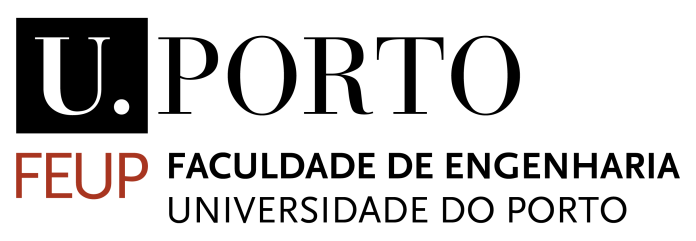
Faculdade de Engenharia da Universidade do Porto



*PDSI – Second Assignment*

Afonso Queirós up201808903

Bruno Baptista up201708981

14/05/2019

# 2a)

Começa-se por testar o algoritmo que nos é dado no ficheiro “fft\_DIT.m”. Este script contém o código que calcula a transformada de Fourier de base 2, com base na divisão do tempo. Se, na entrada, o sinal for uma sinusoide perfeita, o resultado da transformada é o que se apresenta na figura seguinte:

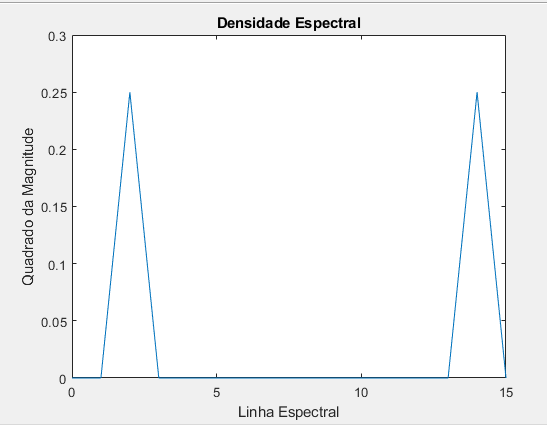
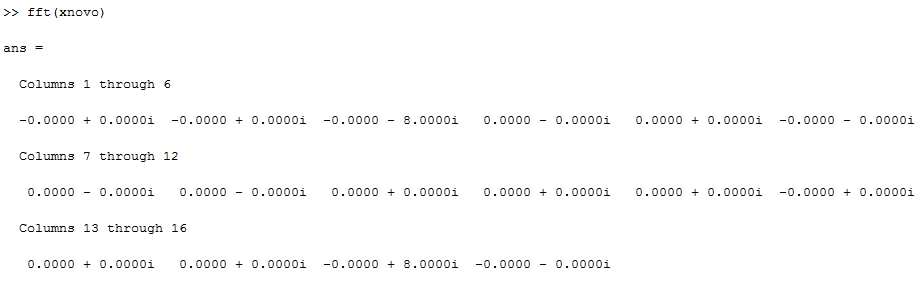
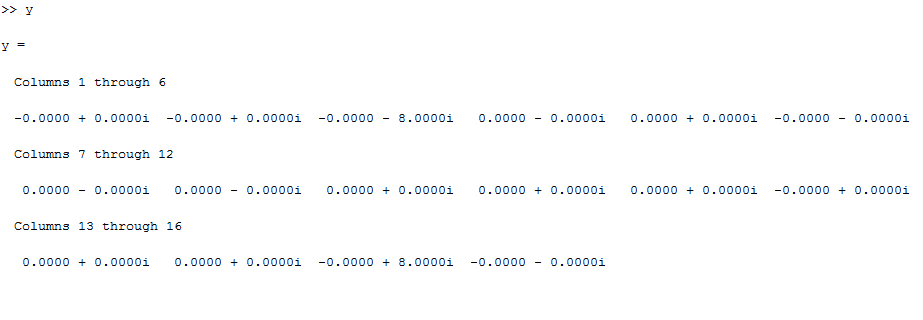


Figura 1 - Gráfico espetral da sinusoide.

Como foi abordado nas aulas da unidade curricular de Processamento Digital de Sinais, a transformada de uma onda sinusoidal é dada por dois diracs localizados na frequência fundamental. Pela figura acima, conseguimos comprovar a ocorrência desse fenómeno. Para confirmar que o algoritmo está correto, é executado no terminal o comando “fft(xnovo)” de modo a comparar os resultados obtidos pelas duas formas distintas.

Outra maneira de testar o algoritmo é aplicar à entrada sinal definido por uma rampa, normalizada entre zero e um.

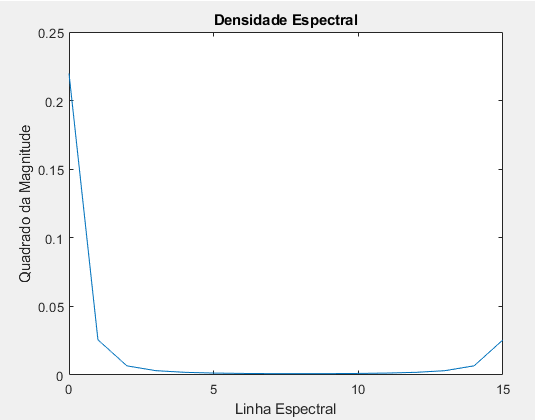
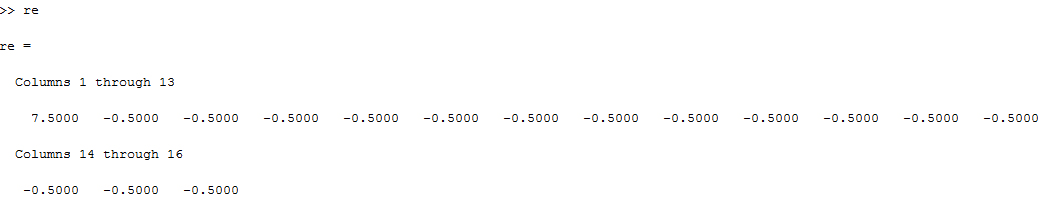


Figura 2 - Espetro da rampa.

Executando o comando “re”, comprova-se que os resultados são os esperados, visto que apenas a primeira componente espetral tem como valor 7.5 e as restantes valem exatamente -0.5.

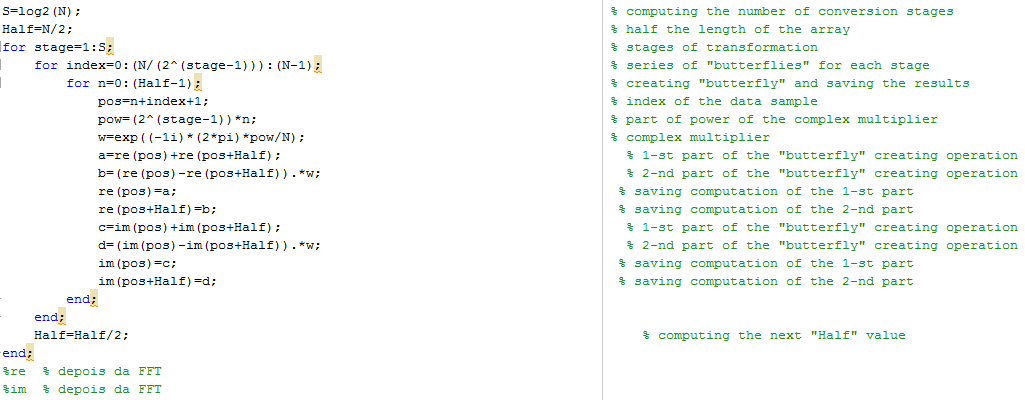


# 2b)

Esta pergunta pede para alterar o algoritmo que foi fornecido, explicado acima, de modo a que uma divisão em frequência irá comprometer a transformação do sinal, ao contrário da pergunta anterior, onde a divisão é segundo o eixo do tempo.

A principal alteração situa-se no conjunto dos ciclos “*for”* entre as linhas 52 e 70. Esta parte do algoritmo permitia concretizar a divisão por tempo, onde, recursivamente, era feita a troca, derivação e soma entre os sinais de entrada e saída. O ciclo de obtenção do sinal transformado ficou da seguinte forma:

Figura 3 - Lógica para cálculo da *FFT*.



A figura acima apresenta o método que foi usado para calcular a *FFT*, onde, em primeiro lugar, se cria a *borboleta* e se guarda os resultados, seguido das operações típicas da transformada. Este processo, como dito anteriormente, é recursivo, para que todo o sinal seja analisado e processado.

Outra mudança importante no *script* foi a mudança da reversão dos bits, que ocorria antes da transformação, para após o cálculo da *FFT*, já que a saída da *DFT* vinha invertida.

Com isto, foi possível chegar á conclusão que este método também e viável, uma vez que os resultados obtidos (gráficos e valores) foram exatamente os mesmos que usando a primeira lógica (*DIT*),

# 2c)

Para substituirmos cada *borboleta,* o cálculo explícito de funções trigonométricas pelo acesso a uma tabela, é necessário que essa tabela tenha uma dimensão de *N/2* por 2, de modo a conter as funções trigonométricas necessárias. Numa da colunas teríamos as funções seno e na outra as funções cosseno de a .

O número de estados é dado por . Portanto, para aceder às diferentes linhas da tabela é criado um ciclo “*for”.* Consoante o estado onde o algoritmo se apresentar (ramo *n* e borboleta *n*), saberemos como calcular a saída. É criada também uma variável auxiliar que é incrementada recursivamente com o ciclo, e, enquanto o valor dela é menor que *n*/2 é guardado noutra variável o valor de , que remete à primeira linha da tabela.

Quando a variável auxiliar chegar ao limite, são sabidas todas as saídas do primeiro estado. Para calcular as saídas dos próximos estados, acede-se recursivamente às próximas linhas.

Executamos esta implementação sucessivamente de modo a acedermos sempre a *n/* linhas da tabela*.* Concluindo, significa que ao longo da posição da linha da tabela que se quer calcular a saída, haverá menos valores a serem calculados.