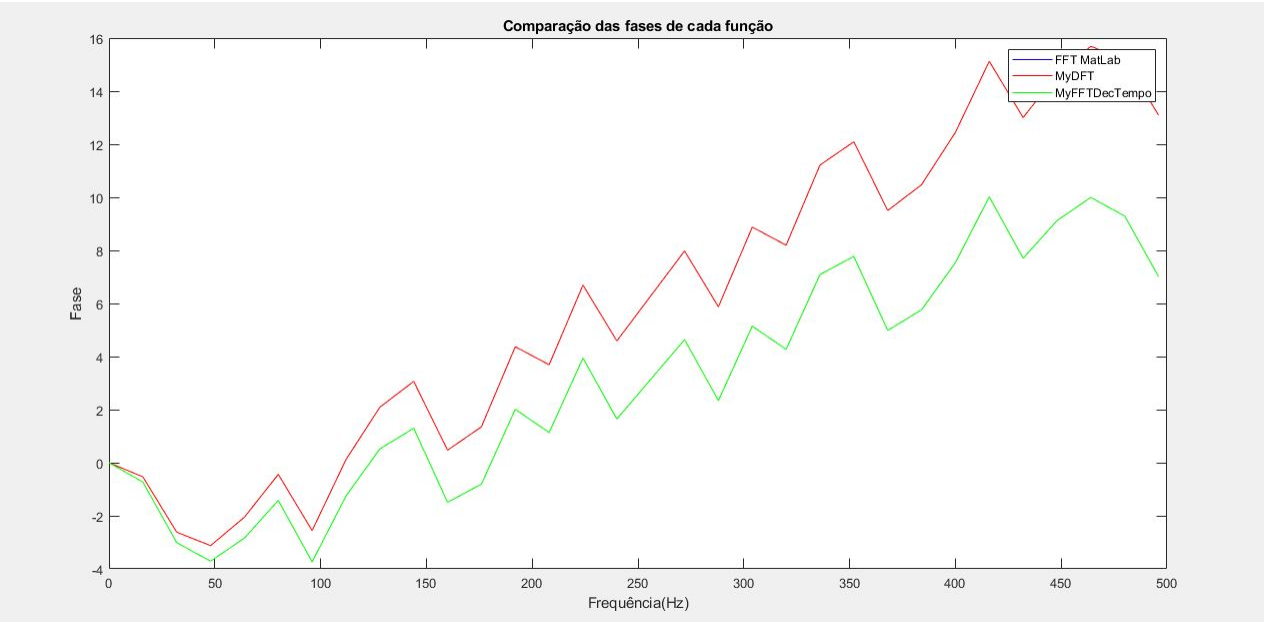
Fonte: do autor

Figura 18: Espectro de magnitude para N=32



Fonte: do autor

Figura 19: Espectro de fase para N=32



Fonte: do autor

5.1.6 Análise para N=64

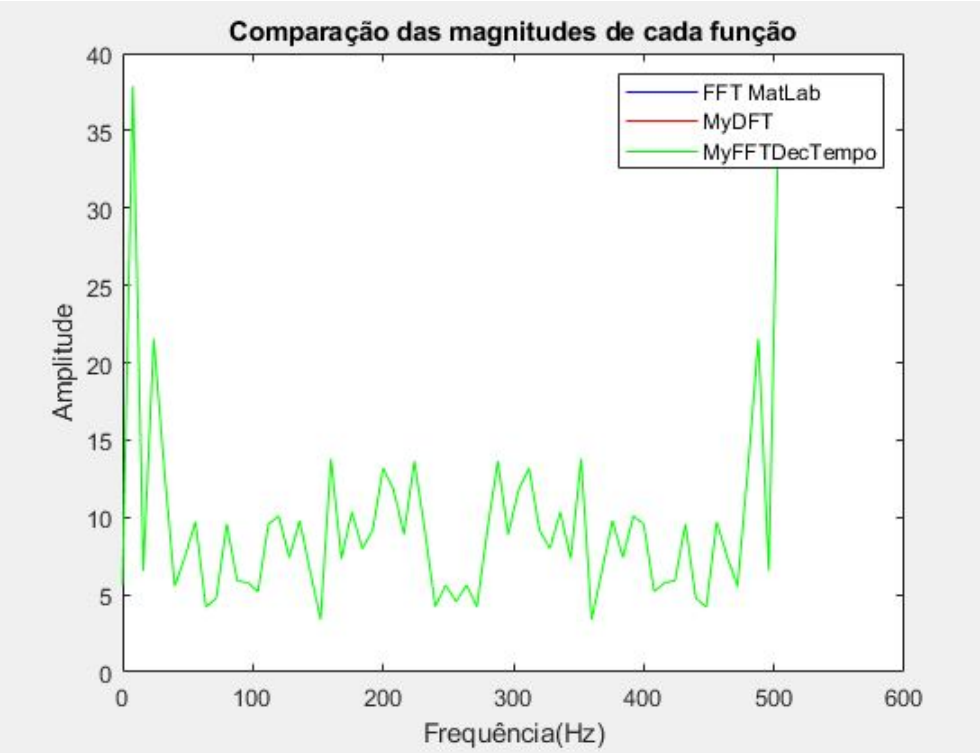
Os espectros de magnitude e fase podem ser observados nas Figuras 21 e 22. Novamente, observa-se que para os espectros de magnitude estão sobrepostos, indicando um erro nulo. Já para os espectros de fase vemos o mesmo padrão apresentado anteriormente. Há um erro significativo entre a FFT nativa e a MyFFTDecTempo.

Figura 20: Erro entre os espectros para N=64

erro1F	2.3572e-27
erro1M	7.8867e-25
erro2F	12.8527
erro2M	2.2116e-29
erro3F	12.8527
erro3M	7.8547e-25

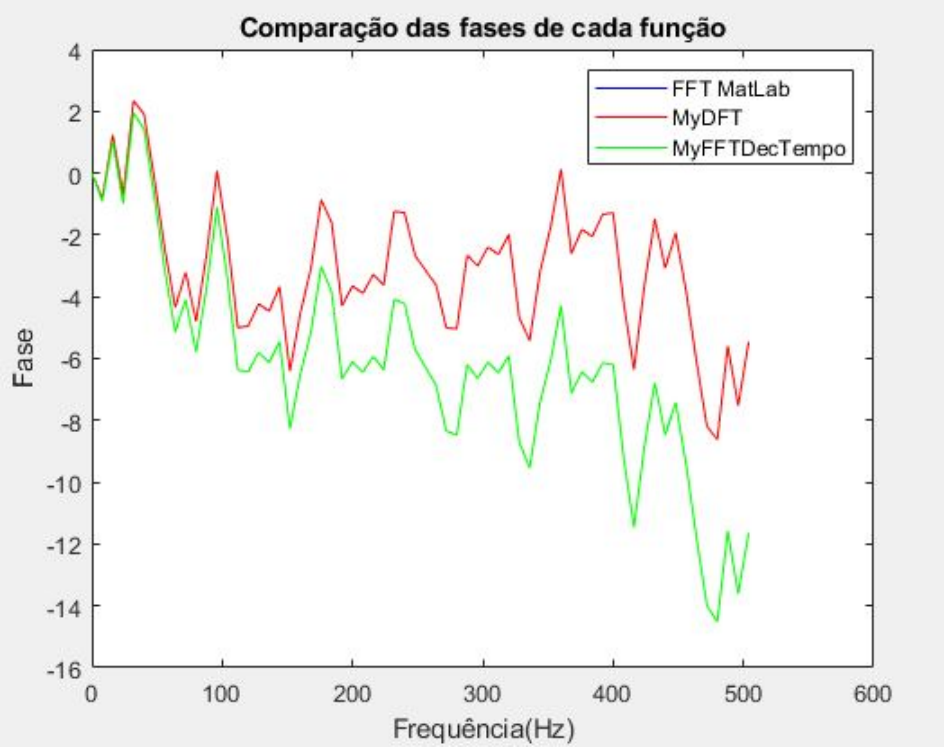
Fonte: do autor

Figura 21: Espectro de magnitude para N=64



Fonte: do autor

Figura 22: Espectro de fase para N=64



Fonte: do autor

5.1.7 Análise para N=128

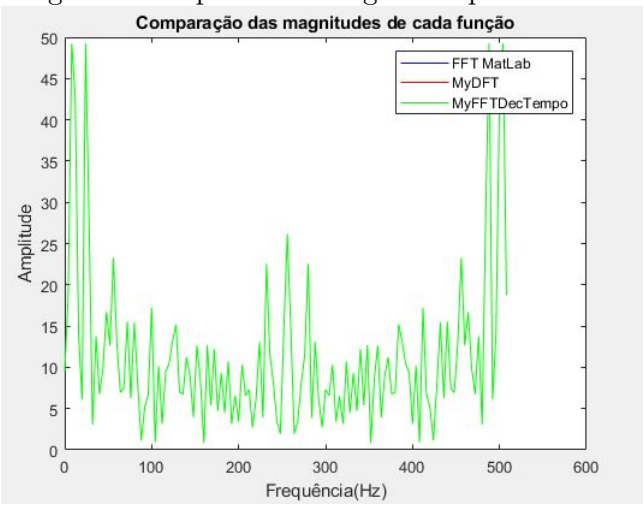
Os espectros de magnitude e fase podem ser observados nas Figuras 24 e 25. Novamente, observa-se que para os espectros de magnitude estão sobrepostos, indicando um erro nulo. Já para os espectros de fase vemos o mesmo padrão apresentado anteriormente. Há um erro significativo entre a FFT nativa e a MyFFTDecTempo.

Figura 23: Erro entre os espectros para N=128

erro1F	1.8960e-26
erro1M	4.1897e-24
erro2F	14.1623
erro2M	1.0220e-28
erro3F	14.1623
erro3M	4.1602e-24

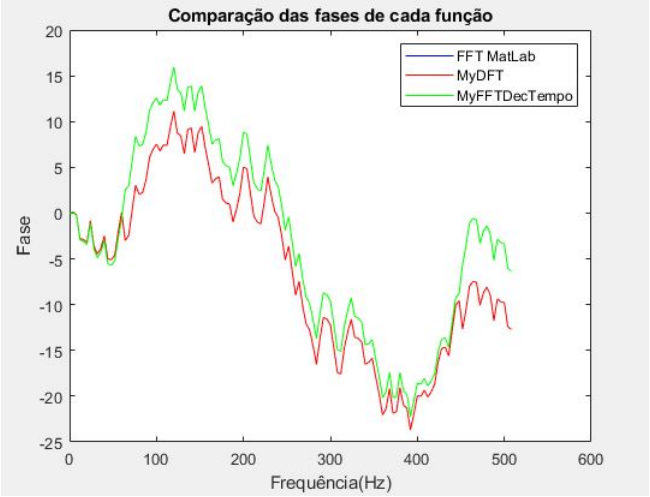
Fonte: do autor

Figura 24: Espectro de magnitude para N=128



Fonte: do autor

Figura 25: Espectro de fase para N=128



Fonte: do autor

5.1.8 Análise para N=256

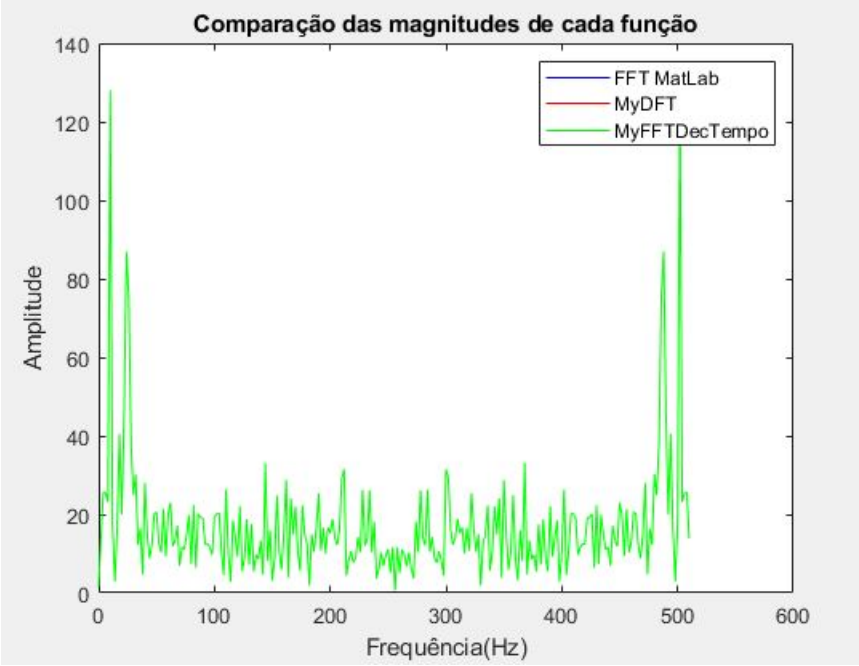
Os espectros de magnitude e fase podem ser observados nas Figuras 27 e 28. Novamente, observa-se que para os espectros de magnitude estão sobrepostos, indicando um erro nulo. Já para os espectros de fase vemos o mesmo padrão apresentado anteriormente. Há um erro significativo entre a FFT nativa e a MyFFTDecTempo.

Figura 26: Erro entre os espectros para N=256

erro1F	8.7854e-26
erro1M	4.5980e-23
erro2F	13.0825
erro2M	1.4047e-27
erro3F	13.0825
erro3M	4.5581e-23

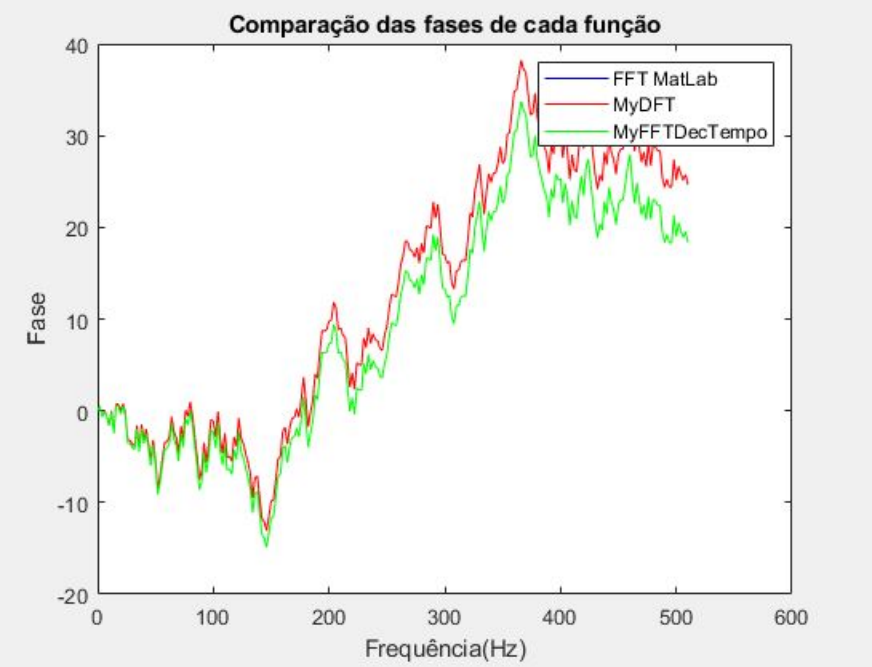
Fonte: do autor

Figura 27: Espectro de magnitude para N=256



Fonte: do autor

Figura 28: Espectro de fase para N=256



Fonte: do autor

5.1.9 Análise para N=512

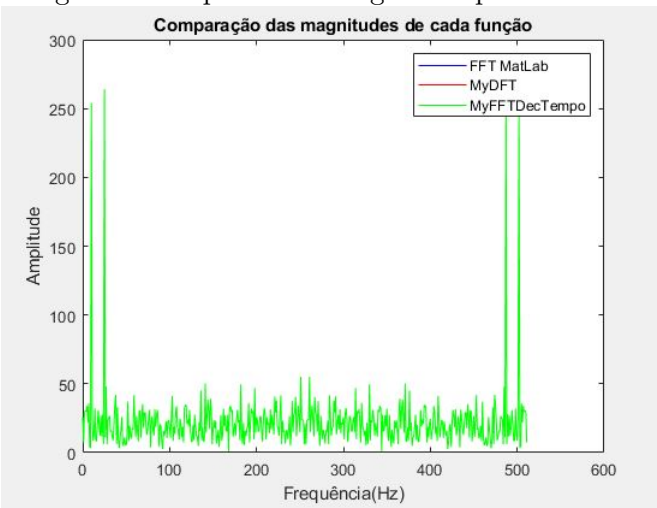
Os espectros de magnitude e fase podem ser observados nas Figuras 30 e 31. Novamente, observa-se que para os espectros de magnitude estão sobrepostos, indicando um erro nulo. Já para os espectros de fase vemos o mesmo padrão apresentado anteriormente. Há um erro significativo entre a FFT nativa e a MyFFTDecTempo.

Figura 29: Erro entre os espectros para N=512

erro1F	5.5256e-23
erro1M	2.0895e-20
erro2F	13.1209
erro2M	9.2586e-27
erro3F	13.1209
erro3M	2.0877e-20

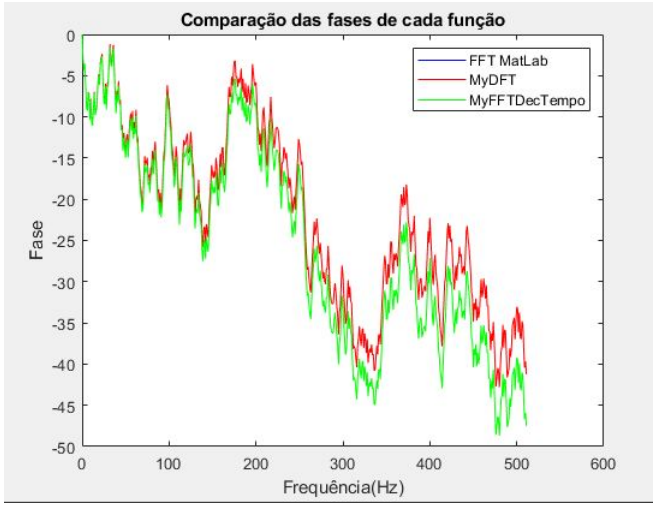
Fonte: do autor

Figura 30: Espectro de magnitude para N=512



Fonte: do autor

Figura 31: Espectro de fase para N=512



Fonte: do autor

5.1.10 Análise para N=1024

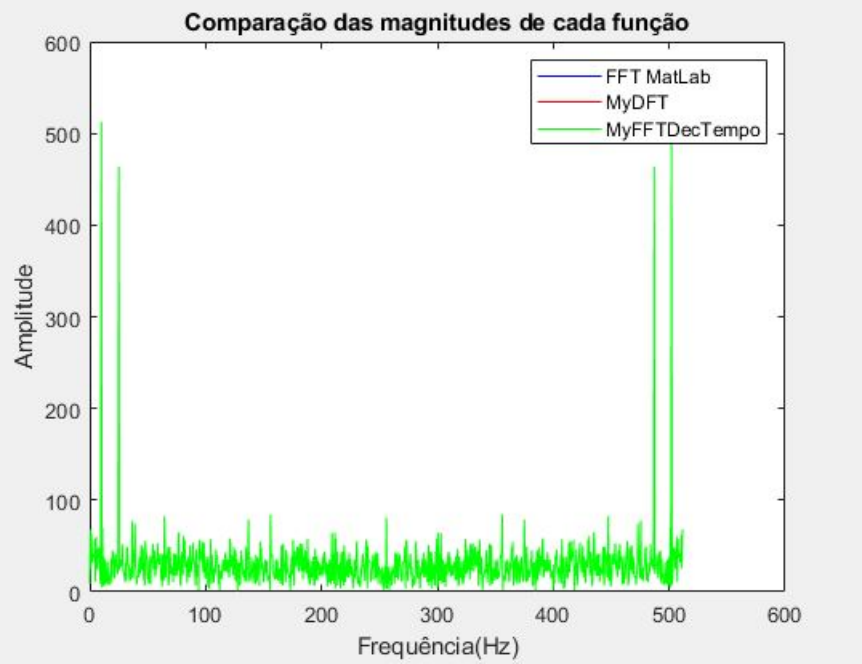
As Figuras 33 e 34 mostram os espectros de magnitude e fase para N=1024,respectivamente. Um observação interessante é que cada vez que se aumenta o tamanho de N, o erro das funções implementadas, MyDFT e MyFFTDecTempo, na fase diminui consideravelmente. Pois os espectros de fase estão quase sobrepostos, assim como os espectros de magnitude.

Figura 32: Erro entre os espectros para N=1024

erro1F	8.9515e-23
erro1M	1.5631e-19
erro2F	13.1402
erro2M	1.4069e-25
erro3F	13.1402
erro3M	1.5609e-19

Fonte: do autor

Figura 33: Espectro de magnitude para N=1024

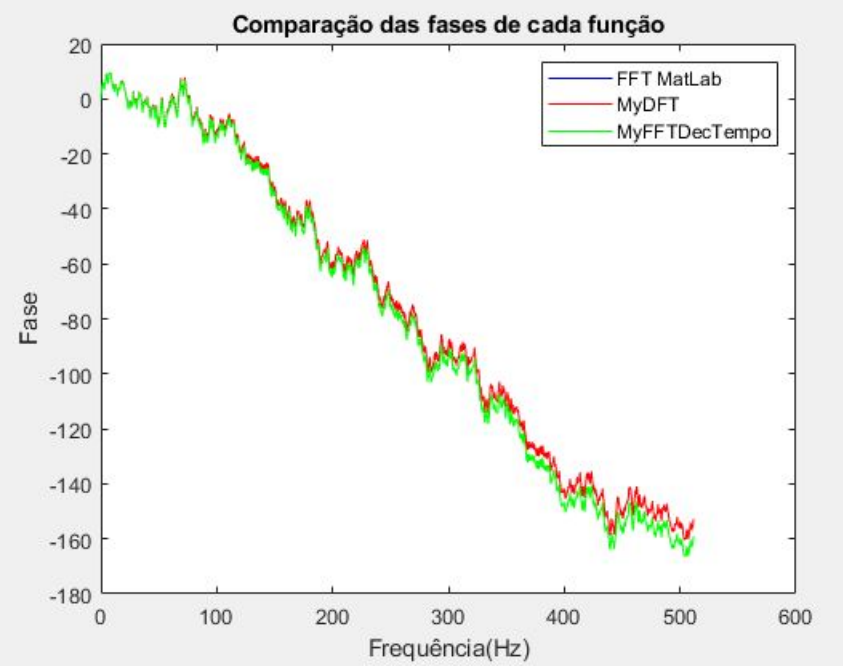


Fonte: do autor

5.2 Análise para N=2048

Por fim, as Figuras 36 e 37 mostram os espectros de magnitude e fase para N=2048, respectivamente. Nota-se que à medida que a janela N aumenta, os espectros, de cada função (FFT, MyDFT e MyFFTDecTempo) se sobrepõem tanto na magnitude quanto na fase. Ou seja, nota-se que o erro vai diminuindo com o aumento de N.

Figura 34: Espectro de fase para N=1024



Fonte: do autor

Figura 35: Erro entre os espectros para N=2048

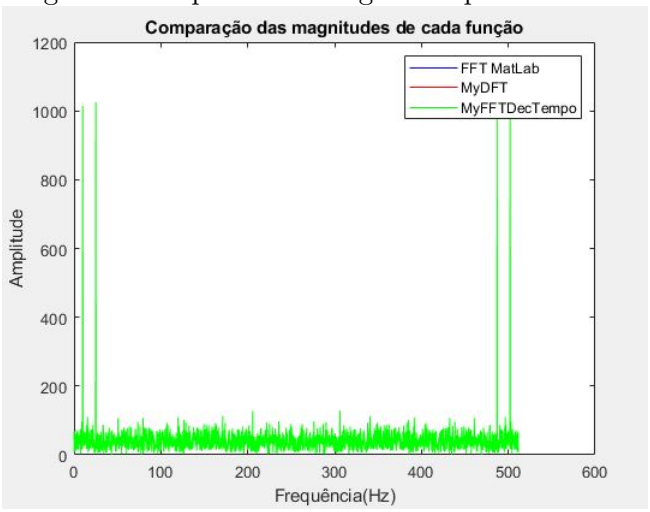
erro1F	1.1885e-20
erro1M	1.5107e-17
erro2F	13.9371
erro2M	1.8097e-25
erro3F	13.9371
erro3M	1.5107e-17

Fonte: do autor

5.3 Observações

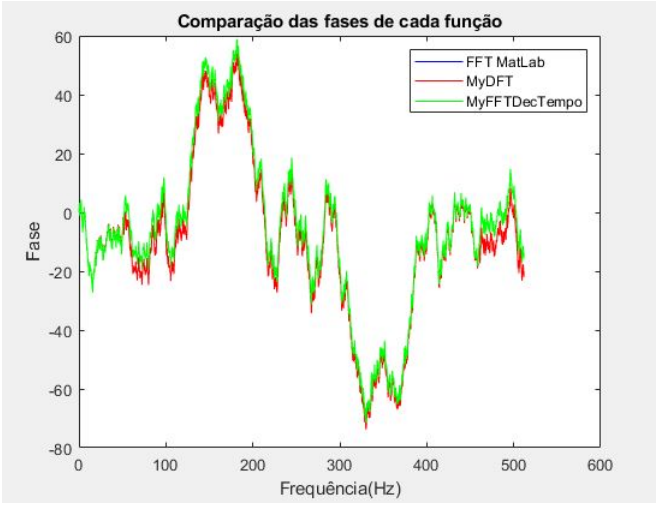
Apresentados todos os espectros de fase e magnitude para cada N, concluímos que a maior diferença se apresentou no espectro de fase. Nota-se que para N pequeno ($N < 128$) houve uma distorção no espectro de fase, da função MyFFTDecTempo em comparação com a FFT. E a medida que aumentamos o N essa diferença foi diminuindo até que os espectros de fases ficaram quase sobrepostos. O espectro de magnitude não apresentou diferenças ou variações para os diferentes valores de N.

Figura 36: Espectro de magnitude para N=2048



Fonte: do autor

Figura 37: Espectro de fase para N=2048



Fonte: do autor

5.4 Tabela de comparação

Com os dados obtidos na simulação para todos os N, foi preenchida a Tabela 1.

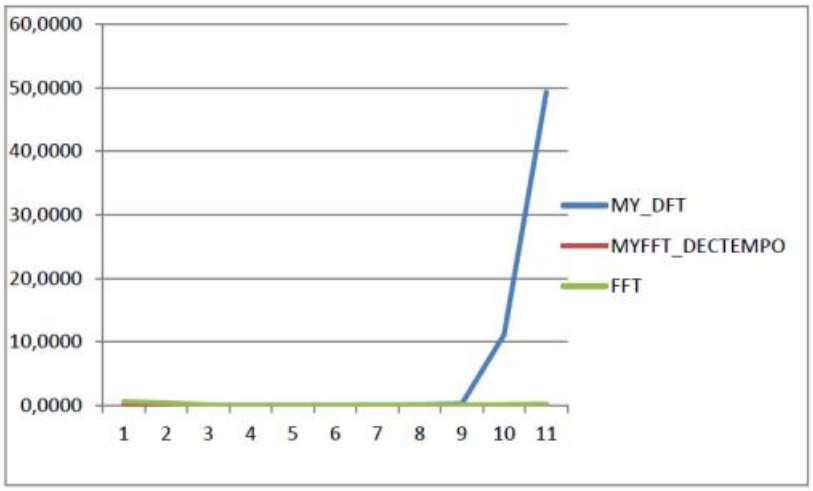
Também foi gerado um gráfico mostrando o tempo computacional, de cada função implementada, versus o N. E isso pode ser observado na Figura 38.

Tabela 1: Tempo computacional e erro

	Tempo Computacional			Erro Quadrático Médio	
N	MyDFT	MyFFTDecTempo	FFT	MyDFT vs FFT	MyFFTDectempo vs FFT
2	0,011	0,012	0,561	0	0,004
4	0,0001	0,0007	0,377	0	0,007
8	0,0001	0,0005	0,029	0	0,0078
16	0,0002	0,0004	0,041	0	0,0083
32	0,0006	0,0008	0,035	0	0,0095
64	0,0024	0,0016	0,026	0	0,0102
128	0,014	0,0037	0,035	0	0,0405
256	0,0526	0,0093	0,054	0,002	0,0088
512	0,2390	0,0176	0,067	0,018	0,0076
1024	11,320	0,0396	0,078	0,018	0,0054
2048	49,338	0,0801	0,157	0,024	0,0043

Fonte: do autor

Figura 38: Tempo computacional versus N



Fonte: do autor

6 Conclusão

Neste trabalho, foi solicitada a implementação utilizando o Cálculo Direto da Transformada de Fourier Discreta, (MyDFT) e a implementação do algoritmo da FFT com dizimação no tempo (MyFFTDecTempo). A partir de sinais gerados pelo somatório de senoides com frequências diversas e um sinal aleatório com N pontos - a serem variados ao decorrer do trabalho para comparação - foi solicitado a análise destas duas implementações realizadas com a função nativa do MatLab, a

FFT.

Percebe-se que o espectro de frequência obtido entre eles foi bastante satisfatório, obtendo erros de valores de ordem muito baixo para N maiores que 128. A maior discrepância obtida, visualizada pelos gráficos plotados e pelos valores apresentados na seção anterior foi no espectro de fase.

Outro ponto perceptível foi o tempo de execução das funções. Esperava-se um comportamento de maior custo operacional e menor eficiência para o algoritmo da MyDFT pela quantidade de multiplicações e adições requeridas e isto foi provado, principalmente, para N muito grandes, ressaltando a afirmação que ela se torna inviável para estes casos. As funções MyFFTDecTempo e FFT apresentaram grande semelhança para o tempo de execução em casos diversos de N.

7 Anexos: Algoritmos implementados

7.1 Main

```
1
2  % teste: sinal aleat rio + sen(2*pi*10*t) + sen(2*pi*25*t)
3  % com fs=512 e N=2,4,8,16,32,64,128,256,512,1024 e 2048)
4  clc
5  clear all
6  close all
7
8   N=2048; %AQUI VARIA-SE O N DESEJADO
9   fs =512;
10  T = 0: 1 /fs :((N/fs)-(1/fs ));
11  ws = 0:fs/N:fs-(fs/N);
12  x = randn (1,N);
13  x1 = sin(2*pi*10*T);
14  x2 = sin(2*pi*25*T);
15  X = x + x1 +x2;
16
17 %1) plot sinal X no tempo
18
19 figure()
20 plot(T,X)
21 title('Sinal de entrada no tempo')
22 xlabel('Tempo')
23 ylabel('Amplitude')
24
25
26 %2) Implementacao com FFT nativa matlab:
27
28  tic % contagem de tempo para execucao da FFT nativa
29  Y1 = fft(X,N);
30  mag1= abs(Y1);
31  timeFFT = toc;
```

```
 angle1= unwrap(angle(Y1));

 %3)Implementacao usando MyDFT:

 tic
 [dft_x,t]= MyDFT(X);
 mag2= abs(dft_x);
 timeMyDFT= toc;
 angle2= unwrap(angle(dft_x));

 %4)Implementacao usando MyFFT_DecTempo:

 tic
 [FFT_X]=MyFFT_DecTempo(X);
 mag3= abs(FFT_X);
 timeMyFFT=toc;
 angle3= unwrap(angle(FFT_X));

 %5)Plotagem das magnitudes para comparacao:

 figure()
 subplot(3,1,1)
 plot(ws,mag1,'b')
 title('Magnitude da FFT nativa do MatLab')
 xlabel('Frequ ncia(Hz)')
 ylabel('Amplitude')

subplot(3,1,2)
plot(ws,mag2,'r')
title('Magnitude MyDFY')
xlabel('Frequ ncia(Hz)')
ylabel('Amplitude')

subplot(3,1,3)
plot(ws,mag3,'g')
title('Magnitude MyFFTDecTempo')
xlabel('Frequ ncia(Hz)')
ylabel('Amplitude')


%6)Comparacao das magnitudes das funcoes juntas:

 figure()
 plot(ws,mag1,'b')
 hold on
```

```
 plot(ws,mag2,'r')
 hold on
 plot(ws,mag3,'g')
 title('Comparacao das magnitudes de cada funcao')
 xlabel('Frequ ncia(Hz)')
 ylabel('Amplitude')
 legend('FFT MatLab', 'MyDFT', 'MyFFTDecTempo')


 %7)Plot das fases separadas:

figure()
subplot(3,1,1)
plot(ws,angle1,'b')
title('Fase da FFT nativa do MatLab')
xlabel('Frequ ncia(Hz)')
ylabel('Fase')

subplot(3,1,2)
plot(ws,angle2,'r')
title('Fase MyDFY')
xlabel('Frequ ncia(Hz)')
ylabel('Fase')

subplot(3,1,3)
plot(ws,angle3,'g')
title('Fase MyFFTDecTempo')
xlabel('Frequ ncia(Hz)')
ylabel('Fase')

%8)Comparacao das fases juntas:

figure()
plot(ws,angle1,'b')
hold on
plot(ws,angle2,'r')
hold on
plot(ws,angle3,'g')
title('Comparacao das fases de cada funcao')
xlabel('Frequ ncia(Hz)')
ylabel('Fase')
legend('FFT MatLab', 'MyDFT', 'MyFFTDecTempo')

%9)Comparacao via erro:
% magnitude
```

```
erro1M=immse(mag1,mag2);
erro2M=immse(mag1,mag3);
erro3M=immse(mag2,mag3);
disp(erro1M)
disp(erro2M)
disp(erro3M)


%fase
erro1F=immse(angle1,angle2);
erro2F=immse(angle1,angle3);
erro3F=immse(angle2,angle3);
disp(erro1F)
disp(erro2F)
disp(erro3F)
```

7.2 Algoritmo MyDFT

```
% Implementacao da MyDFT:


function [dft_x,ws]=MyDFT(X)
fs=512;
N = length(X);
v=ceil(log2(N));
if (2^v ~= N)
    X = [X zeros(1,(2^v)-N)];
end

N=length(X);
wn =exp(-1*1j*(2*pi/N));
ws = 0:fs/N:fs-(fs/N);

for k=1:1:N
    soma=0;
    for n=1:1:N
        soma=X(n)*wn^((k-1)*(n-1)) +soma;
    end
    dft_x(k)=soma;
end

end
```

7.3 Algoritmo MyFFTDecTempo

```
function [FFT_X]=MyFFT_DecTempo(x)

%Verifica se o sinal de entrada    um vetor linha:
tam1=size(x);
if tam1(1,1)>tam1(1,2) %Se for maior    um vetor coluna
    x=x'; %Transposi  o
end

%Garantir que o comprimento do vetor de entrada seja pot ncia de 2
N=length(x); %N recebe o tamanho do sinal de entrada
vet=log2(N); % v recebe o log(N) na base 2
vet=ceil(vet); %Funcao ceil arrendonda o  n mero para o inteiro mais pr ximo
aux=2^vet; %Vari vel auxiliar para computar 2^v

if N~=aux %Se o tamanho do vetor  n  o    pot ncia de 2, deve portanto ser
    completado com zeros
    aux=aux-N;
    x=[x zeros(1,aux)];
end

n=length(x); %Armazena o tamanho do vetor x  ap s  verificar se o tamanho
    pot ncia de 2
Wn=exp((2*pi*1i)/n); %Armazena a exponecial complexa
w=1; %A  vari vel que armazenar  os valores multiplicados de Wn

if n==1 %Condi  o de parada para a recurs o, n=1 significa que o vetor x  s   tem
    um elemento e o c lculo terminou
    FFT_X=x;
else %Realizar  o c lculo novamente at  o vetor x tiver 1 elemento
    %% Separa em dois vetores par ou  mpar
    x1_par=zeros(1,length(x)/2);  %Armazena as posicoes pares de x
    x1_impar=zeros(1,length(x)/2); %Armazena as posicoes  mpares  de x

    cont1=1; %Contador para incrementar a posicao  do vetor par
    cont2=1; %Contador para incrementar a posicao do vetor impar

    for a=1:1:length(x) %Separacao de posicoes pares e  mpares
        if mod(a,2)==0   %Se o resto da divisao da posi  o por 2 for zero a
            posicao    par
            x1_par(1,cont1)=x(1,a); %Os valores pares  s  o armazenados no vetor par
            cont1=cont1+1; %Incrementa o contador
        else  %Caso contrario a posicao e impar
```

```
39              x1_impar(1,cont2)=x(1,a);  %Os valores impares sao armazenados no vetor
                    impar
40              cont2=cont2+1; %Incrementa o contador
41          end
42      end
43
44      y_par=MyFFT_DecTempo(x1_par); %Realiza a recursao do algoritmo para o vetor par
            subdividindo as operacoes
45      y_impar=MyFFT_DecTempo(x1_impar); %Realiza a recursao do algoritmo para o vetor
            impar subdividindo as operacoes
46
47
48      for k=0:1:((n/2)-1) %Usa do conceito de simetria para preencher o vetor
            frequencia
49          FFT_X(1,(k+1))=y_par(1,k+1)+w*y_impar(1,k+1); %Faz o preenchimento da metade
                 inferior da matriz de frequencia
50          FFT_X(1,((k+1)+n/2))=y_par(1,k+1)-w*y_impar(1,k+1); %Faz o preenchimento da
                metade superior da matriz de frequencia
51          w=w*Wn; %Faz o armazenamento dos valores de Wn a cada operacao
52      end
53  end
```

Referências

[1] Alan V Oppenheim e Ronald W Schafer. “Processamento em tempo discreto de sinais”. Em: *Tradução Daniel Vieira. 3ª ed.-São Paulo: Pearson Education do Brasil* (2012).