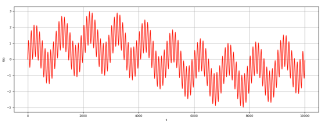
AULA PRÁTICA 3. *PRACTICAL CLASS 3*

1. TRANSFORMADA DE FOURIER 1D. *1D FOURIER TRANSFORM.*

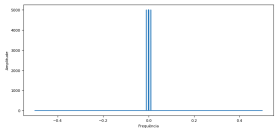
1.1. Ler o ficheiro **soma3freq.npy** fornecido e representar a série correspondente a um sinal com a duração de 1 segundo. *Read the supplied **soma3freq.npy** file and represent the series corresponding to a signal lasting 1 second.*

```
serie = np.load('soma3freq.npy')
plt.plot(serie, 'r')
plt.xlabel('t', fontsize=8); plt.ylabel('f(t)', fontsize=8)
plt.tick_params(labelsize=8); plt.grid()
```



1.2. Aplicar a transformada de Fourier 1D à série e representar o espectro na forma centrada. *Apply the 1D Fourier transform to the series and represent the spectrum in centered form.*

```
dft = np.fft.fft(serie-np.average(serie))
plt.figure()
plt.plot(np.fft.fftshift(np.abs(dft)))
plt.ylabel('Amplitude', fontsize=8)
plt.xlabel('Frequência', fontsize=8)
plt.tick_params(labelsize=8)
```



1.3. A série resulta da soma de três senoidais com frequências diferentes. Representar cada uma isoladamente, com ao eixo das abcissas em função do tempo. *The series results from the sum of three sine waves with different frequencies. Represent each one separately, with the axis of the abscissa as a function of time.*

```
...

ift = np.fft.ifft(dft1)
plt.figure()
plt.plot(ift)
plt.ylabel('f1(t)', fontsize=8)
plt.xlabel('t', fontsize=8)
plt.tick_params(labelsize=8)
```

1.4. Ler a imagem **marilyn.tif** e dela extrair uma linha, ou coluna, ou um perfil (alínea 2.3 da aula prática 1). Aplicar a transformada de Fourier 1D ao perfil e representar o espectro na forma centrada.

2. TRANSFORMADA DE FOURIER 2D. *2D FOURIER TRANSFORM.*

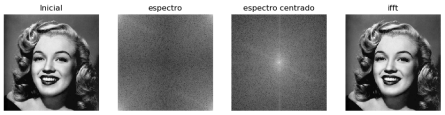
2.1. Fazer um pequeno programa para calcular o espectro e a magnitude da imagem seguinte. *Develop a small program to calculate the spectrum and magnitude of the next image.*

$$f(x,y) = \begin{bmatrix} 97 & 150 \\ 123 & 27 \end{bmatrix}$$

2.2. Calcular a imagem inicial a partir do resultado da alínea anterior. *Calculate the initial image from the result of the previous paragraph.*

2.3. Determinar a Transformada Direta de Fourier (DFT) da imagem **marilyn.tif** e representar o respectivo espectro da imagem de magnitude nas formas não-centrada e centrada. Para poder visualizar o espectro, usar o operador logarítmico para escalar a imagem da magnitude. Determinar a Transformada Inversa de Fourier (IFT) do espectro (completo) da alínea anterior e ver o resultado. *Determine the Fourier Discrete Direct Transform (DFFT) of the image **marilyn.tif** and represent the respective magnitude image spectrum in non-centered and centered forms. To be able to view the spectrum, use the logarithmic operator to scale the magnitude image. Determine the Inverse Fourier Transform (IFT) of the (complete) spectrum of the previous paragraph and see the result.*

```
dfft = np.fft.fft2(Img)
espectro = abs(dfft)
plt.imshow(np.log10(abs(np.fft.fftshift(dfft))), 'gray')
ifft = abs(np.fft.ifft2(dfft))
```



3. FILTRAGEM NO DOMÍNIO DAS FREQUÊNCIAS. *FILTERING IN FREQUENCY DOMAIN.*

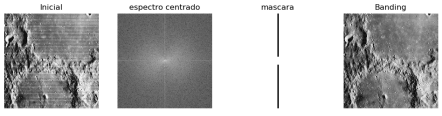
3.1. Construír, no espaço de Fourier, os filtros passa-baixa plano, gaussiano e Butterworth, e aplicá-los à DFT da imagen **circles.tif**. *To construct, in Fourier space, the low-pass, Gaussian and Butterworth filters, and apply them to the DFT of the **circles.tif** image.*

```
def plano_lp_2d(a, b, f):
    mask = np.ones((a, b))
    mask[0, 0] = 0
    mask[0, b-1] = 0
    mask[a-1, 0] = 0
    mask[a-1, b-1] = 0
    D = scipy.ndimage.distance_transform_edt(mask)
    p = D<=f
    return p
```

```
def gauss_lp_2d(lin, col, mu, sigma):
    x, y = np.meshgrid(np.linspace(-col/2, col/2, col), \
                       np.linspace(-lin/2, lin/2, lin))
    g = np.exp(-((x-mu)**2+(y-mu)**2)/(2*sigma**2))
    return g
```

3.2. Atenue o ruído periódico da imagem **Stripping_Noise.tif** usando a filtragem no espaço de Fourier. *Attenuate the periodic noise of the **Stripping_Noise.tif** image using Fourier space filtering.*

```
mask1 = espectro_c_log>=3
mask2 = np.zeros((lin, col))
q = 5
mask2[q:int(lin/2), 0:q] = 1; mask2[int(lin/2):lin-q, 0:q] = 1
mask2[q:int(lin/2), col-q+1:col] = 1; mask2[int(lin/2):lin-q, col-q+1:col] = 1
mask2 = np.fft.ifftshift(mask2)
mask = np.logical_not(np.logical_and(mask1, mask2))
```



4. TEOREMA DA CONVOLUÇÃO. *CONVOLUTION THEOREM.*

4.1. Ler a imagem **marilyn03.tif**. Comprovar o Teorema da Convolução, com a aplicação de um filtro passa-baixa $H_{11\times11}$. *Read the picture marilyn03.tif. Confirm the Convolution Theorem by applying a $H_{11\times11}$ low-pass filter.*

```
filtro0 = np.ones((11, 11))/(11*11)
a = lin-11; b = col-11
filtro1 = np.pad(filtro0, [(a//2, ), (b//2, )], mode='constant')
filtro1 = np.fft.fftshift(filtro1)
```