Universidade Tecnológica Federal do Paraná - UTFPR
Campus Curitiba
Departamento Acadêmico de Eletrônica - DAELN
Curso de Engenharia Eletrônica
Prof. Gustavo B. Borba e Humberto R. Gamba



# Notas de aula e prática MATLAB #08

**Vale nota, individual ou em dupla, observar prazo e instruções de entrega no moodle**

***Também tem bastante conteúdo caprichado nos slides e código em IF69D_Slides_08_arquivos.zip***

**Arquivos necessários**

1. cameraman.tif [MATLAB built-in]

## *8a) DFT 1D*

A transformada de Fourier é a operação que recebe como entrada um sinal no domínio do tempo (ou do espaço, no caso de uma imagem) e apresenta na saída o sinal no domínio da frequência. Na maioria dos textos, a transformada discreta de Fourier (DFT) de um sinal 1D é definida pela equação a seguir [[BB] Eq. 13.45].

$$G(m) = \frac{1}{\sqrt{M}} \sum_{u=0}^{M-1} g(u) \cdot e^{-i2\pi \frac{mu}{M}}$$

$g(u)$ é o sinal original, de comprimento $M$ ($u = 0 \ldots M-1$) e $G(m)$ é o espectro (domínio da frequência), também de comprimento $M$. Ambos são vetores de números complexos, isto é, cada valor de $g(u)$ e de $G(m)$ possui uma parte real e uma parte imaginária [[BB] Eq. 13.47]:

$$g(u) = g_{Re}(u) + i \cdot g_{Im}(u)$$

$$G(m) = G_{Re}(m) + i \cdot G_{Im}(m)$$

Uma das conveniências desta equação é que ela é concisa. Mas se fosse pra implementar a DFT computacionalmente, poderíamos expandir esta equação usando a fórmula de Euler para números complexos, mostrada a seguir.

$$e^{-ix} = cos(x) - i \cdot sin(x)$$

Após a substituição e mais uma manipuladinha, obtém-se as equações a seguir [[BB] Eq 13.49 e 13.50]. No entanto, é importante lembrar que, computacionalmente, não se realiza a implementação destas equações para obter a DFT. O método utilizado para a obtenção da DFT é o da FFT (Fast Fourier Transform).

$$G_{Re}(m) = \frac{1}{\sqrt{M}} \sum_{u=0}^{M-1} g_{Re}(u) \cdot C_m^M(u) + g_{Im}(u) \cdot S_m^M(u)$$

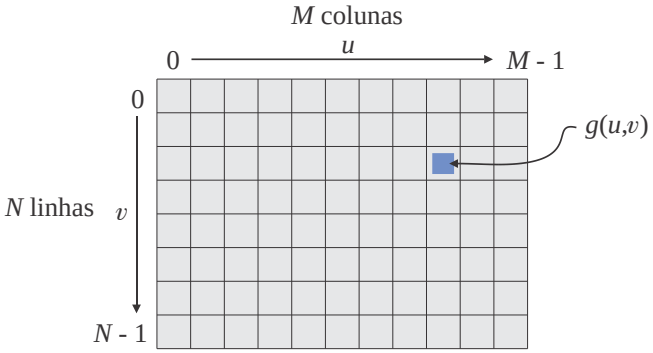
$$G_{Im}(m) = \frac{1}{\sqrt{M}} \sum_{u=0}^{M-1} g_{Im}(u) \cdot C_m^M(u) - g_{Re}(u) \cdot S_m^M(u)$$

$$C_m^M(u) = cos\left(2\pi \frac{mu}{M}\right), \quad S_m^M(u) = sin\left(2\pi \frac{mu}{M}\right)$$

### *8b) DFT 2D*

A definição da DFT para um sinal de duas dimensões, com é o caso de imagens, é mostrada a seguir [[BB] Eq 14.1]. A figura que mostra a convenção para o sistema de coordenadas foi redesenhada de [[BB] Figura 2.6].

$$G(m,n) = \frac{1}{\sqrt{MN}} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} g(u,v) \cdot e^{-i2\pi \frac{mu}{M}} \cdot e^{-i2\pi \frac{nv}{M}}$$




A equação anterior, da definição da DFT em 2D pode ser reescrita na sua forma *separável* conforme mostrado a seguir [[BB] Eq. 14.7].

$$G(m,n) = \frac{1}{\sqrt{N}} \sum_{v=0}^{N-1} \left[ \frac{1}{\sqrt{M}} \sum_{u=0}^{M-1} g(u,v) \cdot e^{-i2\pi \frac{mu}{M}} \right] \cdot e^{-i2\pi \frac{nv}{M}}$$

A parte dentro dos colchetes é a DFT 1D da linha $v$ da imagem, pois $v$ é fixo e o somatório é feito para todos os valores de $u$ (todas as colunas). Matematicamente, pode-se utilizar a notação $g(\cdot,v)$ (lê-se: todas as colunas da linha $v$). Isto significa que, para calcular a DFT 2D, pode-se fazer primeiramente a DFT 1D de cada linha da imagem, e em seguida aplicar uma nova DFT 1D em cada coluna da DFT anterior, conforme expresso pelas equações a seguir [[BB] Tópico 14.1.2; [GW] Tópico 4.11.1 denomina esta propriedade de *separabilidade da DFT 2D*]. $g''(u,\cdot)$ é a própria DFT 2D, $G(m,n)$.

$$g'(\cdot,v) \leftarrow DFT\{g(\cdot,v)\} \quad para \;\; 0 \le v < N$$

$$g''(u,\cdot) \leftarrow DFT\{g'(u,\cdot)\} \quad para \;\; 0 \le u < M$$

### *8c) Visualização da DFT*

Na transformada de Fourier, o componente correspondente à frequência zero é denominado coeficiente DC. Na saída da DFT 2D, os componentes DC estão localizados nos cantos da matriz e, consequentemente, os componentes de frequência mais alta no centro. No entanto, para a visualização e também para facilitar a aplicação de filtros, convenciona-se posicionar o componente DC no centro da matriz de saída. Isso pode ser conseguido multiplicando-se os pixels por $(-1)^{u+v}$ antes de submeter a imagem à transformada. A outra maneira é reposicionar cada quadrante da saída da transformada, conforme mostrado na figura a seguir [[BB] Figura 14.3]. No MATLAB, este reposicionamento pode ser obtido utilizando-se a função `fftshift`. Com isso, na DFT as componentes de frequência mais baixa estão posicionadas na região central, e as de frequência mais alta, nas bordas.


Cada elemento da saída da DFT é um número complexo. Para visualizar esses valores, deve-se calcular as suas *magnitudes* (módulo de um número complexo). A magnitude do espectro é obtida a partir da equação a seguir. No MATLAB, a função que calcula a magnitude do espetro é a `abs`.

$$|G(m)| = \sqrt{G_{Re}^2(m) + G_{Im}^2(m)}$$

Ainda, é necessário levar em consideração que os valores numéricos na saída da DFT apresentam diferenças de amplitude muito grandes, especialmente do componente DC em relação aos demais. Com isso, ao tentar visualizar diretamente a saída da DFT, o que pode ser feito reescalonando os valores para a faixa de 0 até 1, por exemplo (função `mat2gray`), o resultado é um único ponto de intensidade alta no centro (componente DC). Assim, costuma-se utilizar o logaritmo para mostrar os valores. No MATLAB, basta fazer `mat2gray(log(1+dftmag))`, onde `dftmag` é a magnitude do espectro.

```
% fft_vis [script]
clc, clear, close all

img = imread('cameraman.tif');
img_d = double(img);
figure, imshow(img)
title('Imagem de entrada')

% DFT 2D
dft = fft2(img_d);
% Centraliza
% (reposicionamento ou shifiting)
dft_s = fftshift(dft);
% Magnitude
dft_s_m = abs(dft_s);
% Visualização
dft_s_m_v = mat2gray(log(1+dft_s_m));
figure, imshow(dft_s_m_v)
title('DFT 2D')
```


### *8d) Propriedades da translação e da rotação da DFT 2D*

O espectro não é sensível à translação da imagem, mas rotaciona no mesmo ângulo que a imagem [[GW] Tópico 4.6.5].

### *8e) Teorema da convolução*

Afirma que uma convolução no domínio do espaço é equivalente a uma multiplicação ponto-a-ponto no domínio da frequência. Na equação abaixo, $g(u,v)$ é a imagem no domínio espacial (imagem original), $h(u,v)$ é a máscara de convolução, * significa convolução, e $g_f(u,v)$ é a imagem de saída, isto é, a imagem filtrada. Com isso, é possível implementar, por exemplo, filtros passa-baixas, passa-altas, e inúmeras outras possibilidades de filtros no domínio da frequência. Na equação abaixo, $G(m,n)$ é a transformada de Fourier da imagem, $H(m,n)$ é o filtro, de mesmas dimensões da imagem, que é multiplicado ponto-a-ponto pela transformada de Fourier da imagem, e $DFT^{-1}$ é a transformada inversa de Fourier, necessária para reconstruir a imagem, isto é, devolvê-la para o domínio espacial e ser visualizada.

$$g_f(u,v) = g(u,v) * h(u,v) = DFT^{-1}\{G(m,n) \cdot H(m,n)\}$$

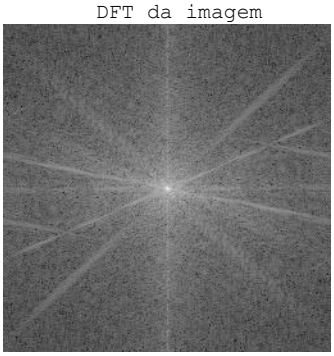
### *8f) Filtragem no domínio da frequência*

Lembre que os componentes de baixa frequência então no centro da matriz da transformada e, as componentes de alta frequência, nas bordas. No exemplo Octave a seguir são mostradas as operações de passa-altas e passa-baixas no domínio da frequência usando filtros ideais (o perfil é um degrau). Pode-se observar o efeito indesejado de *ringing*. Isto ocorre devido ao corte abrupto das componentes de frequência na transformada. Em sinais e sistemas e processamento digital de sinais isso é conhecido como *efeito Gibbs* [[SS] Capítulo 11].

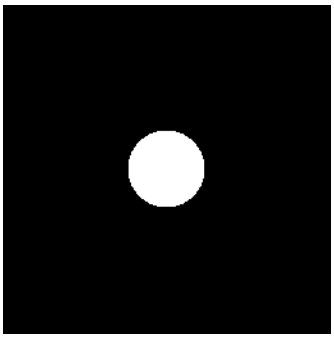
```
% fft_PbPaIdeal [script]
clc, clear, close all

img = imread('cameraman.tif');
img_d = double(img);
figure, imshow(img)
title('Imagem de entrada')

% DFT 2D
F = fft2(img_d);
% Centralizada
% (reposicionamento ou shifiting)
F_s = fftshift(F);
imshow(mat2gray(log(1+abs(F_s))))
title('DFT da imagem')

% Filtro passa-baixas ideal
[nr nc] = size(img_d);
rc = round(nr/2);
cc = round(nc/2);
bw = false(nr,nc);
bw(rc,cc) = 1;
d = bwdist(bw);
raio = 30;
H_pb = double(d < raio);
figure, imshow(H_pb)
title('Filtro PB ideal')
% Filtro passa-altas ideal
H_pa = 1 - H_pb;
figure, imshow(H_pa)
title('Filtro PA ideal')

% Multiplicação ponto-a-ponto
% (aplicação do filtro)
F_pb = F_s .* H_pb;
F_pa = F_s .* H_pa;

% DFT 2D inversa
img_pb = ifft2(F_pb);
img_pa = ifft2(F_pa);

% Magnitude e visualização
img_pb = uint8(abs(img_pb));
figure, imshow(img_pb)
title('Saída PB')
img_pa = uint8(abs(img_pa));
figure, imshow(img_pa)
title('Saída PA')
```

Imagem de entrada

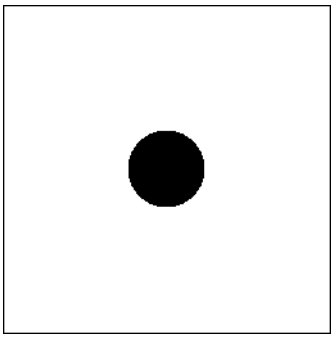
DFT da imagem



Filtro PB ideal



Filtro PA ideal



Saída PB

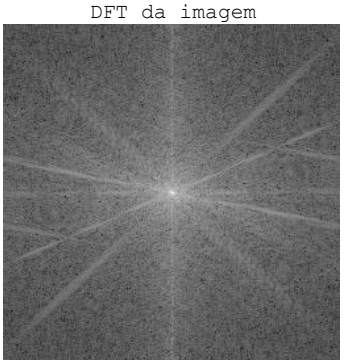
Saída PA

A seguir é mostrado um exemplo utilizando filtro Gaussiano no domínio da frequência. Só lembre que a `fspecial`, que está sendo utilizada para gerar a função Gaussiana 2D (filtro que é multiplicado ponto-a-ponto pelo espectro), gera Gaussianas cuja soma dos coeficientes é sempre igual a 1. Então, para não alterar o valor médio da imagem e usá-las como filtros no domínio da frequência, elas são reescalonadas para a faixa de 0 até 1 (função `mat2gray`). Dessa forma, o centro do espectro, que contém o valor DC (valor médio da imagem), é multiplicado por 1 (pico da Gaussiana).

```
% fft_PbGaussiano [script]
clc, clear, close all

img = imread('cameraman.tif');
img_d = double(img);
figure, imshow(img)
title('Imagem de entrada')

% DFT 2D
F = fft2(img_d);
% Centralizada
% (reposicionamento ou shifiting)
F_s = fftshift(F);
figure,
imshow(mat2gray(log(1+abs(F_s))))
title('DFT da imagem')

% Filtro passa-baixas Gaussiano
[nr nc] = size(img_d);
H_pb=fspecial('gaussian',[nr nc],25);
H_pb = mat2gray(H_pb);
figure, imshow(mat2gray(H_pb))
title('Filtro PB Gaussiano')
figure, surf(H_pb)
title('Filtro PB Gaussiano')

% Multiplicação ponto-a-ponto
% (aplicação do filtro)
F_pb = F_s .* H_pb;

% DFT 2D inversa
img_pb = ifft2(F_pb);

% Magnitude e visualização
img_pb = uint8(abs(img_pb));
figure, imshow(img_pb)
title('Saída PB')
```

Imagem de entrada

DFT da imagem



Filtro PB Gaussiano

Saída PB

## Referências


[GW] Rafael C. Gonzalez, Richard E. Woods, Steven L. Eddins, Digital image processing, Pearson Prentice Hall, 3rd ed, 2008.

[BB] Wilhelm Burger, Mark Burge, Digital image processing – an algorithmic introduction using Java, Springer, 2010.

[SS] Steven W. Smith, The Scientist and Engineer's Guide to Digital Signal Processing, California Technical Pub, 1997. Disponível em http://www.dspguide.com/ch11/4.html