# notebook3

April 5, 2016

## 1 Sumário



Teste da transformada rápida de Fourier (FFT)

Nessa parte do trabalho iremos testar como o cálculo da FFT irá se comportar com as bibliotecas **Numpy** e **Scipy**. Cooley e Tukey mostraram que é possível dividir a DFT em duas partes similiares. Portanto pela pela definição da DFT teremos:

$$y[k] = \sum_{n=0}^{\text{N-1}} W_n x[n]$$

$$W_n = e^{-2\pi j \frac{km}{N}}$$

$$= \sum_{m=0}^{N/2-1} x_{2m} \cdot e^{-i\ 2\pi\ k\ (2m)\ /\ N} + \sum_{m=0}^{N/2-1} x_{2m+1} \cdot e^{-i\ 2\pi\ k\ (2m+1)\ /\ N}$$

$$= \sum_{m=0}^{N/2-1} x_{2m} \cdot e^{-i\ 2\pi\ k\ m\ /\ (N/2)} + e^{-i\ 2\pi\ k\ /\ N} \sum_{m=0}^{N/2-1} x_{2m+1} \cdot e^{-i\ 2\pi\ k\ m\ /\ (N/2)}$$

Concluindo que dessa forma, podemos obter uma forma de calcular a FFT. Iremos calcular a FFT de um vetor com 4096 valores aleatórios usando as bibliotecas propostas e assim iremos analisar o tempo de compilação de cada.

Resolução da FFT usando a função nativa

In [6]:
```
# Aqui iremos implementar o  o cálcula da FFT usando julia
vetorFftJulia=[1:1:4096];
@time fft(vetorFftJulia);
```

```
0.000165 seconds (66 allocations: 131.141 KB)
```

Resolução da FFT usando myFFT

```
In [7]: # criando a função myIFFT, proposta para o trabalho
        function myifft(x)
            N=4096
            par=Complex32[1:1:N];
            impar=Complex32[1:1:N];
            N=4096
            for i=1:N
                if i%2 == 0
                    par[i]=x[i]
                elseif i%2 == 1
                    impar[i]=x[i]
                end
            end
            N1=N/2
            for i=1:N1
                par[i]=par[i]*exp(-1im*2*pi*i/N1) + exp(-1im*pi/N)
                for j=1:N1
                    impar[j]=par[i]*impar[j]*exp(-1im*2*pi*i/N1)
                end
            end

        end

Out[7]: myfft (generic function with 1 method)

In [8]: N=4096
        vetorFftMy=Complex32[1:1:N];
        @time myfft(vetorFftMy)

0.009181 seconds (3.94 k allocations: 181.248 KB)
```

Teste da transformada rápida de Fourier inversa (IFFT)

Nessa parte do trabalho iremos testar como o cálculo da IFFT irá se comportar com as bibliotecas **Numpy** e **Scipy** Cooley e Tukey mostraram que é possível dividir a IDFT em duas partes similiares. Pela definição da IDFT teremos:

$$y[k] = \frac{1}{N} \times \sum_{n=0}^{\text{N-1}} W_n x[n]$$

$$W_n = e^{2\pi j \frac{km}{N}}$$

$$= \frac{1}{N} \times \sum_{m=0}^{N/2-1} x_{2m} \cdot e^{i\ 2\pi\ k\ (2m)\ /\ N} + \frac{1}{N} \times \sum_{m=0}^{N/2-1} x_{2m+1} \cdot e^{i\ 2\pi\ k\ (2m+1)\ /\ N}$$

$$= \frac{1}{N} \times \sum_{m=0}^{N/2-1} x_{2m} \cdot e^{i\ 2\pi\ k\ m\ /\ (N/2)} + e^{i\ 2\pi\ k\ /\ N} \frac{1}{N} \times \sum_{m=0}^{N/2-1} x_{2m+1} \cdot e^{i\ 2\pi\ k\ m\ /\ (N/2)}$$

Concluindo que dessa forma, podemos obter uma forma de calcular a IFFT. Iremos calcular a IFFT de um vetor com 4096 valores aleatórios usando as bibliotecas propostas e assim iremos analisar o tempo de compilação de cada.

Resolução da IFFT usando a função nativa

```
In [36]: # Aqui iremos implementar o  o cálcula da FFT usando julia
         vetorIfftJulia=[1:1:4096];
         @time ifft(vetorIfftJulia);
```

```
0.000211 seconds (76 allocations: 131.688 KB)
```

Resolução da IFFT usando myIFFT

```
In [ ]: # criando a função myIFFT, proposta para o trabalho
        function myifft(x)
            N=4096
            par=Complex32[1:1:N];
            impar=Complex32[1:1:N];
            N=4096
            for i=1:N
                if i%2 == 0
                    par[i]=x[i]
                elseif i%2 == 1
                    impar[i]=x[i]
                end
            end
            N1=N/2
            for i=1:N1
                par[i]=1/N*par[i]*exp(-1im*2*pi*i/N1) + exp(-1im*pi/N)
                for j=1:N1
                    impar[j]=1/N*par[i]*impar[j]*exp(-1im*2*pi*i/N1)
                end
            end

        end
```

```
In [ ]: N=4096
        vetorIfftMy=Complex32[1:1:N];
        @time myifft(vetorIfftMy)
```

## 1.1 # Conclusão

Ao termino do trabalho podemos concluir que assim como no Python(notebook 2), as bibliotecas ou funções nativas(no caso do julia,que já nos fornece uma gama de funções), tiveram um tempo de compilação menor,para as funções FFT() e IFFT(),já para as funções myFFT() e myIFFT() - que foram criadas no decorrer do trabalho- tiveram um tempo de compilação maior.

## 1.2 # Referências


www.docs.julialang.org/en/release-0.4/manual/

www.en.wikibooks.org/wiki/Introducing_Julia/