

**ELT062 - OFICINA DE SIMULAÇÃO ANALÓGICA E DIGITAL  
EM CONTROLE**

**TUTORIAL 10**

Instrutores: Alexandre R. Mesquita e Bruno O. S. Teixeira

O objetivo dessa prática é familiarizar o aluno com o uso da ferramenta conhecida como FFT (*Fast Fourier Transform*). FFT é o nome dado a algoritmos rápidos que calculam a DFT (*Discrete Fourier Transform*). Na DFT, o sinal é discreto nos domínios do tempo e da frequência e, portanto, pode ser representado em um computador digital. Note a diferença com relação à DTFT, que transforma um sinal discreto no tempo em um sinal contínuo na frequência.

Conforme a propriedade que já conhecemos da DTFT, a um sinal discreto no tempo corresponde um sinal periódico na frequência. Analogamente, um sinal discreto no domínio da frequência implica um sinal periódico no tempo. Dessa forma, podemos entender a DFT como sendo a DTFT de um sinal discreto e finito no tempo que foi estendido para um sinal discreto e periódico no tempo.

Uma DFT mapeia um sinal discreto no tempo  $x[n]$ ,  $n = 0, 1, 2, \dots, N - 1$  para um sinal discreto na frequência  $X[\omega]$ . Ao vetor de tempos de  $x[n]$

$$[0 \ 1 \ 2 \ 3 \ \dots \ N - 1] \cdot T_s$$

corresponde um vetor de frequências de  $X[\omega]$

$$[0 \ 1 \ 2 \ 3 \ \dots \ N - 1] \cdot \frac{2\pi F_s}{N}$$

onde  $T_s$  é o tempo de amostragem,  $F_s = 1/T_s$  é a frequência de amostragem e  $N$  é o número de pontos disponíveis. Aqui há duas observações importantes:

- o número de pontos da DFT é o mesmo do sinal original;
- a resolução no domínio da frequência ( $2\pi F_s/N$ ) será tanto melhor quanto maior o número  $N$  de pontos amostrados para uma taxa de amostragem fixa.

Ainda é importante notar que a DFT  $X[\omega]$  de sinais reais é anti-simétrica em torno de  $\pi F_s$ . Por isso é frequentemente desnecessário plotar os pontos além dessa frequência.

Iniciemos nosso estudo criando um arquivo de nome `tut10.m`. Calculemos a FFT de uma amostra do sinal

$$x(t) = \cos(2\pi t) * \sin(4\pi t) .$$

Isso é feito com as linhas de código

```
close all
clear
clc

Ts=0.01;
t=0:Ts:8;
x=cos(2*pi*t).*sin(4*pi*t); %sinal no tempo
X=fft(x);                  %transformada
```

Em seguida usamos a definição de vetor de frequências acima para exibir um gráfico da FFT calculada.

```
N=length(t);
f=(0:(N-1))/(N*Ts);    %compare com a expressao do tutorial
figure
subplot(2,1,1)
plot(f,abs(X));        %grafico de magnitude
ylabel('|X|')
subplot(2,1,2)
plot(f,angle(X)*180/pi);    %grafico de fase
xlabel('f [Hz]')
ylabel('\angle X [deg]')
```

Execute o arquivo e observe o gráfico obtido. Uma primeira observação é de que o gráfico é anti-simétrico em torno de  $\pi F_s$ . Observamos também que  $X[\omega]$  apresenta dois picos, um em 1 [Hz] e outro em 3 [Hz]. Esses picos confirmam nosso conhecimento de que  $x[n]$  consiste num sinal periódico. De fato, se plotarmos  $x(t)$ , veremos que é um sinal periódico com período 1 [s].

```
figure
plot(t,x)
xlabel('t [s]')
ylabel('x(t)')
```

Calculemos em seguida a DFT de um sinal mais complexo, que corresponde à resposta de um sistema dinâmico linear a ruído branco.

```
G=tf([10],[1 1 10]);    %funcao de transferencia do sistema
u=0.1*randn(N,1);      %ruído de entrada
y=lsim(G,u,t');        %resposta do sistema
```

Temos, portanto, os seguintes sinais no tempo

```
figure
plot(t,u,t,y)
legend('u','y')
xlabel('t [s]')
```

Analisemos o espectro de  $y$ :

```
Y=fft(y);
figure
subplot(2,1,1)
semilogx(f(1:end/2),20*log10(abs(Y(1:end/2))));
ylabel('|Y|')
subplot(2,1,2)
semilogx(f(1:end/2),unwrap(angle(Y(1:end/2)))*180/pi);
xlabel('f [Hz]')
ylabel('\angle Y [deg]')
```

Note que aqui usamos o comando `unwrap` para evitar saltos no diagrama de fase. Sem esse recurso, o diagrama de fase pode ter uma aparência ruidosa, dado que saltos de múltiplos de  $2\pi$  são possíveis quando calculamos a fase pelo comando `angle`. Execute o arquivo até este ponto e observe os resultados. Note como  $y$  é um sinal passa baixa.

Conhecendo um par de entradas e saídas  $u$  e  $y$  do sistema  $G$ , podemos estimar a função de transferência fazendo  $G(j\omega) = Y(j\omega)/U(j\omega)$ . Este consiste num método empírico de identificar um sistema dinâmico linear. Em seguida, vamos comparar essa função de transferência obtida “experimentalmente” com a função de transferência conhecida.

```

U=fft(u);
G2=Y./U;      %funcao de transferencia estimada
figure
[mag,pha,w]=bode(G);
subplot(2,1,1)
semilogx(w/(2*pi),20*log10(squeeze(mag)),'b');
hold on
semilogx(f(1:end/2),20*log10(abs(G2(1:end/2))),'r');
ylabel('|G|')
legend('conhecida','estimada')
subplot(2,1,2)
semilogx(w/(2*pi),squeeze(pha),'b');
hold on
semilogx(f(1:end/2),unwrap(angle(G2(1:end/2)))*180/pi,'r');
ylabel('\angle G')
xlabel('f [Hz]')
```

Execute o arquivo até este ponto e observe os resultados. Aqui é interessante executar o arquivo várias vezes. Você notará que, como a entrada  $u$  é sempre diferente, algumas vezes a aproximação da função de transferência será melhor que outras.

Há três fatores que limitam a acurácia desse método. O primeiro deles é o fato de a DFT apenas aproximar a transformada de Fourier, sendo essa aproximação tanto melhor quanto maior o número de pontos. O segundo fator é o sinal de entrada, que deve possuir um conteúdo espectral rico o bastante. Se, por exemplo,  $U(j\omega_0) \approx 0$ , não há como identificar  $G(j\omega_0)$  a partir da divisão  $Y(j\omega_0)/U(j\omega_0)$ . O terceiro fator, que não esteve presente em nossa simulação, é a influência do ruído de medida e de processo.

Para entender um pouco melhor as diferenças entre a DFT e a FT, tentemos realizar uma convolução usando `fft`. Definiremos dois vetores  $a$  e  $b$  e realizaremos a convolução por dois métodos: no primeiro usaremos o comando `conv`; no segundo vamos multiplicar as DFTs dos dois sinais e depois tomar a transformada inversa.

```

a=[1 2 3 4 5];
b=[6 5 4 3 2];
c=conv(a,b) %convolucao no tempo
```

```

A=fft(a);
B=fft(b);
```

```

C=ifft(A.*B) %DFT inversa
```

Execute o arquivo até este ponto. Vemos que os resultados são discrepantes. A explicação para isso é que, enquanto o primeiro método consiste em uma convolução linear, o segundo consiste numa convolução circular, o que acontece devido ao fato de que a DFT assume que o sinal discreto no tempo é periódico.

Ainda é possível obter a convolução linear a partir da DFT. Para isso, basta preencher os sinais  $a$  e  $b$  com zeros de forma que a convolução circular dê o mesmo resultado que a convolução linear.

```
la=length(a);  
lb=length(b);  
  
a=[a zeros(1,lb-1)] %zero-padding  
b=[b zeros(1,la-1)] %zero-padding  
  
A=fft(a);  
B=fft(b);  
  
C=ifft(A.*B)
```

Execute o arquivo e observe que a convolução linear foi obtida por ambos os métodos. Usar a FFT desta forma para realizar convoluções é uma prática comum em filtragem de sinais.

**Comentário:** Um detalhe digno de nota sobre a implementação da FFT é que os algoritmos são mais rápidos quando o número de pontos é uma potência de 2. Por isso, é uma prática comum completar o sinal com zeros até que o número de pontos atinja a próxima potência de 2.