Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Aplicadas
Departamento de Engenharia Elétrica

Processamento Digital de Sinais

Transformada Rápida de Fourier - FFT

Ana Luiza Morais Oliveira
Carine Madeira Soares
Vitória Cordeiro Ventura

Professor - Frabricio Javier Erazo Costa

João Monlevade
22 de novembro de 2019

Sumário

1 Introdução **2**
1.1 Cálculo da Transformada de Fourier Discreta 2
1.2 Cálculo Direto pela Definição de TFD 2
1.3 Algoritmo de Goertzel 3
1.4 Algoritmos de FFT com Dizimação na Frequência 5

2 Algoritmos Implementados **7**
2.1 Algoritmo da Transformada Discreta de Fourier (DFT) 7
2.2 Algoritmo da Transformada Rápida de Fourier com Dizimação na Frequência 8

3 Resultados **11**
3.1 Cenário 1 11
3.2 Cenário 2 15

4 Análise dos Resultados **22**
4.1 Cenário 1 22
4.2 Cenário 2 23

5 Conclusão **24**

6 Referências **25**

1 Introdução

A Transformada de Fourier Discreta (TFD) é uma sequência, em vez de uma função de variável contínua, e corresponde a amostras em frequência, igualmente espaçadas, da TFTD do sinal.Além de sua importância teórica como uma representação de Fourier de sequências, a TFD desempenha um importante papel na implementação de uma variedade de algoritmos de processamento digital de sinais (OPPENHEIM, 2011).
O principal objetivo é analisar e encontrar algoritmos particularmente eficientes para implementação computacional da TFD de N pontos.
Existem várias maneiras de medir o custo e a eficiência de uma implementação ou algoritmo, e uma avaliação final depende tanto do tipo de implementação disponível quanto da aplicação pretendida. O número de multiplicações e adições aritméticas é um bom parâmetro como medida do custo computacional, uma vez que, estes parâmetros estão relacionados diretamente com a velocidade computacional.

1.1 Cálculo da Transformada de Fourier Discreta

A TFD de uma sequência finita de comprimento N é dada por:

$$x[k] = \sum_{k=0}^{N-1} \omega_N^{kn}, k = 0, 1, ...N-1 \tag{1}$$

em que $\omega_N = e^{\frac{-j2\pi}{N}}$.
A maioria das técnicas para melhorar a eficiência do cálculo da TFD explora as propriedades de simetria e periodicidade de ω_N^{kn}, que são:

- Simetria Complexa Conjugada :

$$\omega_N^{k(N-n)} = \omega_N^{-kn} = (\omega_N^{kn})* \tag{2}$$

- Periodicidade em n e k:

$$\omega_N^{kn} = \omega_N^{k(N+n)} = \omega_N^{(k+N)n} \tag{3}$$

  Como $\omega_N^{kn} = \cos\frac{2kn}{N} - j\sin\frac{2kn}{N}$, essas propriedades são, portanto, uma consequência direta da simetria e da periodicidade das funções seno e cosseno.

1.2 Cálculo Direto pela Definição de TFD

Considerando inicialmente o cálculo direto da definição dado pela expressão da TFD na Equação (1) e que a sequência $x[n]$ pode ser complexa,e que N multiplicações complexas e $(N1)$ adições complexas são requeridas para calcular cada valor da TFD se a Equação (1) na forma direta for utilizada. Para calcular todos os N valores, portanto, é preciso um total de N^2 multiplicações complexas e $N(N1)$ adições complexas.

$$x[k] = \sum_{n=0}^{N-1} [Re(x[n])Re(\omega_N^{kn}) - Im(x[n])Im(\omega_N^{kn})] + j[Re(x[n])Im(\omega_N^{kn}) + Im(x[n])Re(\omega_N^{kn})] \tag{4}$$

com $k = 0, 1, ...N-1$.
Ao observar a Equação (4) é visto que cada multiplicação complexa $x[n]\omega_N^{kn}$ requer quatro

multiplicações reais e duas adições reais, resultando em $4N$ multiplicações reais e $(4N2)$ adições reais para o cálculo de cada valor de $x[k]$. Para realizar o cálculo dos N valores da TFD são requeridas $4N_2$ multiplicações reais e $N(4N2)$ adições reais. Além das multiplicações e adições associadas à Equação (4) se faz necessário o armazenamento e acesso dos N valores da sequência de entrada complexa $x[n]$ e os valores dos coeficientes complexos ω_N^{kn}.
Devido a complexidade dos cálculos (custo proporcional a N^2) este método de cálculo da TFD se torna inviável para grandes valores de N.Logo, procedimentos de cálculo que reduzem o número de multiplicações e adições se tornam interessantes.
Usando a propriedade da simetria na Equação (2) é possível agrupar parcelas do somatório da Equação (4) para n e $(N-n)$.Ao utilizar estes agrupamentos em parcelas da Equação (4) o número de multiplicações pode ser reduzido aproximadamente por um fator 2. Também é possível assumir que, para certos valores do produto kn, as funções seno e cosseno implícitas assumem o valor 1 ou 0, eliminando assim a necessidade de multiplicações. Porém, mesmo após realizar esta simplificação a complexidade dos cálculos também é grande.

1.3 Algoritmo de Goertzel

A segunda propriedade (3) explora a periodicidade da sequência complexa e pode ser implementada com recursão para alcançar reduções de cálculo ainda mais significativas (OPPENHEIM, 2011).
O algoritmo de Goertzel é um típico exemplo de como a periodicidade da sequência complexa ω_N^{kn} pode ser usada para redução de cálculos. Para dedução do mesmo são feitas as seguintes considerações:

$$\omega_N^{-kn} = e^{\frac{2\pi}{N}Nk} = e^{j2\pi k} = 1 \tag{5}$$

já que k é um inteiro.Assim, multiplicando o membro direito da Equação (1) por ω_N^{kn} não afetará a equação.chegando em que:

$$x[k] = \omega_N^{kn} \sum_{r=0}^{N-1} x[r]\omega_N^{rn} = \sum_{r=0}^{N-1} x[r]\omega_N^{-k(N-r)} \tag{6}$$

Como exemplo, temos:

$$y_k[n] = \sum_{r=-\infty}^{\infty} x[r]\omega_N^{-k(N-r)}u[n-r] \tag{7}$$

em que $x[k] = y_k[n]|_{n=N*}$.O diagrama de fluxo de sinais da resposta ao impulso do exemplo pode ser observado na Figura 1:

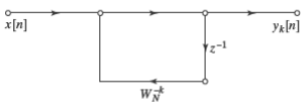

Figura 1: Diagrama de fluxo do cálculo recursivo complexo de primeira ordem de x[k].

Como a entrada geral $x[n]$ e o coeficiente ω_N^{kn} são ambos complexos, o cálculo de cada novo valor de $y_k[n]$ usando o sistema da Figura 1 requer 4 multiplicações reais e 4 adições reais. Esse procedimento é ligeiramente menos eficiente que o método direto , porém ele evita o cálculo ou o armazenamento dos coeficientes ω_N^{kn} , pois estes são calculados pela recursão sugerida na Figura 1.
É possível reter essa simplificação ao mesmo tempo em que se reduz o número de multiplicações por um fator 2. Isso pode ser feito manipulando a função de sistema da Figura 1.

$$H_k[z] = \frac{1}{1 - \omega_N^{-k} z^{-1}} \tag{8}$$

Multiplicando o numerador e o denominador por $(1 - \omega_N^{-k} z^{-1})$, temos:

$$H_k[z] = \frac{(1 - \omega_N^{-k} z^{-1})}{(1 - \omega_N^{-k} z^{-1})(1 - \omega_N^{-k} z^{-1})} \tag{9}$$

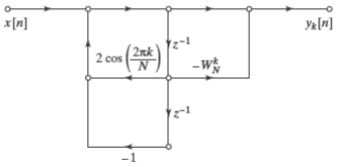

Figura 2: Diagrama de fluxo do cálculo recursivo complexo de segunda ordem de x[k]-Algoritmo de Goertzel.

O diagrama de fluxo de sinais da Figura 2 corresponde a implementação na forma direta II da função de sistema da Equação (9).
Se a saída for complexa, apenas duas multiplicações reais por amostra são necessárias para a implementação dos polos desse sistema, pois os coeficientes são reais e o fator 1 não precisa ser contado como uma multiplicação. No caso do sistema de primeira ordem, para uma entrada complexa, quatro adições reais por amostra são necessárias para implementar os polos se a entrada for complexa.
O total de operações é de $2N$ multiplicações reais e $4N$ adições reais para os polos,mais 4 multiplicações reais e 4 adições reais para o zero. Então o total de operações é $2(N + 2)$ multiplicações reais e $4(N + 1)$ adições reais, cerca da metade do número de multiplicações reais requeridas pelo método direto. Nesse esquema mais eficiente, ainda tem-se a vantagem de que cos $\frac{2\pi k}{N}$ e ω_N^{kn} são os únicos coeficientes que precisam ser calculados e armazenados. Os coeficientes ω_N^{kn} são novamente calculados implicitamente na iteração da fórmula de recursão implicada pela Figura 2.
Nem no método direto nem no algoritmo de Goertzel precisa-se calcular $x[k]$ em todos os N valores de k. De fato, pode-se calcular $x[k]$ para quaisquer M valores de k, sendo cada valor de TFD calculado por um sistema recursivo na forma da Figura 2 com os coeficientes apropriados. Nesse caso, o cálculo total é proporcional a NM. O método de Goertzel e o método direto são atraentes quando M é pequeno, porém, como indicado anteriormente, existem algoritmos para os quais o cálculo é proporcional a $N \log_2 N$ quando N é uma potência de 2. Portanto, quando M é menor do que $\log_2 N$, o algoritmo de Goertzel ou o cálculo direto da TFD pode ser, de fato, o método mais eficiente, mas quando todos os N valores de x[k] são requeridos, os algoritmos de dizimação na frequência apresentado

posteriormente, são aproximadamente $\frac{N}{\log_2 N}$ mais eficientes que o método apresentado e o algoritmo de Goertzel.

1.4 Algoritmos de FFT com Dizimação na Frequência

Os algoritmos de FFT com dizimação na frequência são baseados na divisão da sequência da TFD $x[k]$ em subsequências cada vez menores (OPPENHEIM, 2011).
Novamente esta classe de algoritmos FFT fica restringida a discussão de complexidade de N sendo uma potência de 2. O cálculo será executado separadamente com $\frac{N}{2}$ amostras de frequência com índice par e $\frac{N}{2}$ amostras de frequência com índice ímpar.
Uma representação básica deste algoritmo pode ser observada na Figura 3 em que $X_0[k] = X[2k]$ e $X_1[k] = X[2k+1]$

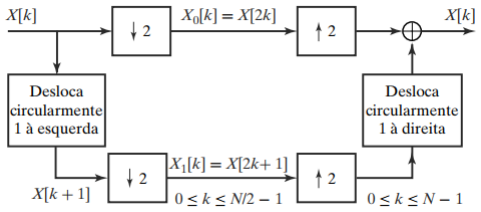

Figura 3: Exemplo do princípio básico da dizimação na frequência

È importante lembrar que a TFD $X[k]$ é implicitamente periódica,com período N.Assim, o sinal no domínio do tempo periódico implícito representado por $X_0[k] = X[2k]$ é:

$$x_0[n] = \sum_{m=-\infty}^{\infty} x[n + \frac{mN}{2}], -\infty < n < \infty \tag{10}$$

Como $x[n]$ tem comprimento N, somente duas das cópias deslocadas de $x[n]$ se sobrepõe no intervalo $0n\frac{N}{2-1}$, de modo que a sequência correspondente de tamanho finito $x_0[n]$ é:

$$x_0[n] = x[n] + x[n + \frac{N}{2}], 0 < n < \frac{N}{2-1} \tag{11}$$

Assim, a sequência de $\frac{N}{2}$ pontos $x_1[n]$ correspondente a $X_1[k] = X[2k+1]$ é:

$$x_1[n] = (x[n] - x[n + \frac{N}{2}])\omega_N^n, 0 < n < \frac{N}{2-1} \tag{12}$$

já que $\omega_N^{\frac{N}{2}} = -1$.
Das Equações 11 e 12 segue que:

$$X_0[k] = \sum_{n=0}^{N/2-1} x[n] + x[n + \frac{N}{2}]\omega_{N/2}^{kn} \tag{13}$$

$$X_1[k] = \sum_{n=0}^{N/2-1} (x[n] - x[n + \frac{N}{2}]\omega_N^n)\omega_{N/2}^{kn} \tag{14}$$

A Equação 13 é uma TFD de N/2 pontos da sequência $x_0[n]$ e a Equação (14) é a TFD de N/2 pontos da sequência $x_1[n]$. Assim, utilizando as equações, os pontos de saída com

índice par e ímpar de $X[k]$ podem ser calculados, já que $X[2k] = X_0[k]$ e $X[2k+1] = X_1[k]$ respectivamente. O procedimento sugerido pelas equações é descrito na Figura 4.

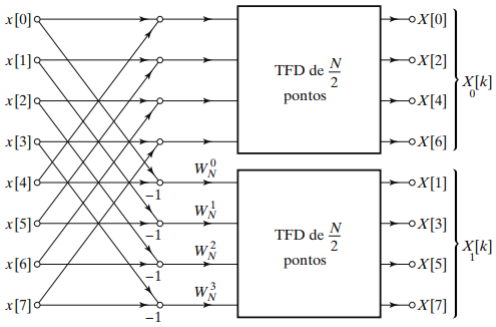

Figura 4: Diagrama de fluxo da decomposição para dizimação na frequência, de um cálculo de TFD de N pontos em dois cálculos de TFD de (N/2) pontos (N=8).

Nota-se que, sendo N uma potência de 2, N/2 é divisível por 2, de modo que as TFDs de (N/2) pontos podem ser computadas por meio do cálculo separado dos pontos de saída com índice par e ímpar dessas TFDs. As Figuras correspondentes são ilustradas a seguir:

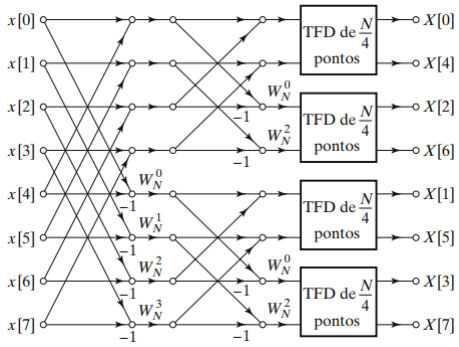

Figura 5: Diagrama de fluxo da decomposição para dizimação na frequência de uma TFD de 8 pontos em quatro cálculos de TFD de 2 pontos.

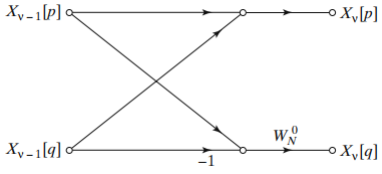

Figura 6: Diagrama de fluxo de uma TFD de 2 pontos típica como requerida no último estágio da decomposição para dizimação na frequência.

Pela contagem das operações aritméticas na Figura 4 e generalizando para N = 2v, percebe-se que o cálculo requer $(N+2)\log_2 N$ multiplicações complexas e $N\log_2 N$ adições complexas. Logo, o número total de cálculos é o mesmo para os algoritmos de dizimação na frequência e dizimação no tempo.

2 Algoritmos Implementados

Nesta seção serão apresentadas as implementações do algoritmo da Transformada Direta de Fourier (DFT) assim como a forma de dizimação na frequência. Também será apresentados também todos os pseudocódigos e o detalhamento de todas as funções realizadas.

2.1 Algoritmo da Transformada Discreta de Fourier (DFT)

A primeira implementação foi feita utilizando a fórmula direta da transformada discreta de Fourier (1) e o algoritmo implementado pode ser visualizado a seguir:

```
    function [Y,ws] = MyDFT(X,fs)

 %Atribuicoes primarias:

 N = length(X);
 ws = 0:fs/N:fs-(fs/N);

 %Implementacao da DFT via formula direta:

 for k=1:1:N
  somaX = 0;
   for n=1:1:N
    somaX = X(n)*exp(-1*1j*(2*pi/N)*(k-1)*(n-1))+ somaX;
 end
 Y(k)= somaX;
 end
 end
```

O pseudocódigo do algoritmo *MyDFT* é:

```
função MyDFT(X,Fs)
 N ← tamanho de X;
 ws ← vetor de frequências correspondente à frequência de amostragem Fs;
 para k ← 1 até N faça
 somaX ← 0;
 para n ← 1 até N faça
 somaX ← X(n)e^{j(2/N)(k1)(n1)} + somaX;
 fim para
 Y (k) ← somaX ;
 fim para
 retorna Y,ws;
 fim função
```

A função implementada recebe como entradas o sinal ao qual se deseja aplicar a transformada de Fourier (X) e a frequência de amostragem utilizada na transformação do mesmo (fs). As saídas são o próprio sinal transformado (Y) e um vetor de frequências

correspondente à frequência de amostragem adotada (ws).
Toda a função segue a equação (1) e é importante salientar que a mesma não conta com quaisquer mecanismos de redução de custos e tempo computacional. A função possui dois laços de repetição. O laço mais externo é responsável por percorrer todos os N valores do vetor de entrada afim de preencher todas as N posições do vetor de saída. O somatório interno é responsável por fazer as N multiplicações pela exponencial e assim obter o valor correspondente à posição N no vetor de saída.
Como o sinal de entrada $x[n]$ pode ser um número complexo, N multiplicações complexas e $(N1)$ adições complexas são requeridas para calcular cada valor da TFD. Para o cálculo de todos os N valores, portanto, é preciso um total de N^2 multiplicações complexas e $N(N1)$ adições complexas.

2.2 Algoritmo da Transformada Rápida de Fourier com Dizimação na Frequência

A segunda implementação realizada é correspondente ao algoritmo da transformada rápida de Fourier com dizimação na frequência. As classes de dizimação em particular, contam com mecanismo de análise capazes de identificar possíveis erros e transformarem os sinais de entradas em potências de 2.

```
    %Esta função é responsável por realizar a transformada discreta de Fourier
%com dizimação na frequência. A função é responsável por analisar o número
%de estágios de cálculo e calcular a transformada de Fourier
%com base nestes estágios. Os parâmetros de entrada serão o
%sinal original a ser transformado (X) e o parâmetro
%de saída será o sinal transformado (FFTDF).

function [FFTDF] = MyFFT_DecFreq(X)

%Análise do sinal de entrada:
%A primeira análise do sinal de entrada é muito importante pois a dizimação
%no tempo só é feita a partir de sinais multiplos de 2, ou seja, o sinal de
%entrada deve ser uma potência de 2 para a dizimação ser realizada. Desta
%maneira será verificado se o sinal é ou não uma potência de 2 e em caso
%negativo será utilizado o mecanismo de preenchimento de zeros
%(zero padding) afim de tornar o sinal uma potência de 2.

N = length(X);       %Variável utilizada para guardar o tamanho
%do sinal de entrada.
NP = ceil(log2(N));  %O comando ceil retorna a potência
%de 2 mais próxima do valor de entrada atual.
%Por exemplo, um sinal de 2000 pontos, a próxima
%potência de 2 seria 2048 o que daria 2^11 e então NP seria 11.

if (2^NP ~= N)
    X = [X zeros(1,(2^NP)- N)];% Preenchimento com zeros até chegar a 2^NP.
end

N = length(X); %Agora o sinal já é uma potência de 2 e poderá ser utilizado
```

```
%o mecanismo de dizimação no tempo.

%DIZIMAÇÃO NA FREQUÊNCIA:

NE = log2(N); %A variável NE é responsável por contar o número de estágios
%de cálculo que serão utilizados para resolução do problema. Como exemplo,
%um sinal de 2048 pontos pode ser dividido 11 vezes, ou seja, 11 estágios
%de cálculo serão utilizados.

VE = 1:1:NE; %Como a dizimação na frequência será realizada na ordem normal,
%o primeiro estágio será o último e assim sucessivamente, logo é necessário
%inverter a ordem dos estágios, assim:
VE = wrev(VE);

k = 0:1:N-1;%Variável utilizada para determinação dos
%índices dos coeficientes.
Wn = exp(-1j*2*pi*k/N); % Coeficientes Wn utilizados na FFT.


%EXECUÇÃO DO FLUXO EM BORBOLETA

for n=VE:-1:1 %Laço de repetição utilizado
    %para contagem dos estágios de cálculo.

    %A parcela 2^n corresponde à amostra envolvida em cada estágio. No caso
    %em que NE = 1 as amostras são feitas de 2 em 2, e assim sucessivamente.

    %A parcela N/2^n corresponde ao expoente dos coeficientes Wn utilizados.
    %Se N/2^n = 10 então os coeficientes serão calculados W^0 W^10 W^20, e
    %assim sucessivamente.

    WnAUX=Wn([1:(N/2^n):(N/2)]);
    %Percorrer os coeficientes ao passo de N/2^n que é dado pelos
    %coeficientes.

    for k=1:(N/2^n)

        AUX = X(k*2^n - 2^n+1:k*2^n); %Blocos de Dados em forma matricial.
        AUXUP = AUX(1:end/2);         %Porcao superior da matriz de dados.
        AUXDOWN = AUX((end/2)+1:end); %Porcao inferior da matriz de dados.

        %Operacao Borboleta entre porcao superior e inferior:
        RUP = AUXUP + AUXDOWN;

        %Operacao Borboleta entre porcao inferior e superior:
        RDOWN = (AUXUP - AUXDOWN).*WnAUX;

        %Resultado da operacao local:
```

```
        X(k*2^n - 2^n+1:k*2^n)=[RUP RDOWN];
    end
end
%Para o cálculo da FFT com dizimação na frequência,
%a entrada do sinal será na
%ordem NORMAL e a saída na ordem BIT-REVERSA. Logo:

FFTDF = bitrevorder(X);  %Saida na ordem BIT-REVERSA

end
```

Segue o pseudocódigo do algoritmo $MyFFT_DecFreq$:

```
     função My_FFT_DecFreq(X)
 N ← tamanho do sinal de entrada;
 NP ← índice inteiro da próxima potência de 2 do sinal de entrada;
 se 2 NP 6= N então

 X ← completa com zeros até que o tamanho seja correspondente à proxima
potência de 2 (NP);
 fim se
 NE ← número de estágios de cálculo;
 V E ← vetor com estágios de cálculo;
 V E ← VE invertido;
 k ← vetor de índices para os coeficientes Wn (0 a N-1);
 W n ← e^{j2k/N} ; . Determinação dos coeficientes Wn

 para n ← 1 até NE
 faça:W nAUX ← W_n([1 : (N/2^{2}) : N/2]);.Vetor auxiliar para operação borboleta

 para k ← 1 até N/2^{n}
faça
 AUX ← X(k  2n  2n + 1 : k  2^n);. Bloco de dados em forma matricial
 AUXUP ← porção superior da matriz de dados;
 AUXDOW N ← porção inferior da matriz de dados;
 RUP ← AUXUP + AUXDOWN; . Operacao Borboleta entre os valores da porcao superior e
 RDOW N ← (AUXP - AUXDOWN)*WnAUX; . Operacao Borboleta
entre os valores da porcao inferior e superior da matriz de dados
 X(k 2n  2n + 1 : k  2^n) ← concatena RUP e RDOWN; . Operação realizada localmente
 fim para
 fim para
 X ← X na ordem BIT-REVERSA; . A saida se encontra na ordem
BIT-REVERSA
 FFTDF ← X;
 retorna FFTDF;
 fim função
```

3 Resultados

Neste capítulo trataremos sobre os resultados obtidos tanto do cenário um quanto no dois.A análise das figuras serão feitas no capítulo seguinte.

3.1 Cenário 1

O primeiro caso de análise é composto por um sinal aleatório de 2000 pontos somado a três senoides ($15Hz$, $45Hz$ e $70Hz$). Neste cenário de testes foram utilizadas tanto a função fft, do próprio software de implementação MATLAB, quanto as funções implementadas MyDFT e MyFFT-DecFreq.
Os principais pontos de análises estão no erro quadrático médio entre as funções implementadas e a função já existente no software. Além da análise dos erros foram analisados os espectros de fase e magnitude entre todas as funções.
A Figura 7 ilustra o sinal de entrada do cenário 1 composto pela soma entre um sinal

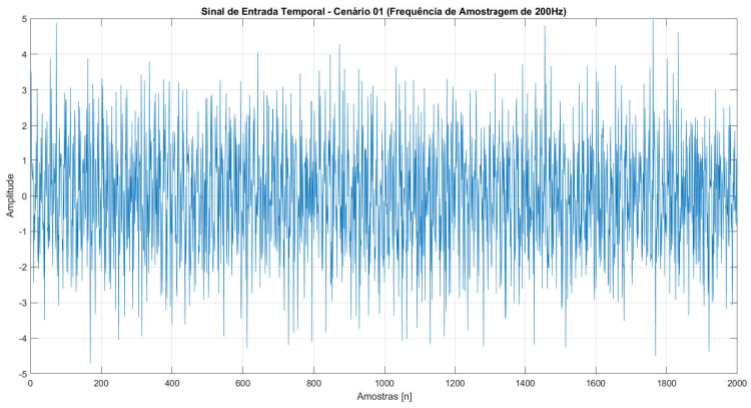

Figura 7: Sinal de Entrada Cenário 1

aleatório com 2000 pontos e três senoides de frequências 15Hz, 45Hz e 70Hz. A frequência de amostragem para todos os testes feitos neste cenário foi de 200Hz.

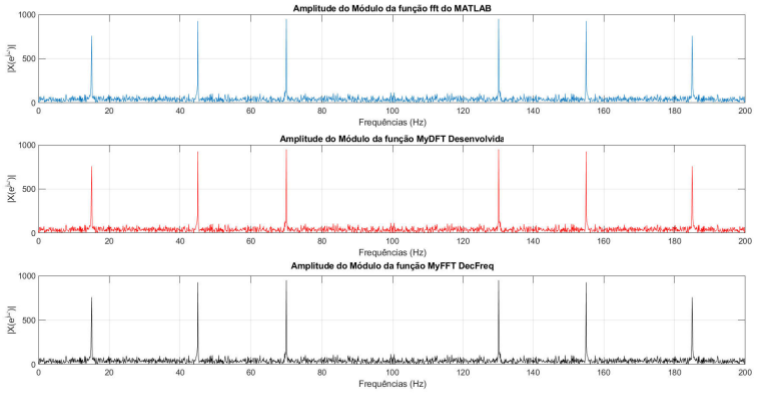

Figura 8: Comparação entre as magnitudes das diferentes implementações

Na Figura 8 está ilustrado as diferentes magnitudes das implementações das Transformadas de Fourier.Em azul temos a própria fft do MATLAB, em vermelho é a magnitude da função implementada My-DFT e em preto a MyFFT-DecFreq.

Na Figura 9 estão ilustradas todas as magnitudes contidas na Figura 8 porém agora

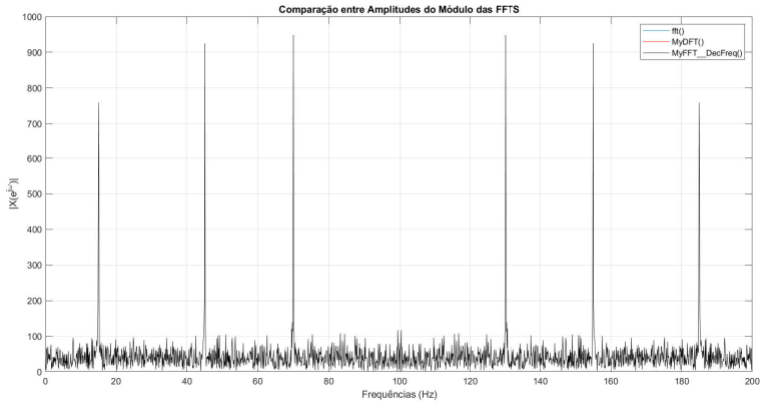

Figura 9: Comparação entre as magnitudes das diferentes implementações das Transformadas Rápidas de Fourier

plotadas em um mesmo gráfico. Vale salientar que ocorre sobreposição pelo fato do erro ser mínimo entre as implementações.

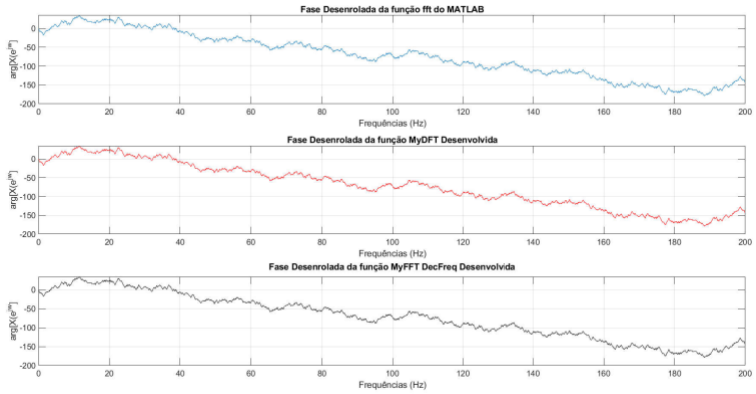

Figura 10: Comparação entre as fases das diferentes implementações

Na Figura 10 está ilustrado as diferentes fases das implementações das Transformadas de Fourier.Em azul temos a própria fft do MATLAB, em vermelho é a magnitude da função implementada My-DFT e em preto a MyFFT-DecFreq.

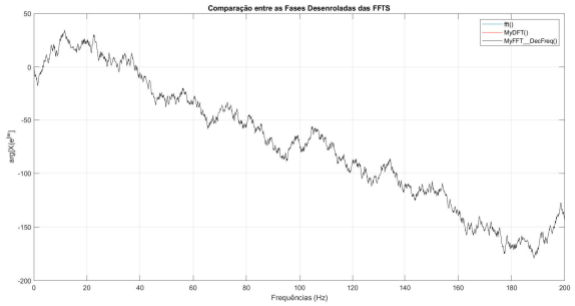

Figura 11: Comparação entre as fases das diferentes implementações

Na Figura 11 estão ilustradas todas as magnitudes contidas na Figura 10 porém agora plotadas em um mesmo gráfico. Vale salientar que ocorre sobreposição pelo fato do erro ser mínimo entre as implementações.

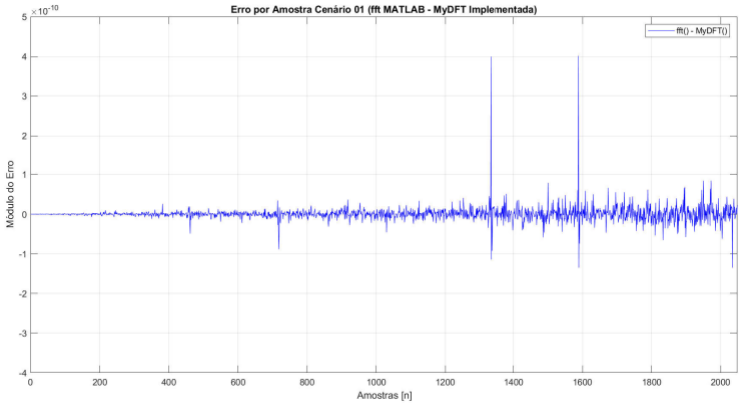

Figura 12: Erro por Amostra

Na Figura 12 observa-se o gráfico do erro por amostra entre a função fft do MATLAB e a função implementada My-DFT.
Na Figura 13 é ilustrado o erro por amostra entre a função fft e a função MyFFT-DecFreq

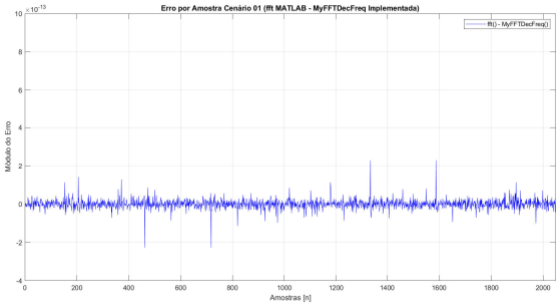

Figura 13: Erro por Amostra

em cada amostra do sinal transformado. O erro por amostra é obtido através da subtração entre o módulo da fft e o módulo de MyFFT-DecFreq em cada uma das 2048 amostras.

Além das análises em magnitude e fase das Transformadas Rápidas de Fourier implementadas foram analisados os tempos computacionais entre cada implementação e o erro quadrático médio entre as mesmas.Estes parâmetros são ilustrados na tabela a seguir:

fft	MyDFT	MyFFT-DecFreq
0,0073	1,1571	0,0420

Tabela 1: Tempo Computacional em segundos - Cenário 1

fft x MyDFT	fft x MyFFT-DecFreq	MyDFT x MyFFT-DecFreq
3,3872 e^{-19}	7,3441e^{-25}	3,3868e^{-19}

Tabela 2: Erros Quadráticos Médios- Cenário 1

3.2 Cenário 2

O segundo cenário de testes realizado é composto por um sinal aleatório somado a duas senoides de $10Hz$ e $25Hz$ com uma frequência de amostragem fs de $512Hz$. Foram utilizados sinais com N (número de pontos) = 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048 e foram encontrados os espectros de fase e frequência utilizando tanto a função fft do próprio MATLAB quanto as funções MyDFT e MyFFT-DecFreq implementadas.
Afim de não estender ao máximo o relatório serão descritos apenas os gráficos relevantes, correspondentes a sinais com $N = 32$, 256, 1024 e 2048 pontos. Todas as análises serão feitas no capítulo seguinte.

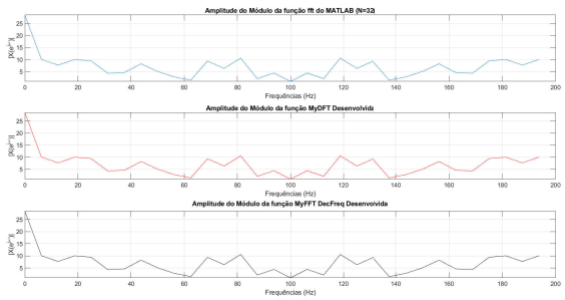

Figura 14: Comparação entre as Magnitudes das diferentes implementações da Transformada Rápida de Fourier para N=32 pontos

Tanto a Figura 14 quanto a Figura 15 estão ilustrando a comparação entre as Magnitudes entre as funções implementadas e a função já existente no MATLAB para um sinal aleatório com 32 pontos. Observa-se nestes gráficos que a frequência de amostragem é equivalente a 512Hz.
E na Figura 15 foram plotadas todas as resposta em um mesmo gráfico.

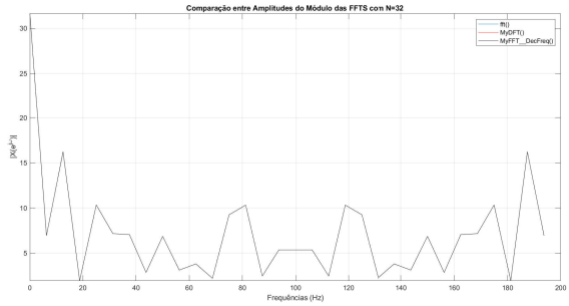

Figura 15: Comparação entre as Magnitudes

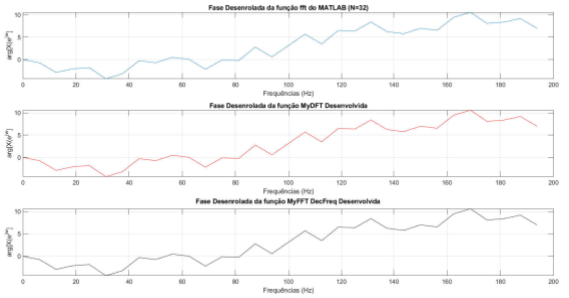

Figura 16: Comparação entre as fases desenroladas das diferentes implementações

Nas Figuras 15 e 16 ,assim como na comparação entre as magnitudes, estão apresentadas as comparações entre as fases desenroladas oriundas das diferentes implementações da Transformada Rápida de Fourier. Na Figura 16 são plotados gráficos separados onde o gráfico em azul representa a função fft do MATLAB, em vermelho a função MyDFT implementada e em preto a função e MyFFT-DecFreq.

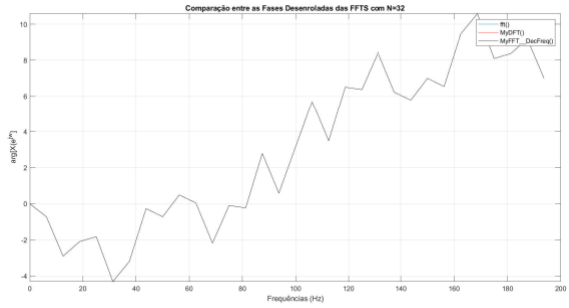

Figura 17: Comparação entre as fases desenroladas em um mesmo gráfico

Na Figura 17 todos os espectros de fase desenrolada são plotados no mesmo gráfico com legendas. Como o erro é muito baixo os gráficos estão sobrepostos. Nas Figuras 18 e 19

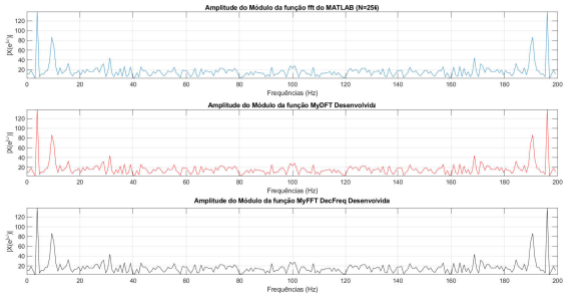

Figura 18: Comparação entre as Magnitudes das diferentes implementações da Transformada Rápida de Fourier para N=256 pontos

são mostrados os espectros de magnitude das diferentes implementações da Transformada Rápida de Fourier para um sinal com 256 pontos. Na Figura 18 os resultados são plotados separadamente assim como em todos os outros gráficos apresentados. Na Figura 19 os espectros de magnitude são plotados juntos afim de comparação entre os mesmos.

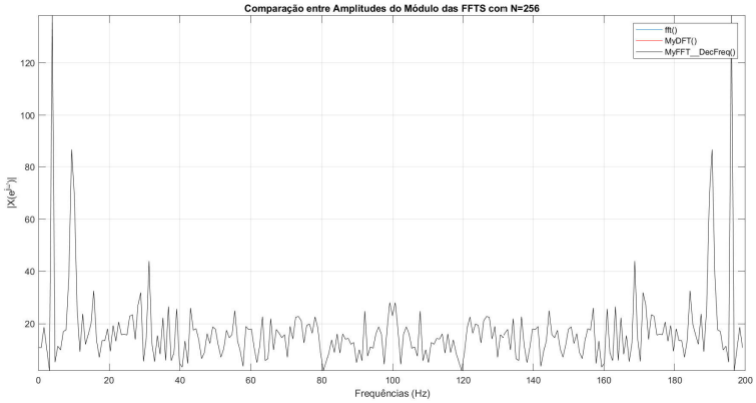

Figura 19: Comparação entre as Magnitudes das diferentes implementações em um mesmo gráfico

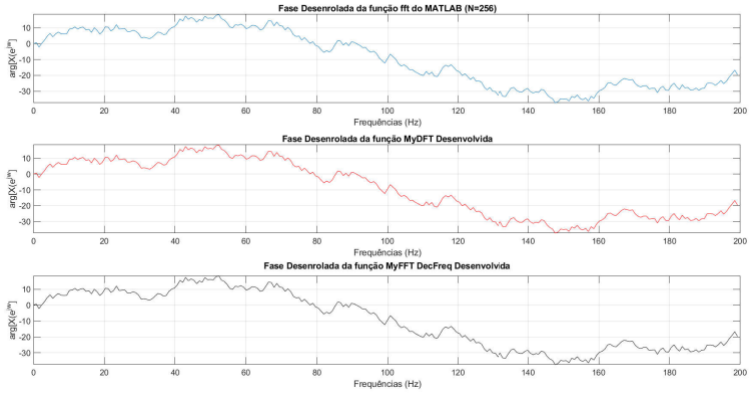

Figura 20: Comparação entre as Fases Desenroladas das diferentes implementações da Transformada Rápida de Fourier para N=256 pontos

As Figuras 20 e 21 apresentam a comparação entre as fases desenroladas nas diferentes implementações. No gráfico 20 estão plotadas as fases separadas e na Figura 21 os resultados são plotados em um mesmo gráfico.

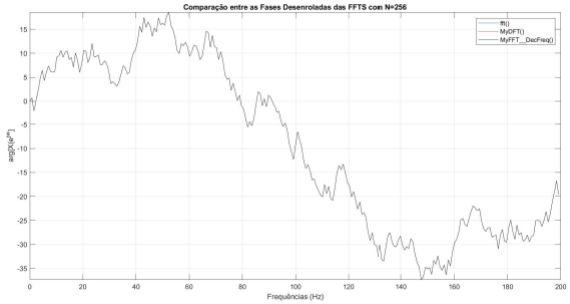

Figura 21: Comparação entre as Fases Desenroladas em um mesmo gráfico

Nas Figuras 22 e 23 são apresentadas as comparações entre as magnitudes de um sinal aleatório com $N = 1024$ pontos e frequência de amostragem $fs = 512Hz$. Na Figura 22 em azul é descrito o espectro proveniente da função fft contida no MATLAB, em vermelho, o resultado da função MyDFT, e em preto temos a função MyFFT-DecFreq.

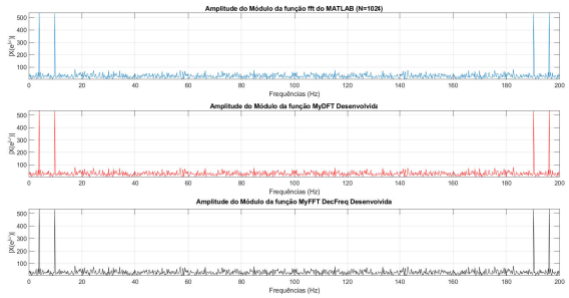

Figura 22: Comparação entre as magnitudes

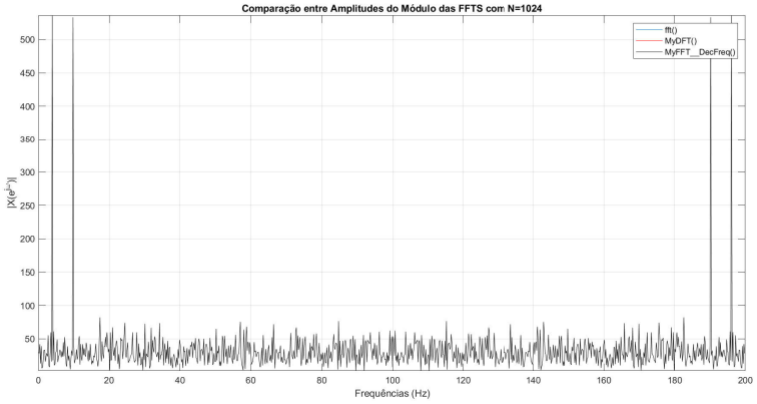

Figura 23: Comparação entre as magnitudes em um mesmo gráfico

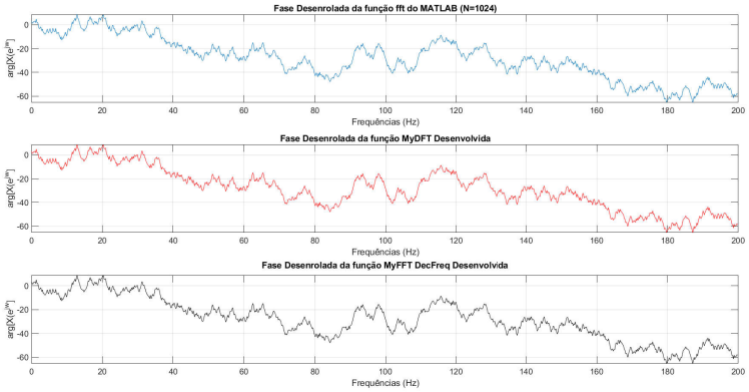

Figura 24: Comparação entre as fases desenroladas N=1024

Na Figura 23 vemos a comparação entre as magnitudes dos diferentes algoritmos implementados. Como o erro é quase nulo, um gráfico sobrepõe o outro.
Por último o sinal com $N = 2048$ pontos. Assim como em todos os gráficos construídos na Figura 25 são apresentadas as magnitudes provenientes das diferentes implementações, cada uma em sua respectiva cor. Na Figura 26 as mesmas magnitudes são plotadas no mesmo gráfico para análise em conjunto.

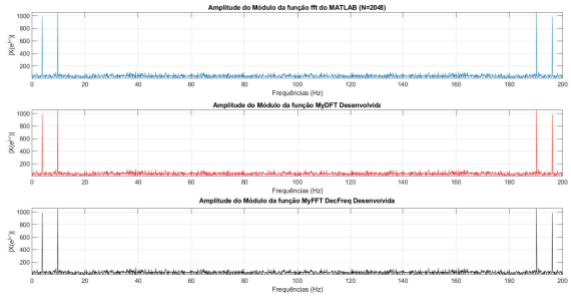

Figura 25: Comparação entre as magnitudes

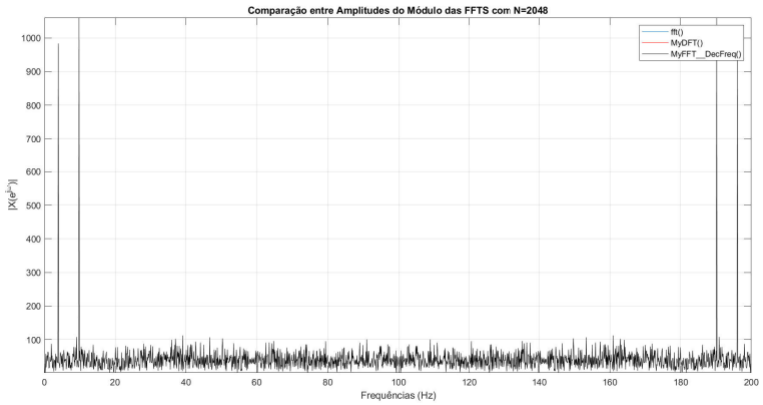

Figura 26: Comparação entre as magnitudes em um mesmo gráfico

E vemos por último a comparação entre as fases desenroladas.Nas Figuras 27 e 28 são evidenciadas as comparações em fase entre as implementações. Mais uma vez, a Figura 27 apresenta as fases de cada implementação separadamente e a Figura 28 apresenta uma gráfico com as fases sobrepostas devido ao erro ser muito baixo.

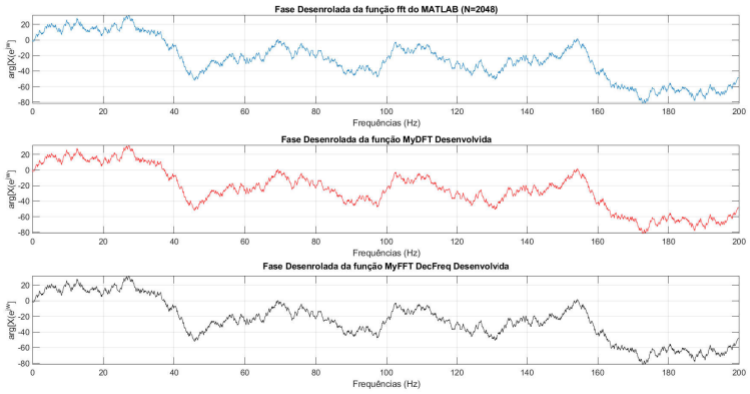

Figura 27: Comparação entre as fases

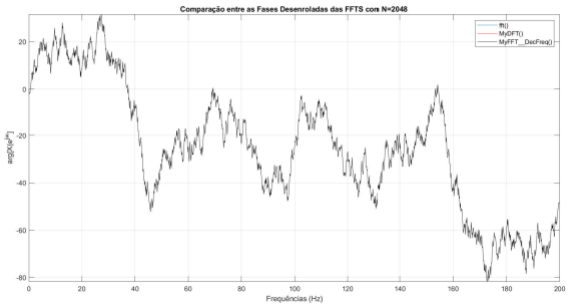

Figura 28: Comparação entre as fases em uma mesmo gráfico

Todos os resultados obtidos como tempo computacional e Erro quadrático médio estão ilustrados na Tabela 3 a seguir. O erro quadrático médio foi calculado também será apresentado na Tabela 3:

N	Tempo FFT	Tempo MyDFT	Tempo MyFFT_DecFreq	Erro FFT vs <>MyDFT	Erro vs <>MyFFT-DecFreq height2
0.0011479	0.0033963	0.037946	7.3775e-33	0	
4	$9.96e^{-05}$	0.0026317	0.0056187	$7.5255e^{-32}$	0
8	$7.28e^{-05}$	0.00126	0.014236	$1.3896e^{-29}$	$2.7733e^{-32}$
16	0.0001025	0.0002602	0.0050875	$1.2409e^{-17}$	$5.6699e^{-31}$
32	0.0004495	0.0019094	0.020939	$2.45e^{-36}$	$4.1739e^{-30}$
64	$3.05e^{-15}$	0.0026956	0.0008042	$4.4243e^{-25}$	$4.4688e^{-29}$
128	$2.22e^{-15}$	0.0050407	0.0022388	$1.2051e^{-23}$	$6.2223e^{-28}$
256	$2.38e^{-05}$	0.028753	0.0033835	$3.9726e^{-22}$	$6.4765e^{-27}$
512	$2.27e^{-05}$	0.1211	0.0047285	$1.2637e^{-20}$	$6.3515e^{-28}$
1024	$2.98e^{05}$	0.37058	0.0082384	$3.73e^{-19}$	$5.8939e^{-26}$
2048	$4.78e^{-05}$	1.4146	0.017403	$1.1499e^{-17}$	$5.5995e^{-24}$

A partir dos tempos computacionais da Tabela 3 é possível a construção do gráfico do tempo computacional em função do numero de amostras N.
A Figura 29 apresenta os tempos computacionais das funções implementadas,sendo em vermelho a função MyDFT, em azul a função fft do MATLAB, em preto a função MyFFT-DecFreq.

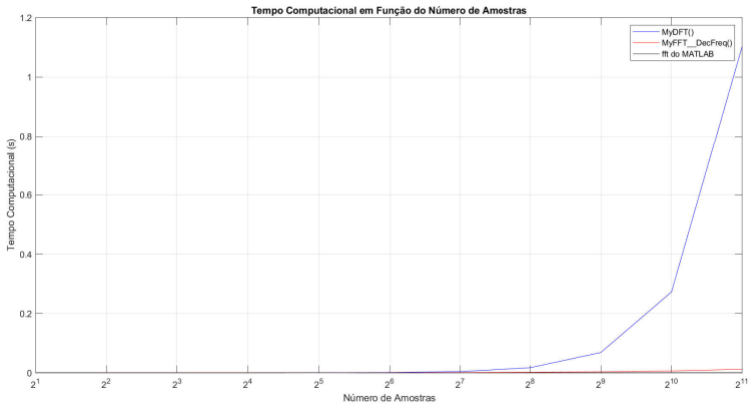

Figura 29: Tempo Computacional em função de N Amostras

4 Análise dos Resultados

4.1 Cenário 1

No primeiro cenário de testes serão analisadas as implementações realizadas em comparação com a função fft já existente no MATLAB. Como o sinal é composto por um sinal aleatório, utilizamos a função *randn*, com 2000 pontos somado a três senoides ($15Hz$, $45Hz$ e $70Hz$) é esperado que o espectro de magnitude do mesmo apresente componentes em todas as frequências (devido a função randn) e picos nas frequências das senoides já mencionadas.
Ao analisar a Figura 8 é possível observar que a magnitude apresenta componentes de pico nas frequências das senoides ($15Hz$, $45Hz$ e $70Hz$) assim como o esperado e que o eixo das abscissas é representado pela frequência de amostragem adotada. Através desta figura fica evidente que em todas as implementações realizadas, assim como o resultado proveniente da função fft do MATLAB são coerentes,ou seja, os resultados são iguais. Para comprovar tal afirmação todas as magnitudes foram analisadas em um mesmo gráfico com legendas nas cores de cada sinal (Figura 9). Como pode ser visto neste gráfico o resultado é sobreposto devido ao pequeno erro entre os mesmos.
Os erros por amostra entre as funções implementadas e a fft do MATLAB foram apresentados nas Figuras 12 e 13. Todos os módulos dos erros por amostra estão em casas de 10^-10 sugerindo um erro muito pequeno entre os resultados analisados e comprovando a sobreposição dos sinais na Figura 13.
Tanto a resposta em magnitude quanto a resposta em fase se comportaram da mesma maneira. Ao analisar a Figura 10 é possível perceber que a resposta da fase desenrolada para todas as implementações se assemelha bastante. O mesmo pode ser constatado na

Figura 11 onde ocorre a sobreposição dos sinais devido ao erro muito pequeno entre as diferentes respostas.
Já constatada a eficácia em termos de resultados esperados é importante analisar o tempo computacional referente as implementações realizadas. Para isto foi construída a Tabela 1 onde estão descritos todos os tempos computacionais para um sinal de 2000 amostras. Ao analisar esta primeira tabela fica constatado que a função nativa do MATLAB, fft, obteve o menor tempo computacional com $0.0073s$. A função com dizimação na frequência se apresentaram satisfatórias com tempo de $0.0420s$.A função MyDFT responsável por realizar os cálculos da transformada de Fourier da forma direta obteve o pior tempo entre as implementações com $1.1571s$. Este resultado era esperado uma vez que o mecanismo de cálculo da função MyDFT apresentado anteriormente é menos eficaz e exige um maior custo computacional em relação a função com dizimação.
A Tabela 2 é contida pelos erros quadráticos médios entre as funções utilizadas. Ao observar esta Tabela fica constatado que a função MyDFT comparada a qualquer outra função apresenta um erro maior na casa de 10^{-19}. A função de dizimação na frequência apresentam erro quadrático médio na casa de 10^{-25} em comparação com a função fft tornando-as mais eficientes tanto em tempo computacional quanto em erro quadrático médio em relação a MyDFT.

4.2 Cenário 2

O segundo cenário de análise é composto por sinais aleatórios de N pontos somados a duas senoides de 10Hz e 25Hz com uma frequência de amostragem de 512Hz. Os espectros relevantes analisados são referentes a sinais aleatórios com $N = 32$, 256, 1024 e 2048 pontos.
As análises do cenário 01 serão válidas nesta seção pois confirmaram a eficácia das funções implementadas tanto em relação a magnitude quanto fase nos espectros analisados.Logo, pressupondo que as funções implementadas são eficientes em termos de resultados os sinais para N pontos poderão ser analisados.
O primeiro sinal analisado é um sinal aleatório de $N = 32$ pontos somado a duas senoides com frequência de amostragem de $512Hz$. A Figura 14 mostra o espectro de magnitude deste sinal a partir das funções implementadas. Ao observar esta Figura fica evidenciado que os sinais aparentemente tem a mesma forma o que pode ser comprovado na Figura 15 onde há sobreposição dos mesmos devido ao erro ser baixíssimo. Ao analisar tanto as Figuras 14 e 15 não é possível a determinação dos picos de magnitude nas frequências das senoides (10 e 25Hz) devido ao baixo número de pontos em relação a frequência de amostragem.
As Figuras 16 e 17 mostram os espectros de fase desenrolada deste sinal. Assim como nas análises de magnitude a Figura 16 apresenta as fases desenroladas provenientes das diferentes implementações e a Figura 17 apresenta uma comparação entre todas as fases. Ao observar a Figura 17 fica evidenciada a sobreposição das fases desenroladas o que pode ser caracterizado como um erro mínimo entre as implementações. A fase desenrolada analisada têm características contínuas em todo o espectro, não havendo descontinuidades em nenhum ponto.
O segundo sinal analisado é um sinal aleatório com $N = 256$ pontos somado as duas senoides com frequência de amostragem de 512Hz. É esperado que no espectro de magnitude ocorra picos nas frequências das senoides em 10 e 25Hz. As Figuras 18 e 19 mostram a magnitude do sinal analisado. A Figura 18 ilustra a resposta em magnitude proveniente

das diferentes implementações e a Figura 19 é uma comparação entre as implementações na qual ocorre sobreposição dos sinais devido ao erro mínimo. Esta análise em relação ao sinal de 32 pontos apresenta picos pouco definidos nas frequências desejadas.
Nas Figuras 20 e 21 são ilustradas as respostas em fase desenrolada deste sinal de 256 pontos. A fase toma uma forma diferente e mais bem definida em relação ao sinal de 32 pontos apresentado anteriormente.É válido salientar que não há descontinuidades na fase analisada deste sinal.
O terceiro sinal relevante para análise consiste em um sinal aleatório com $N = 1024$ pontos somado as duas senoides e uma frequência de amostragem de 512Hz. Ao analisar as Figuras 22 e 23 fica evidenciado as componentes espectrais bem definidas nas frequências desejadas.
A Figura 23 ilustra o espectro de magnitude do sinal aleatório realizado pelas diferentes funções. Ao analisar a Figura 23 é possível afirmar que a magnitude está coerente com o resultado esperado e que o erro é muito baixo sendo os espectros sobrepostos conforme é observado.
Os espectros de fase desenrolada para 1024 pontos são ilustrados nas Figuras 24 e 25.Quanto maior o número de pontos mais bem definida é o espectro de fase comparado com sinais com menos número de pontos. A fase deste sinal assim como a magnitude têm seus valores coerentes e comprovam a eficácia dos resultados provenientes das implementações realizadas.
O último sinal analisado é composto por 2048 pontos aleatórios somado novamente as duas senoides de frequências 10 e 25Hz. Os espectros de magnitude contidos nas Figuras 26 e 27 comprovam os resultados esperados uma vez que há componentes espectrais em todas as frequências (devido a função randn) com picos nas frequências das senoides 10 e 25Hz. Assim como em todas as análises feitas ao longo do trabalho, a Figura 28 mostra os resultados provenientes das diferentes implementações. Visualmente os resultados parecem congruentes porém é preciso analisar a Figura 27 para se ter certeza que os resultados são realmente iguais em magnitude.
A principal parte da análise deste cenário está contido na Figura 29 aliados a Tabela 3. A partir dela é possível a determinação dos erros quadráticos médios entre as funções implementadas e a função já existente no MATLAB (fft) além do tempo computacional das funções em relação ao número de amostras.Observando a Tabela 3 vemos o tempo computacional e os erros são coerentes com o cenário, evidenciando o sucesso da implementação dos algoritmos.

5 Conclusão

O cálculo da Transformada de Fourier é um operação muito importante na área de processamento de sinais pois através do mesmo torna-se possível uma análise detalhada das componentes espectrais do sinal, obtendo características importantes não detectáveis em uma análise temporal.
Atualmente a busca pela excelência faz com que estas operações sejam feitas no menor tempo possível. Sendo assim, surge a necessidade de reduzir o tempo computacional afim de obter um resultado fiel mais rapidamente.
Neste trabalho prático foram implementadas três funções responsáveis por realizar o cálculo da Transformada Rápida de Fourier e a partir dos resultados (magnitude e fase) e do tempo computacional foi possível verificar a eficácia das mesmas. Ambas funções implementadas MyDFT e MyFFT-DecFreq se mostraram eficientes em termos de resultados esperados tanto no primeiro quanto no segundo cenário de testes. Ao analisar o erro quadrático

médio entre as implementações verificou-se um erro extremamente baixo comprovando a eficácia dos resultados.
A principal diferença entre as funções desenvolvidas quando comparadas a função nativa do MATLAB (fft) foi o tempo de execução das operações. Através das análises feitas fica evidente que quanto maior o número de operações realizadas pela implementação maior o tempo computacional da mesma. A função MyDFT obteve o pior desempenho computacional visto que é a implementação na qual são realizados o maior número de operações. A função e dizimação na frequência MyFFT-DecFreq apresenta um tempo computacional menor visto que o número de operações nestas implementações é ligeiramente menor quando comparado à forma direta. Em relação à função nativa fft do MATLAB, a mesma obteve o melhor desempenho computacional uma vez que o tempo de processamento se manteve muito abaixo as outras implementações.
Portanto, vale ressaltar a importância da implementação deste trabalho. Ao realizar a implementação das Transformadas de Fourier via MATLAB é possibilitado o entendimento tanto da forma direta quanto dos diagramas de borboleta além da comprovação prática que a redução do número de cálculos acarreta em um melhor desempenho computacional.

6 Referências

[1] OPPENHEIM, A. V. Discrete-time signal processing. [S.l.]: Pearson Education India,1999.