Implementação e Análise de Resultados para Transformarda Discreta de Fourier

Carlos Luiz, Gabriel Antonio P. S. Carneiro

I. Introdução

A transformada de Fourier discreta (TFD) tem um papel importante na análise, projeto e implementação de algoritmos para sistemas de processamento de tempo discreto de sinais. É importante que existam algoritmos eficientes para o cálculo explicito da TFD. A TFD é um componente importante em muitas aplicações práticas dos sistemas de tempo discreto.

Neste relatório, discutiremos alguns métodos para o cálculo de valores da TFD. O foco principal deste relatório é uma classe de algoritmos particularmente eficiente para a implementação computacional da TFD de N pontos (N = 32, 64, 128 e 256). Esses algoritmos são chamados de algoritmos FFT (do inglês Fast Fourier Transform) . Para obtermos uma máxima eficiência, os algoritmos da FFT devem calcular todos os valores da TFD.

Nas próximas seções, consideraremos dois algoritmos para calcular a transformada de Fourier discreta. Começamos pela Seção II com discussões sobre os métodos de cálculo direto, ou seja, métodos que são baseados diretamente com a definição da TFD como uma fórmula de cálculo. A Seção II.(A) descreve a implementação do algoritmo da Transformada de Fourier de Tempo Discreto (TFTD ou DTFT sigla em inglês(discrete-time Fourier transform)), o entendimento da transformada descrita matematicamente, já na Seção II.(B) discutimos o algoritmo da Transformada Rápida de Fourier - Dizimação no Tempo sua representação matemática para a implementação do algoritmo. todos os algoritmos citados neste relatório necessários para o cálculo da TFD foram implementados no software de simulação matemática MatLAB. Na Seção III são apresentados e discutidos os resultados obtidos para os valores de N calculados pelos algoritmos implementados. Após a discrição teórica e matemática para um amplo entendimento e facilitar a implementação dos algoritmos apresentando e discutido os seus resultados descrevemos a Seção IV para a conclusão deste relatório.

II. Metodologia

Nesta Seção apresentaremos a nossa metodologia para implementação dos algoritmos da Transformada de Fourier de Tempo Discreto e Transformada Rápida de Fourier.

A. Transformada de Fourier de Tempo Discreto

Para construção da função da Transformada de Fourier de Tempo Discreto o primeiro passo foi entender como a transformada funcionava e como poderíamos representá-la programaticamente.

Tendo em mente que a transformada é para o sinal $x[n]$ é dada por $X[k] = \sum_{n=0}^{N-1} x[n]W_N^{kn}$, com $k = 1, 2, 3, ...N-1$, onde $W_N^{kn} = e^{-j(2*\pi/N)}$, percebemos que programaticamente a série poderia ser calculada com dois estruturas de repetições aninhadas, de amneira que o algoritmo resultante teria complexidade $O(n^2)$.

Visando organizar o código, optamos por implementar uma função que receberia um vetor que representa o tempo, um vetor com as N amostras e a frequência de amostragem. O tempo e a frequência de amostragem foram utilizados para plotar o sinal original e o espectro de frequência normalizadamente.

Os código implementado pode ser visualizados num repositório do GitHub[1].

B. Transformada Rápida de Fourier - Dizimação no Tempo

Inicialmente é essencial saber que consideramos todas os aspectos de simetria e periodicidade para W_N^{kn} para nossa implementação. Além do mais, utilizamos a notação de ordem de bits reversa para ordenação das amostras, que implica numa simplificação da função borboleta, resultando no cálculo representado pelo diagrama de fluxo exibido na Fig. 1.

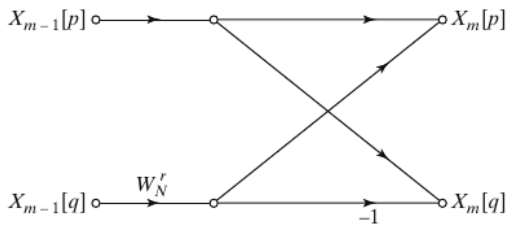

Figura 1. Diagrama da função borboleta utilizado. Fonte: Oppenheim and Schafer [1]

As equações para o cálculo são:

$$X_m[p] = X_{m-1}[p] + W_N^r X_{m-1}[q] \tag{1}$$

$$X_m[q] = X_{m-1}[p] - W_N^r X_{m-1}[q] \tag{2}$$

Nessas equações os valores de p e q variam para cada estágio, como pode ser visto no diagrama de fluxo para um exemplo com $N = 8$ exibido na Fig. 2.

Programaticamente, implementamos uma função que recebe as amostras, um vetor de representação do tempo e a frequência de amostragem. A frequência de amostragem e o vetor de tempo são utilizados na plotagem do sinal e do espectro na frequência.

[1] https://github.com/gabri14el/fft_matalab_pds

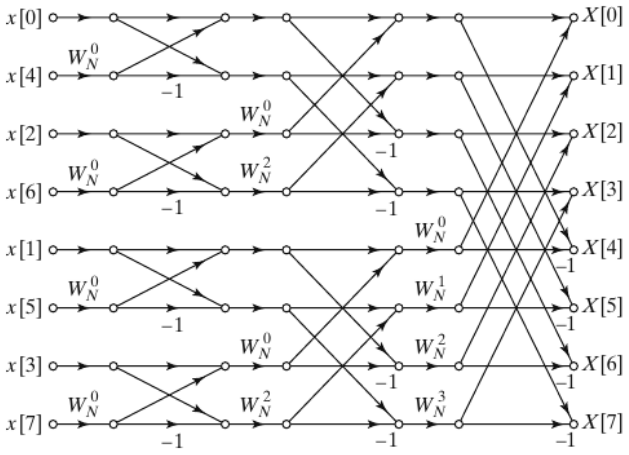

Figura 2. Diagrama de fluxo para exemplo com 8 amostras. Fonte: Oppenheim and Schafer [1]

Como qualquer implementação Radix2 da Transformada de Rápida de Fourier, nosso algoritmo funciona para $N = 2^v$, sendo v a quantidade de estágios que serão necessários para o cálculo. No código, podemos obter a quantidade de amostras N e a partir dela calculamos v utilizando logaritmo na base 2.

A representação em ordem reversa de bits é obtida através da função $bitrevorder$ do MatLAB.

Utilizamos uma variável auxiliar para o calculo do das posições p e q, ela varia de 1 até 2^v em potências de 2, ou seja $1, 2, 4..., 2^v$, aumentando o pulo de acesso ao vetor. Por exemplo, ao observarmos a Fig. 2, com a ordem inversa de bits, no primeiro estágio vetores com diferença de posição 1 são acessadas, no segundo a diferença entre as posições são de 2 posições e no último elas são de 4 posições.

Para o cálculo utilizamos três estruturas de repetição aninhadas, uma que varia de 1 até a v, representado o estágio, outra que varia 0 até $N-1$, sendo que os passos para o incremento da variável é definido pelo nível que estamos calculando no momento, representando as borboletas que devem ser implementadas para cada estágio e, por último, uma repetição com uma variável que varia de 0 até o valor da nossa variável auxiliar, que é utilizada no cálculo individual de cada estrutura borboleta.

Comparando com a Fig. 2, na primeira iteração da repetição que representa o estágio, seria calculado os valores da transformada para x[0] e x[4], ..., x[3] e x[7], na segunda iteração os valores calculados seriam somados aos novos cálculos de x[0] e x[2], ..., x[5] e x[7], no último nível, os valores seriam somados à transformada utilizando x[0] e x[1], ..., x[6] e x[7]. Os valores que estamos vendo dentro dos colchetes, representam os valores de p e q para as Equações 1 e 2.

Note que os valores originais são modificados, adicionando os novos valores calculados a cada iteração das repetições, como os valores anteriores são necessários para os cálculos, utilizamos duas variáveis como *buffers* antes de atualizar os valores para o vetor que representa a transformada durante os cálculos.

No final do algoritmo, como foi utilizada a ordem binária reversa, a transformada já está representada corretamente, assim como pode ser visto no final do diagrama exibido na Fig. 2 e, portanto, o vetor que estava sendo somados os valores é retornado.

Por último, o espectro de frequência é plotado, para isso utilizamos a função $stem$ em conjunto com as funções abs e $angle$, para plotarem a magnitude e fase do espectro, respectivamente. Na plotagem são utilizadas também o vetor de que representa o tempo e a frequência de amostragem.

O código fonte pode ser encontrado no mesmo repositório apresentado na Subseção II-A e foi baseado na implementação de Jacob [2].

C. Transformada Rápida de Fourier - Dizimação na Frequência

A implementação para o algortimo de dizimação na frequência tem muitas similaridades ao de dizimação no tempo. Na Fig. 3 podemos observar o diagrama de fluxo geral para um exemplo de oito amostras seguido pelo algoritmo.

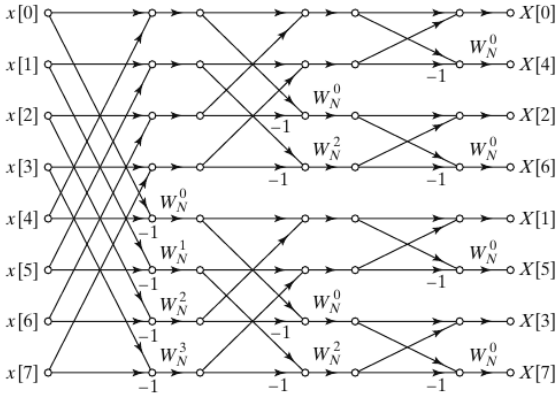

Figura 3. Diagrama de fluxo para exemplo com 8 amostras. Fonte: Oppenheim and Schafer [1]

Note que enquanto na dizimação do tempo a ordem de bits reversa era utilizada para rearranjar as amostras antes do cálculo da transformada, nesse caso, a ordem padrão das amostras gera a transformada em ordem de bits reversa e as conexões das amostras entre o fluxo vão das maiores para a menor.

Matematicamente a operação matemática para dizimação no tempo, a partir da aplicação de propriedades de periodicidade e simetria, tem forma de:

$$X_m[p] = X_{m-1}[p] + X_{m-1}[q] \tag{3}$$

$$X_m[q] = (X_{m-1}[p] - X_{m-1}[q])W_N^r \tag{4}$$

Programaticamente, assim como na dizimação no tempo, o algoritmo é composto por 3 repetições aninhadas, onde a primeira repetição representa o estágio que estamos calculando, a segunda a quantidade de operações borboletas que devemos fazer e a terceira o cálculo das operações borboletas.

Como o o fluxo é diferente da dizimação no tempo, os valores para a variável de controle para a segunda repetição muda. A quantidade de funções borboletas continua variando de 0 até $N-1$, porém com um passo de $N/2^{v-1}$.

A variável auxiliar, para oi cálculo das posições p e q, começa em $N/2$ e vai sendo dividida por 2 quando o cômputo de um estágio é finalizado. Ou seja, no exemplo mostrado na Fig. 3, no primeiro estágio, a diferença de posição entre as posições acessadas pelo cálculo da borboleta é de 4 posições, depois duas posições e por último, uma posição.

Ao final das repetições temos a transformada rápida de Fourier mas em ordem de bits reversa, então aplicamos a função de ordem de bits reverse, $bitrevorder$, e obtemos a transformada na ordem correta, retornando-a.

Os argumentos e a plotagem do espectro funcionam da mesma forma que na dizimação no tempo. O código foi baseado na implementação de Jacob [2] e está disponível no mesmo repositório apresentado na Seção II-A.

III. Resultados e Discussões

Nesta seção apresentaremos o espectro na frequência utilizando as TDFs implementadas para seis casos. A execução dos casos foi baseado na utilização da função senoidal $x(t) = \sin 2\pi * 60t + \sin 2\pi * 200t$, amostrado com um período de amostragem de $T = 1/500$s.

Os parâmetros de execução dos casos podem ser observados na Tabela I. O *Número de Amostras* representa a quantidade de amostras da função utilizadas no calculo da TDF, *Zeros Adicionados* denota a quantidade de amostras com valor zero que foram adicionados depois das amostras do sinal para o cálculo da TDF, enquanto o *Total* exibe o tamanho total do sinal utilizado para o cálculo.

Caso	Número de Amostras	Zeros Adicionados	Total
1	32	0	32
2	32	32	64
3	64	0	64
4	64	64	128
5	256	0	256
6	256	256	512

Tabela I
Parâmetros de execução para cada um dos casos

Como a frequência de amostragem do sinal foi $F_s = 500Hz$, isso implica que poderemos visualizar com a TDF o espectro para até $F = 250Hz$, após isso os valores serão replicados de maneira espelhada. É importante ressaltar que a função senóide $x(t)$ é composta pela soma de duas senóides, uma com $F = 60Hz$ e outra com $F = 200Hz$, estando dentro do limite de visualização possível com a frequência de amostragem utilizada.

Para todos os algoritmos conseguimos o mesmo resultado para todos os casos de teste, logo, nesse documento, apresentaremos apenas um gráfico resultante para cada um dos casos.

A. Casos 1 e 2

O resultados obtido para os Casos 1 e 2 podem ser observados nas Fig. 4 e 5, respectivamente.

Como $250Hz$ representa nossa frequência máxima de representação, a partir desse valor visualizamos um espelhamento do espectro. Note nas Fig. 4 e 5, no módulo do espectro na frequência, que podemos observar um pico em aproximadamente $60Hz$ e outro em aproximadamente $200Hz$, visualizando as parcelas individuais de cada uma das senóides formadoras de $x(t)$.

Se compararmos a Fig. 4 com a Fig. 5, notamos três picos ao invés de um, indicando que os zeros adicionados à amostra cria esse aumento nessas frequências que antes não existiam, porém os picos são próximos dos valores das frequências das senoides que compõem $x(t)$. Isso significa que o espectro está mais denso, mas ainda representa o sinal $x(t)$.

B. Casos 3 e 4

O resultados obtido para os Casos 3 e 4 podem ser observados nas Fig. 6 e 7, respectivamente.

Em ambos os casos podemos visualizar as duas componentes senoidais de $x(t)$. Assim como no Caso 3, há um aumento no número de amostras próximas às frequências das componentes de $x(t)$ quando adicionamos zero às amostras.

Se compararmos o Caso 3 com o Caso 2 podemos notar que no Caso 3 a amplitude nas frequências componentes é maior e que, na fase, no Caso 2 há uma negativação da amplitude em alguns componentes.

C. Casos 5 e 6

O resultados obtido para os Casos 5 e 6 podem ser observados nas Fig. 8 e 9, respectivamente.

Novamente podemos visualizar os componentes das duas senoides de $x(t)$. Fica mais evidente a negação de alguns componentes da fase quando adicionamos zeros à amostras quando observamos as Fig. 8 e 9.

É importante notar que quando adicionamos o zero as frequências próximas às frequências dos componentes senoidais de $x(t)$ aumentam bastante, note que na Fig. 8, no módulo do espectro, na região próxima aos 60 há três componentes, além do 60Hz, que se destacam no espectro, esse número é aumentado para seis na Fig. 9.

Apesar da adição de zeros não prejudicar a visualização do módulo dos componentes de frequência no espectro na frequência, fica claro que a fase é afetada, após observarmos todos os casos.

IV. Conclusão

Apresentamos nesse documento nossos métodos para o desenvolvimento de funções para cálculo da Transfromada Discreta de Fourier e Transformada Rápida de Fourier utilizando o MatLAB. Em nossos testes, todos os três algoritmos alcançaram o mesmo resultado.

Analisamos também o espectro de frequência para uma função composta pela soma de senoides, em seis casos distintos, verificando o impacto da quantidade de amostras que utilizamos nos cálculos e também a adição de zeros ao final das amostras.

Testamos a utilização de 32, 64 e 256 amostras para cálculo da Transformada Discreta de Fourier, e em todos casos foi

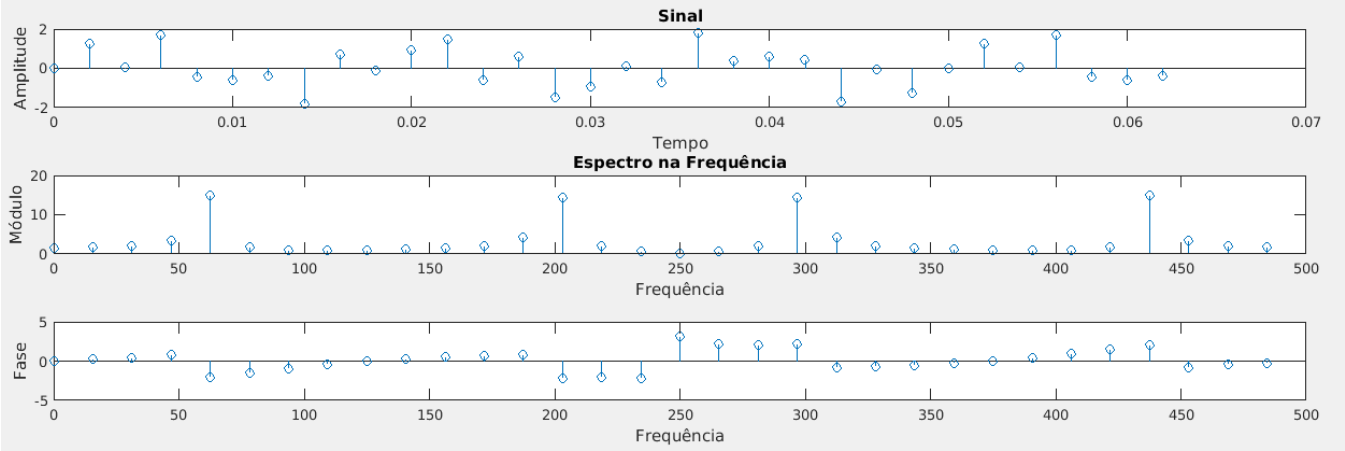

Figura 4. Espectro na Frequência para 32 amostras de $x(t)$

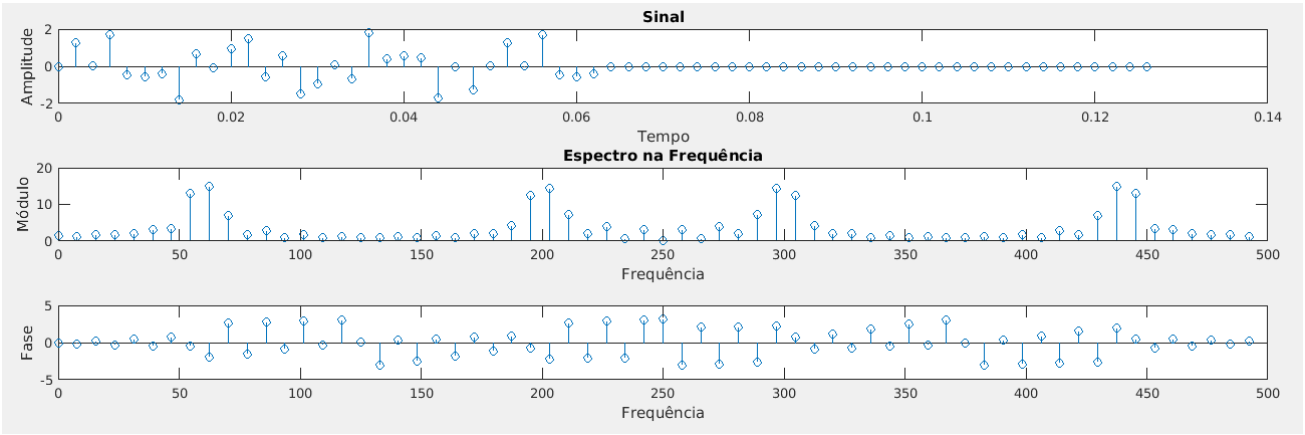

Figura 5. Espectro na Frequência para 32 amostras de $x(t)$ + 32 zeros

possível observar os componentes de frequências das duas senoides formadoras do sinal utilizado.

Concluímos que ao adicionar zeros o módulo do espectro fica mais denso, mas ainda consegue representar corretamente o sinal. Na fase há uma negativação de alguns componentes.

Referências

[1] A. Oppenheim and R. Schafer, *Processamento Em Tempo Discreto De Sinais*. PEARSON BRASIL. [Online]. Available: https://books.google.com.br/books?id=g72vnQEACAAJ

[2] N. A. Jacob, "Radix 2 fast fourier transform decimation in time/frequency," Jun 2013. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/42214-radix-2-fast-fourier-transform-decimation-in-time-frequency

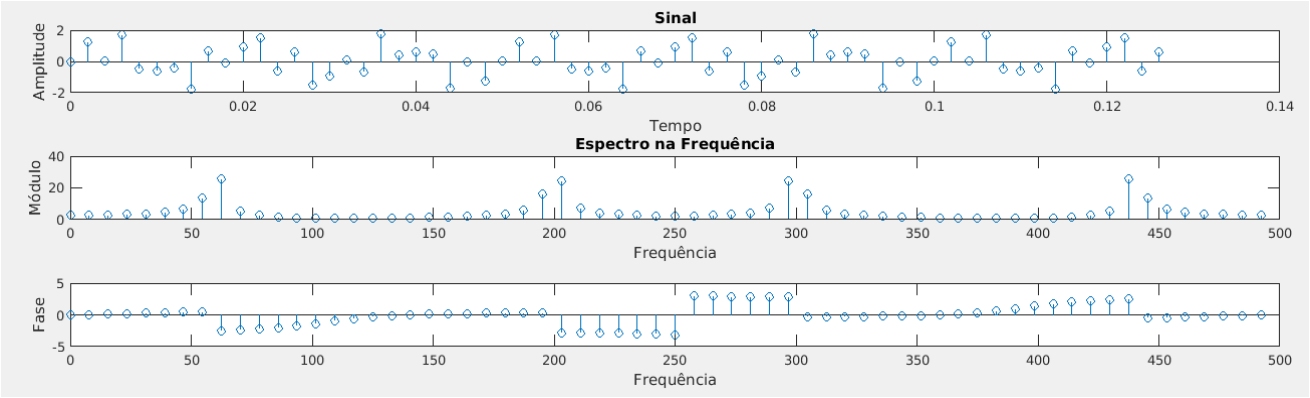

Figura 6. Espectro na Frequência para 64 amostras de $x(t)$

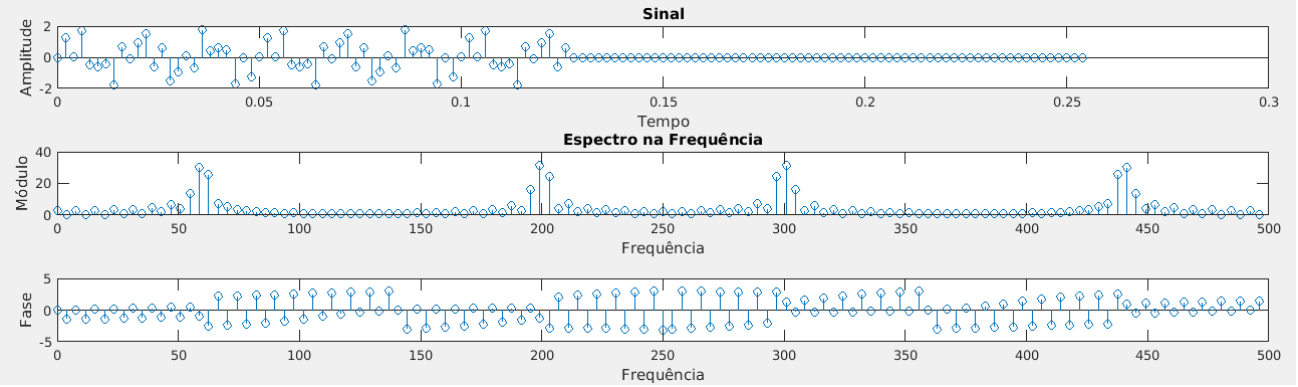

Figura 7. Espectro na Frequência para 64 amostras de $x(t)$ + 64 zeros

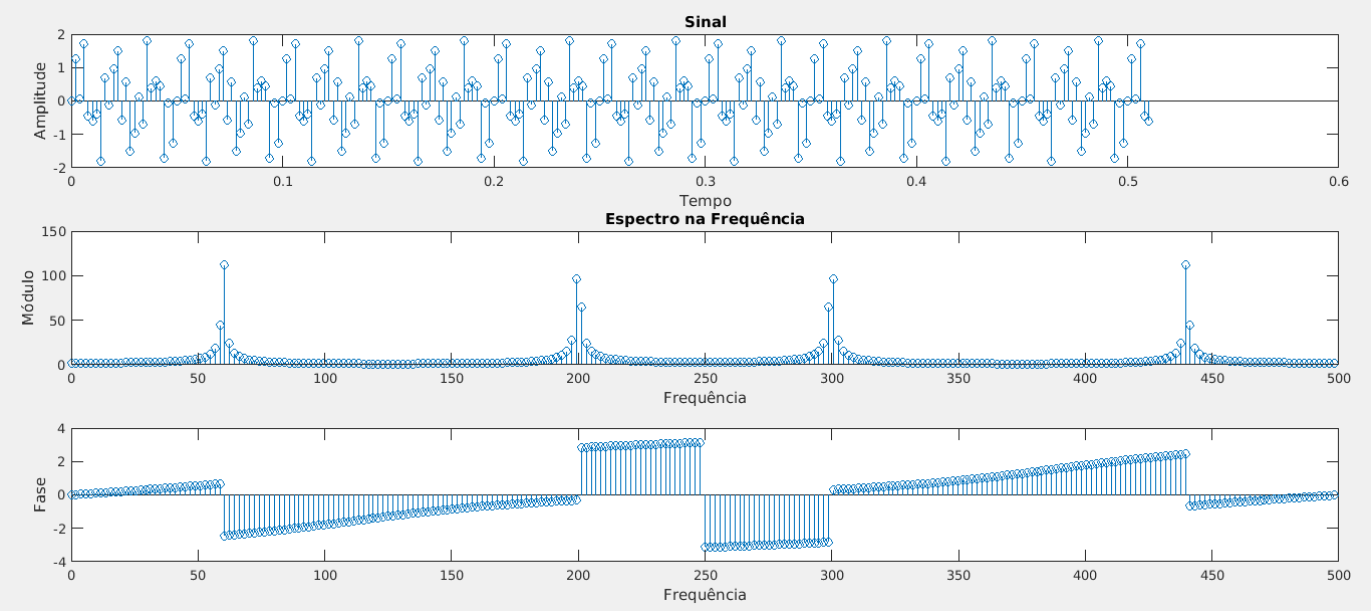

Figura 8. Espectro na Frequência para 256 amostras de $x(t)$

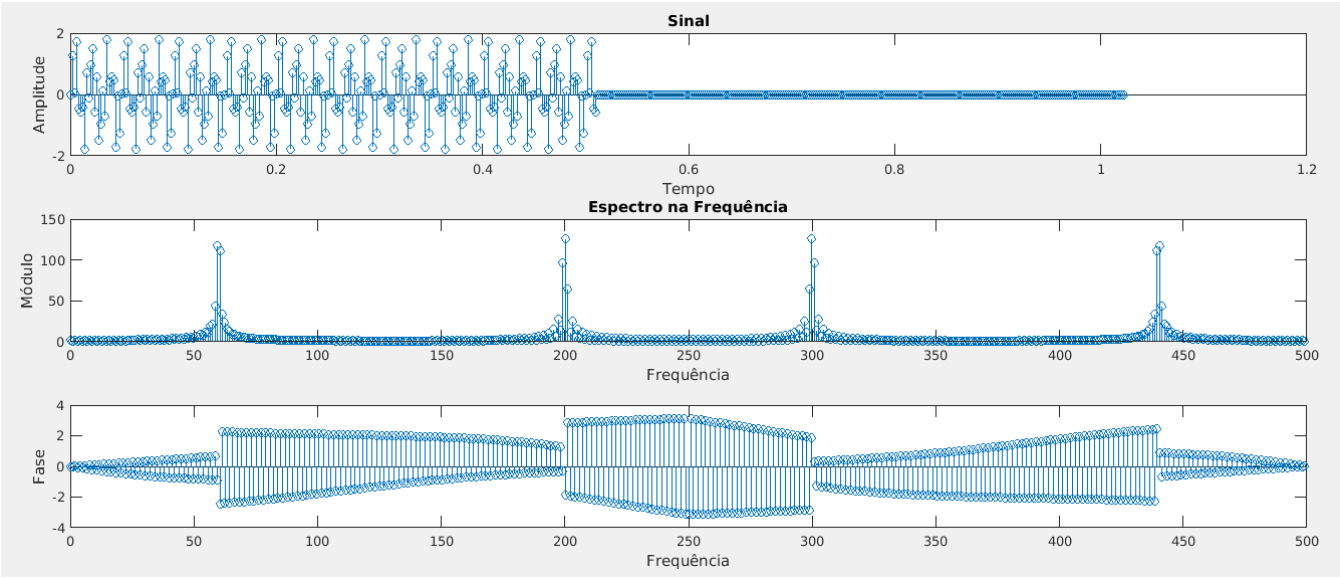

Figura 9. Espectro na Frequência para 256 amostras de $x(t)$ + 256 zeros