SIS Scuola Interdipartimentale delle **Scienze**, dell'**Ingegneria** e della **Salute**

DiST Dipartimento di Scienze e Tecnologie

# L. Magistrale in IA (ML&BD)

# Scientific Computing (part 2 – 6 credits)


## prof. Mariarosaria Rizzardi

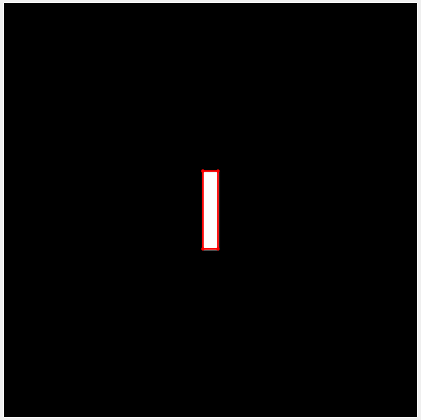
Centro Direzionale di Napoli – Bldg. C4
room: n. 423 – North Side, 4th floor
phone: 081 547 6545
email: mariarosaria.rizzardi@uniparthenope.it

# Contents

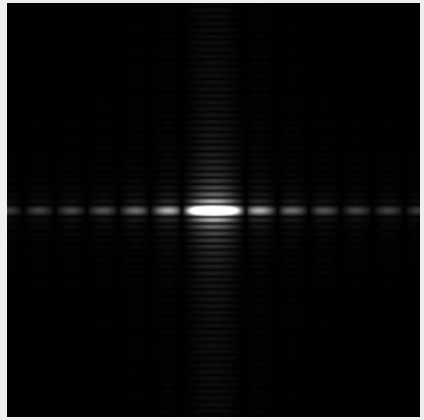
- Other properties of 2D FT.
- Application of 2D FT to images.

# 2D FT properties

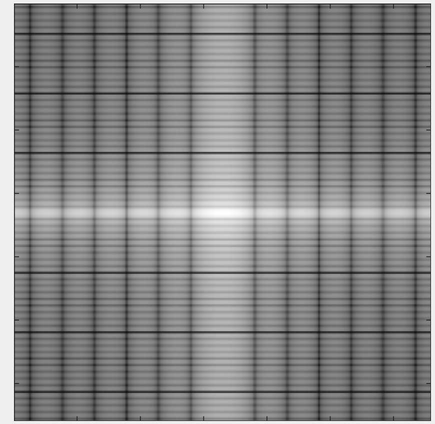
The FT of an image is insensitive to translation.



imshow(IO)

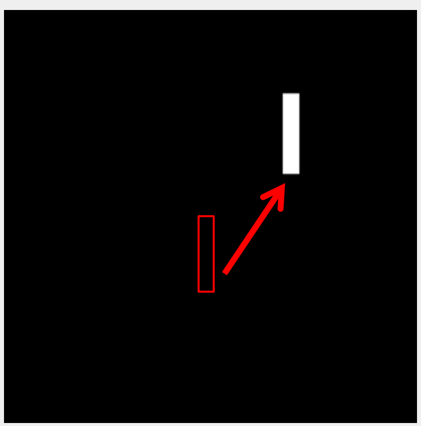


imshow(uint8(abs(FO)))

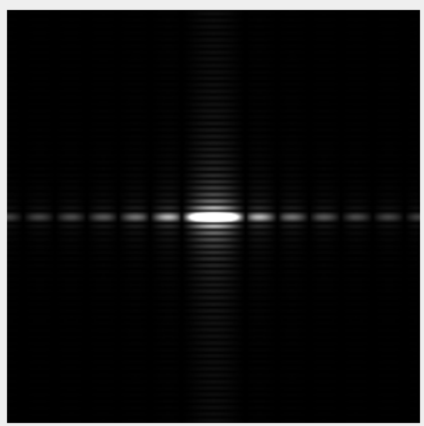


imagesc(log10(abs(FO)));
colormap(gray)

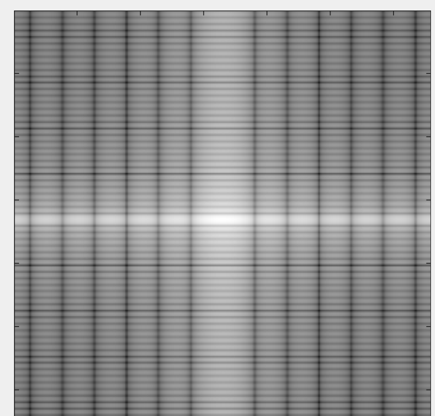
equal spectrum



imshow(IT)



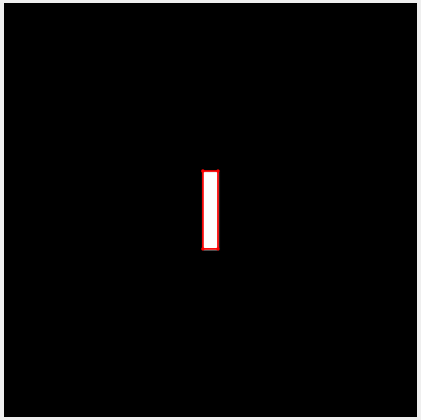
imshow(uint8(abs(FT)))



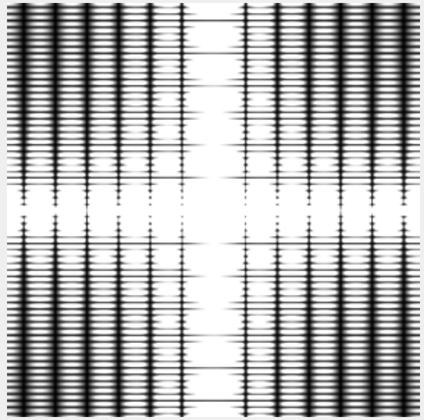
imagesc(log10(abs(FT)));
colormap(gray)

# 2D FT properties

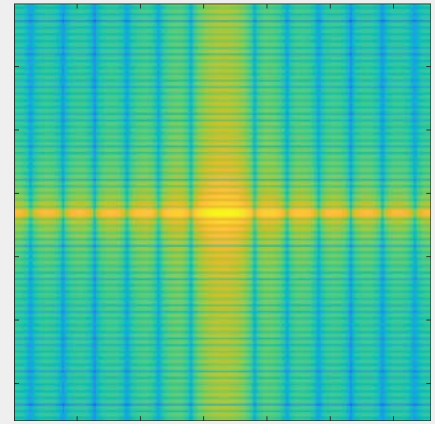
The **FT** of an image is insensitive to translation.



imshow(IO)

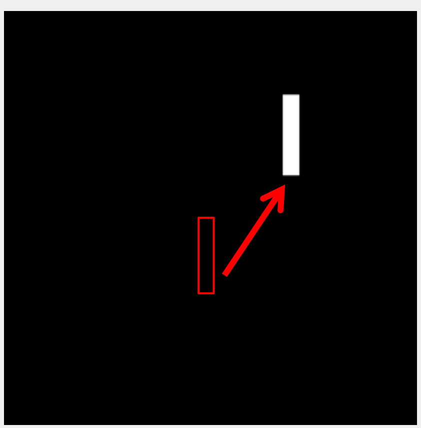


imshow(abs(FO))

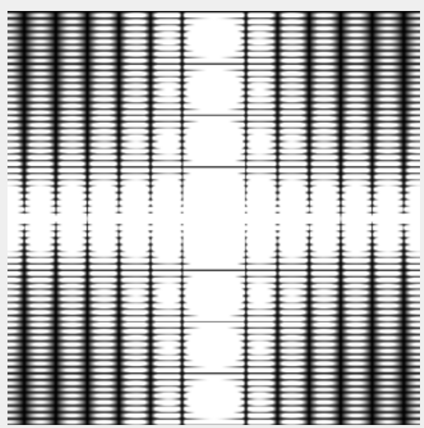


imagesc(log10(abs(FO)))

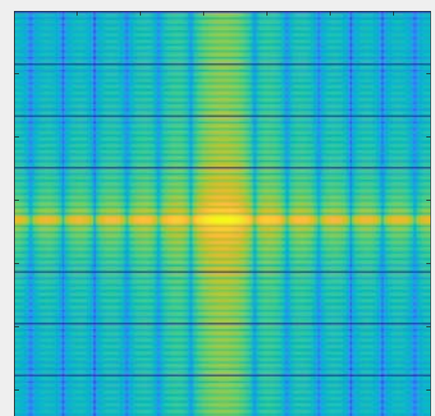
equal spectrum



imshow(IT)



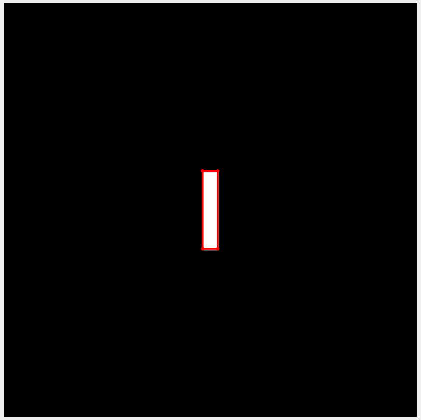
imshow(abs(FT))



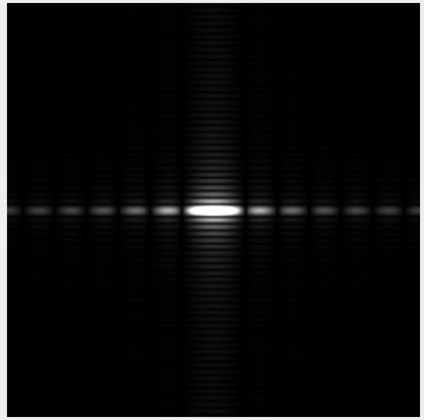
imagesc(log10(abs(FT)))

# 2D FT properties

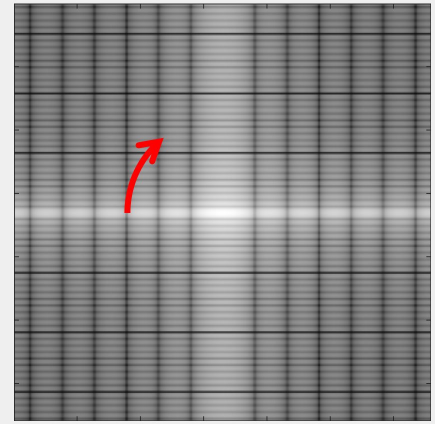
The **FT** of an image is sensitive to rotation.



imshow(IO)

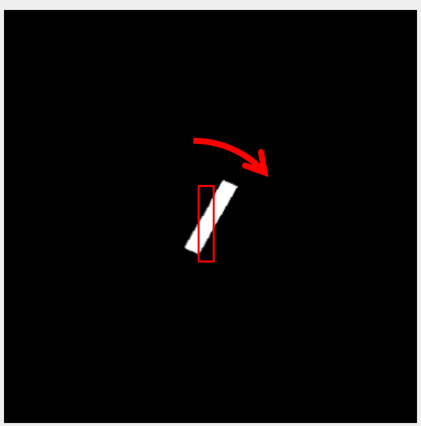


imshow(uint8(abs(FO)))

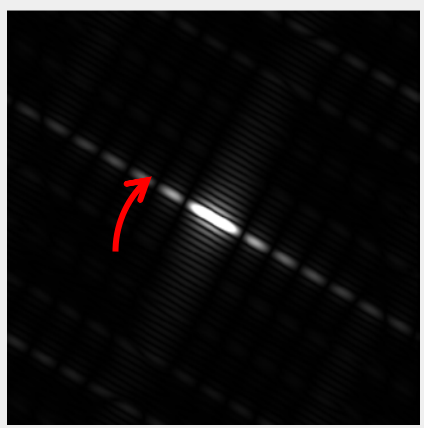


imagesc(log10(abs(FR)));
colormap(gray)

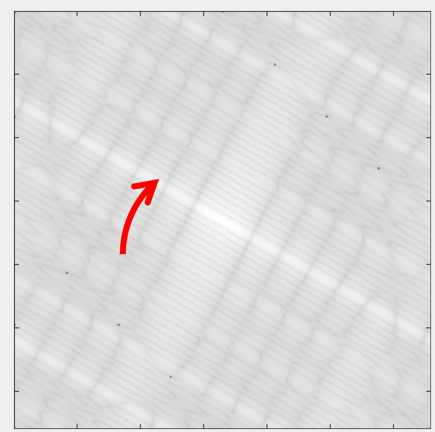
## The spectrum rotates by the same angle as the image



imshow(IR)



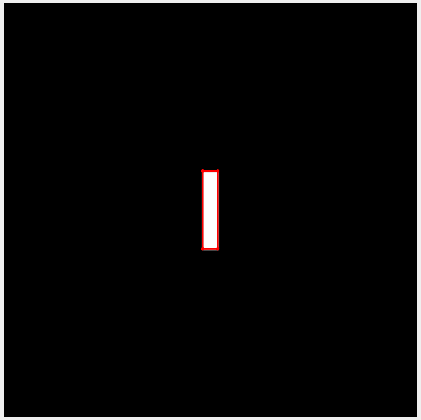
imshow(uint8(abs(FR)))



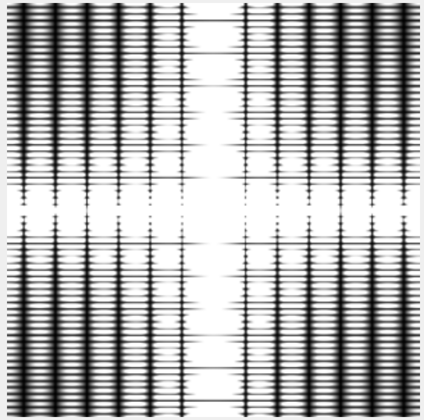
imagesc(log10(abs(FR)));
colormap(gray)

# 2D FT properties

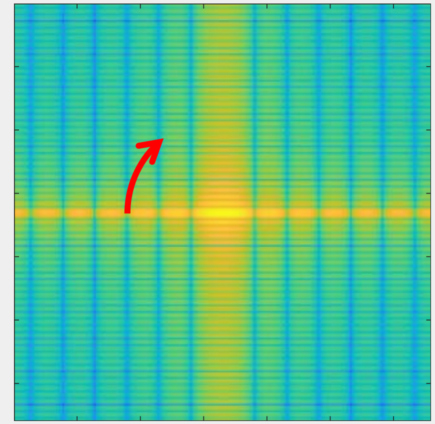
The **FT** of an image is sensitive to rotation.



imshow(IO)

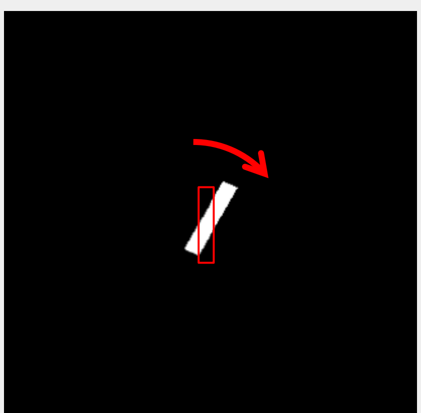


imshow(abs(FO))

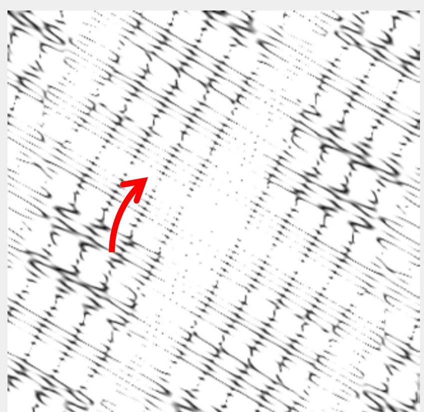


imagesc(log10(abs(FO)))

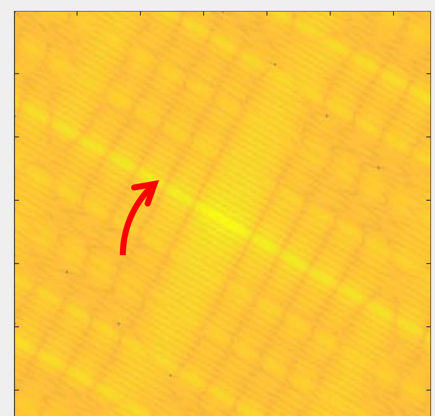
## The spectrum rotates by the same angle as the image



imshow(IR)



imshow(abs(FR))



imagesc(log10(abs(FR)))

# **Example:** Which part of Fourier spectrum is fundamental?

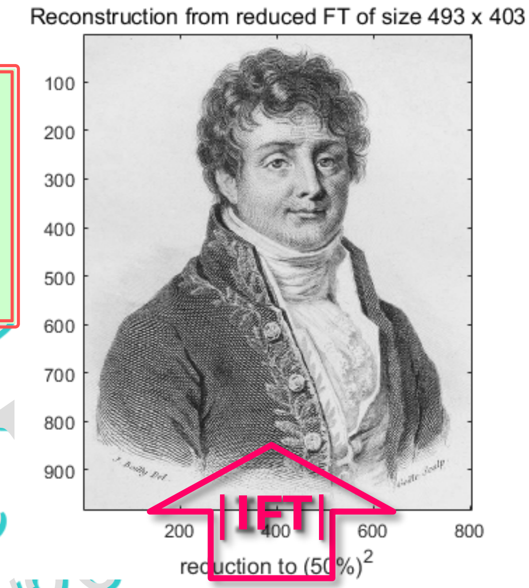
```
f=imread('./Fourier.jpg'); [m,n]=size(f);
if rem(m,2) == 0,  f=[f;zeros(1,size(f,2))];
else               f(end,:)=zeros(1,size(f,2)); m=m-1;
end
if rem(n,2) == 0,  f=[f zeros(size(f,1),1)];
else               f(:,end)=zeros(size(f,1),1); n=n-1;
end
figure; imagesc(f); axis equal; colormap(gray); axis tight
[h,k]=meshgrid(0:n-1, 0:m-1); F=fftshift(fft2(f,m,n)).*(-1).^(h+k);
F=[F;F(1,:)]; F=[F F(:,1)];
figure; imagesc(log10(abs(F))); colormap('jet'); axis equal; axis tight
mMid=m/2+1; nMid=n/2+1;
perc=0.20; Hm=fix(m/2*perc); Hn=fix(n/2*perc);
I=mMid-Hm : mMid+Hm; J=nMid-Hn : nMid+Hn; FF=zeros(size(F)); FF(I,J)=F(I,J);
ff=fftshift(ifft2(FF,m,n)) .*(-1).^(h+k); ff=[ff;ff(1,:)]; ff=[ff ff(:,1)];
figure; imagesc(real(ff)); colormap(gray); axis equal; axis tight
xlabel(['reduction to (' num2str(100*perc) '%)^2'])
title(['Reconstruction from reduced FT of size ' num2str(2*Hm+1) ' x ' num2str(2*Hn+1)])
```

- make the input periodic and m, n even
- original image
- non-optimal algorithm
- plot spectrum
- reduced FT
- non-optimal algorithm

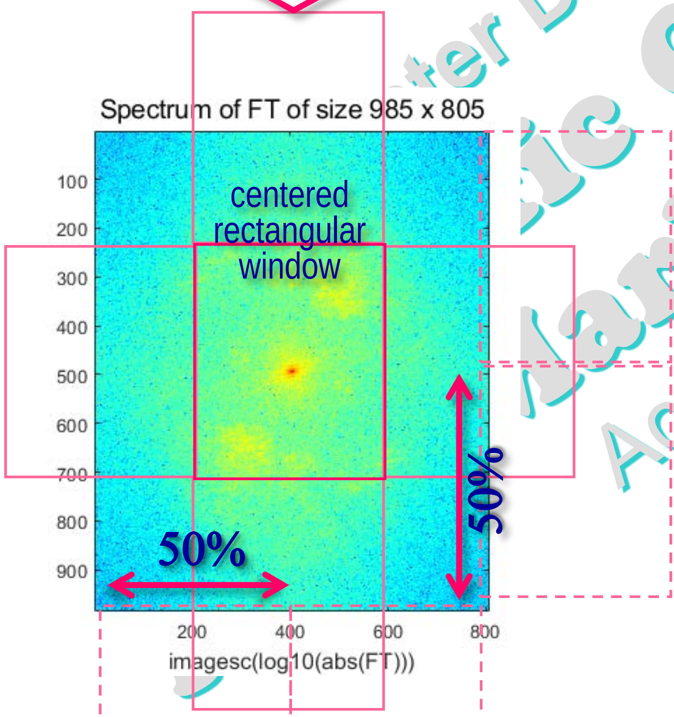
# Example: Which part of Fourier spectrum is fundamental?


```
mMid=m/2+1; nMid=n/2+1;
perc=0.20;
Hm=fix(m/2*perc); Hn=fix(n/2*perc);
I=mMid-Hm : mMid+Hm;
J=nMid-Hn : nMid+Hn;
FF=zeros(size(F)); FF(I,J)=F(I,J);
```

reduced FT

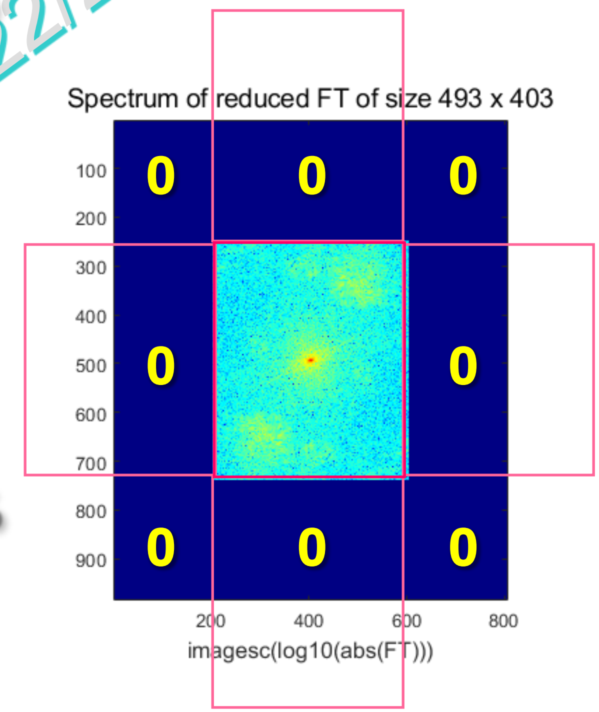


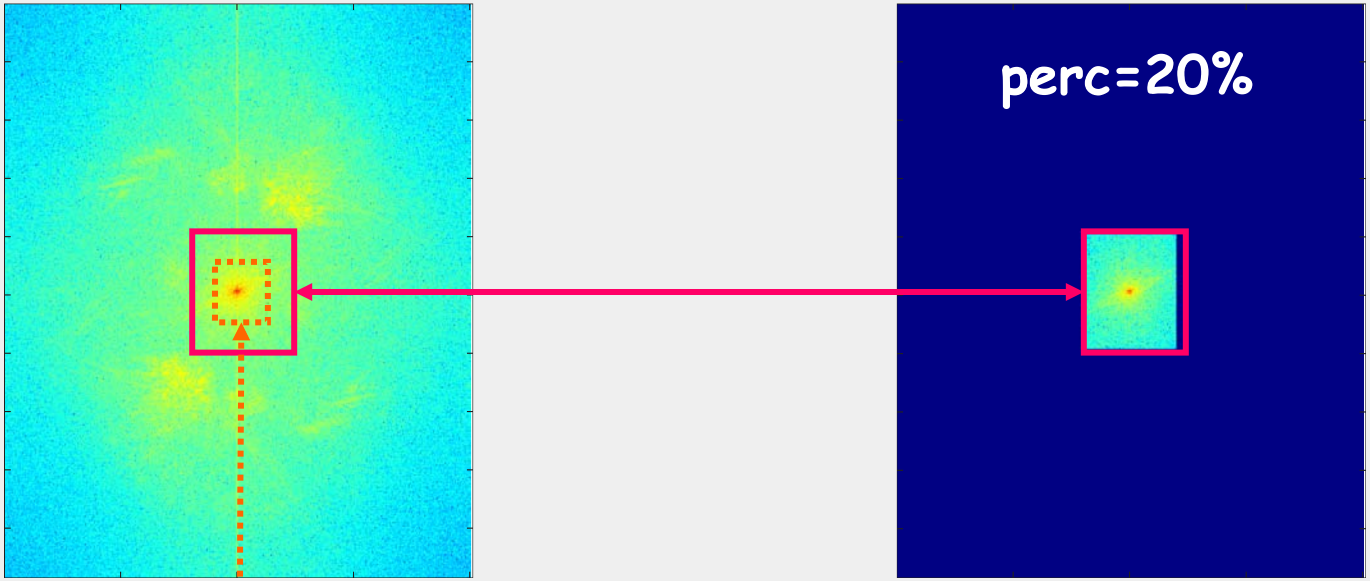

## What is perc?

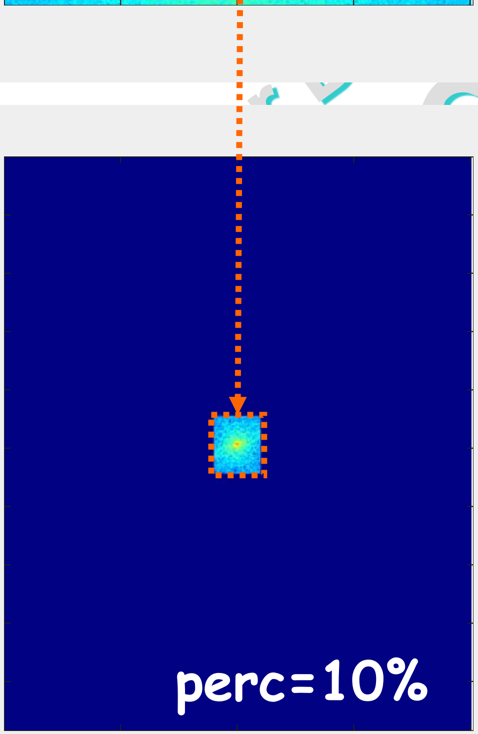



perc=0.50

it corresponds to 25% of the middle part of the Spectrum





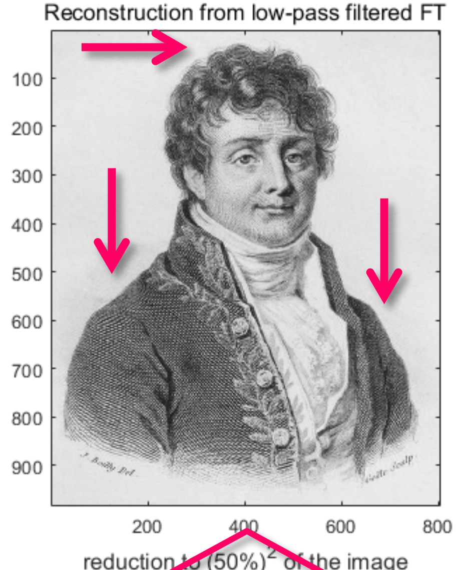



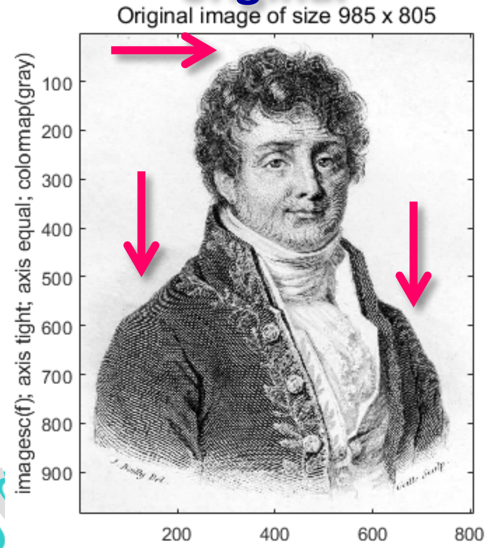

# The **key part** of Fourier Spectrum is the middle!

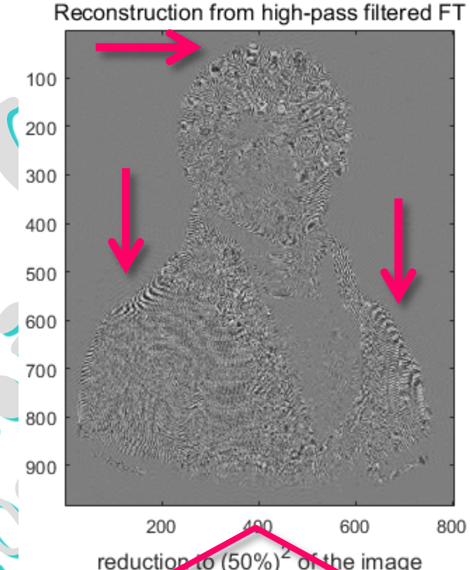
But, to reconstruct the image, we need the whole large matrix, since the reduced FT must be placed in the center.

# low-pass filter VS high-pass filter

## original



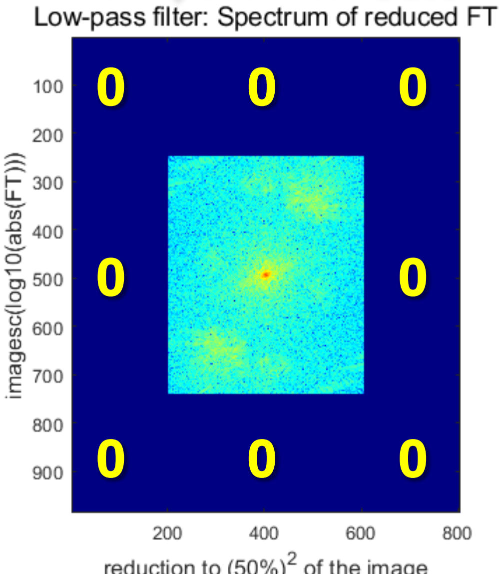

|IFT| details are lost

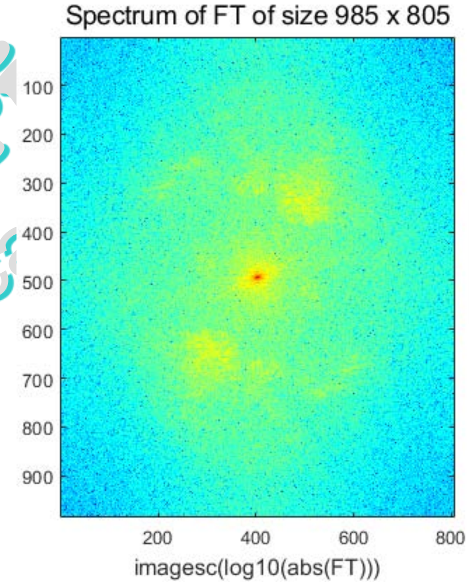
|FT|

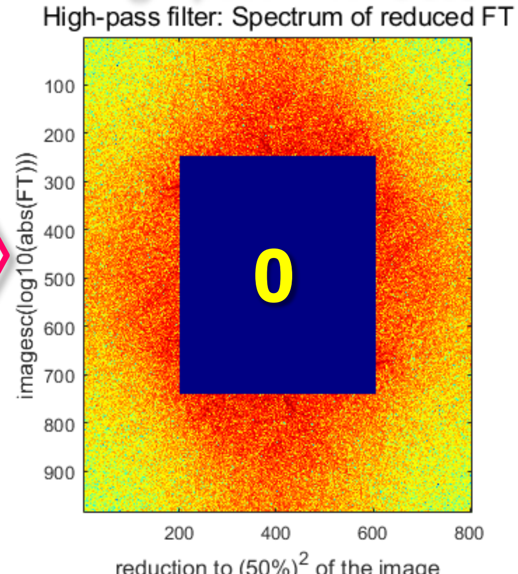
only details are retained |IFT|

low-pass filter 50%

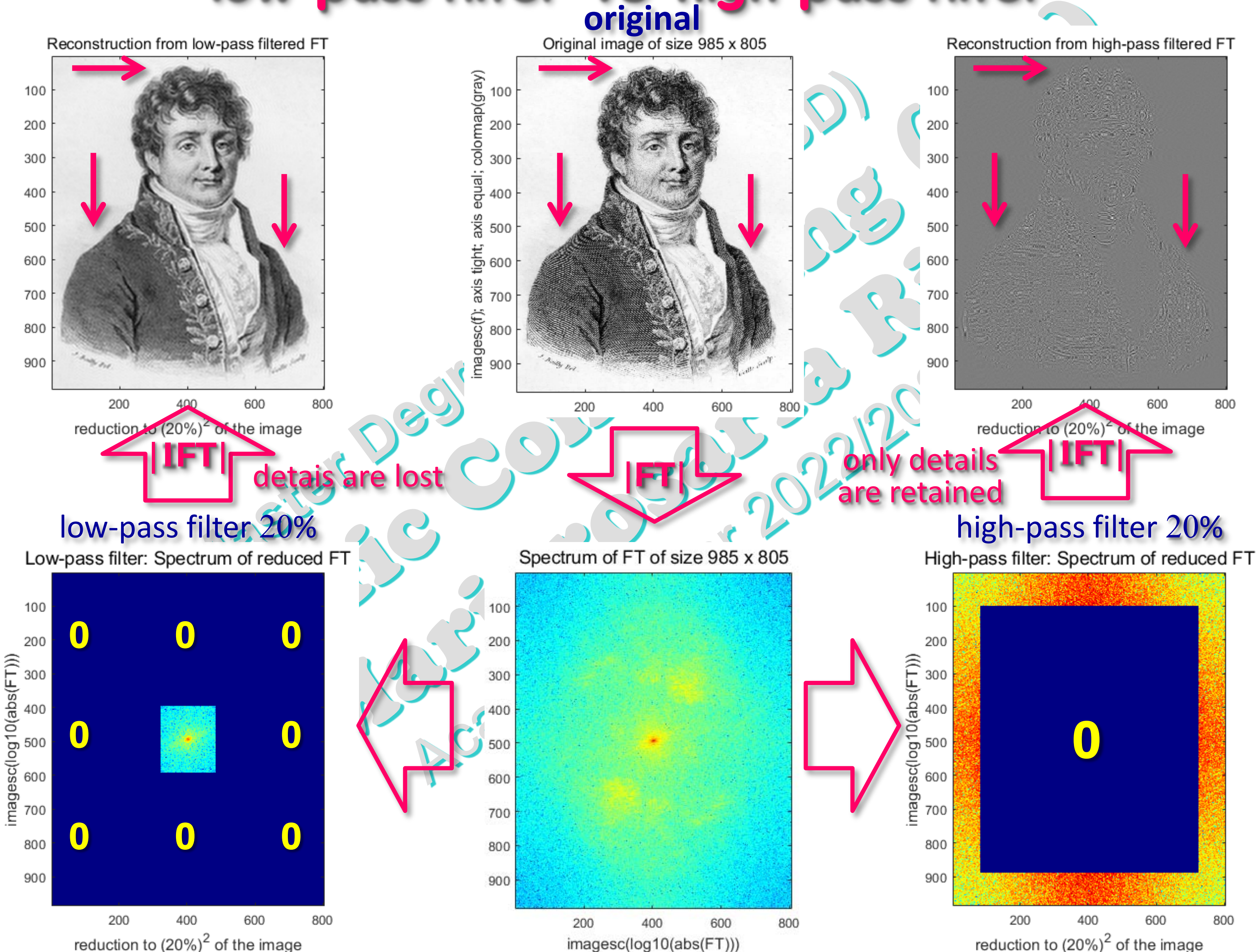
high-pass filter 50%



# low-pass filter VS high-pass filter



# Example: image compression

```
f= ... ;   F= ... ; nMid= ... ; mMid= ... ; perc=0.20;
Hn= ... ; Hm= ... ; i= ... ; j= ... ; FF=zeros(size(F));
FF=F(i,j); ff=fftshift(ifft2(FF,m,n)).*(-1).^(h+k);
figure; imagesc(real(f)); axis equal; colormap(gray); axis tight
...
figure; imagesc(real(ff)); axis equal; colormap(gray) ; axis tight
...
C=dct2(f);  CC=C(1:2*Hm+1,1:2*Hn+1);
cc=idct2(CC,m,n);
figure; imagesc(cc); axis equal; colormap(gray); ...
```

zero padding

2D DCT and reduced DCT

image reconstruction from reduced DCT

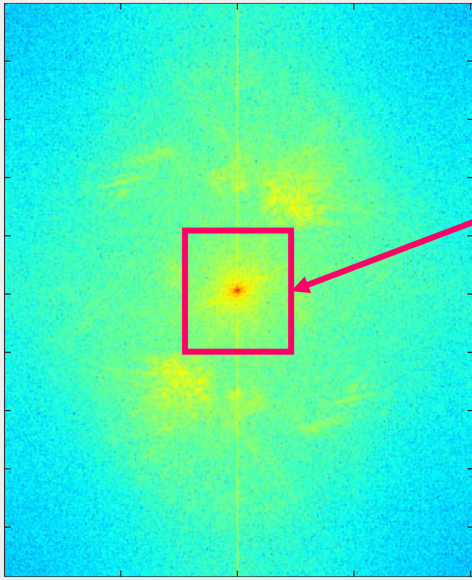
**DCT = Discrete Cosine Transform**

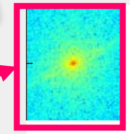
**used in jpeg compression**
(lossy data compression)

in MATLAB for 2D DCT:
**dct2()** and **idct2()**

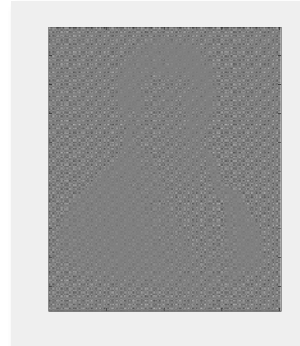
## Why is there no compression?

Fourier Transform





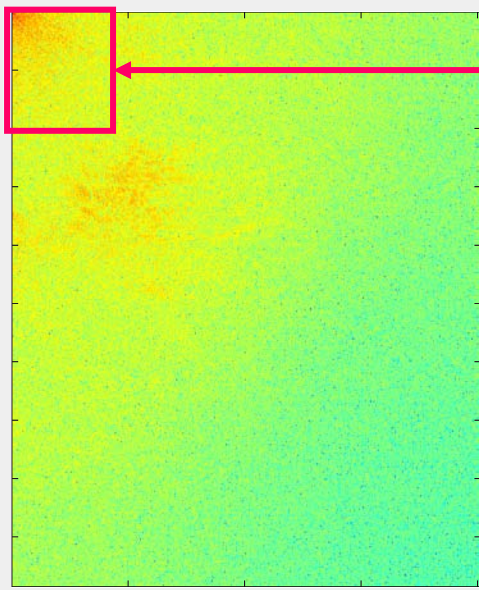
**It doesn't work!**

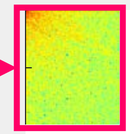


**perc=20%**

## Why compression?

Cosine Transform



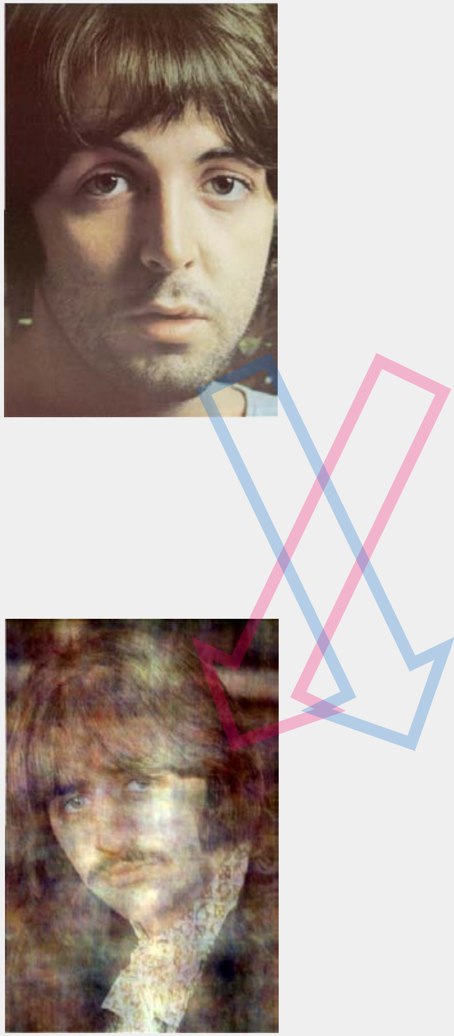


**It works!**

**Example:** FT returns, in general, complex values.

What is more important, between argument (phase angle) and modulus (magnitude) of a FT, in reconstructing an image from its FT?

```
fig1=imread('McCartney.jpg');
fig2=imread('Starr.jpg');
figure(1); clf
subplot(1,2,1); imshow(fig1)
subplot(1,2,2); imshow(fig2)
sgtitle('Original images')
%% mixing Fourier Transforms
Ffig1=fft2(double(fig1));
Ffig2=fft2(double(fig2));
G1=abs(Ffig1).*exp(1i*angle(Ffig2));
G2=abs(Ffig2).*exp(1i*angle(Ffig1));
%% invert new Fourier Transforms
g1=ifft2(G1);
g2=ifft2(G2);
subplot(1,2,1); imshow(uint8(real(g1)))
subplot(1,2,2); imshow(uint8(real(g2)))
```



The phase angle of a FT is predominant w.r.t. the modulus in reconstructing an image.

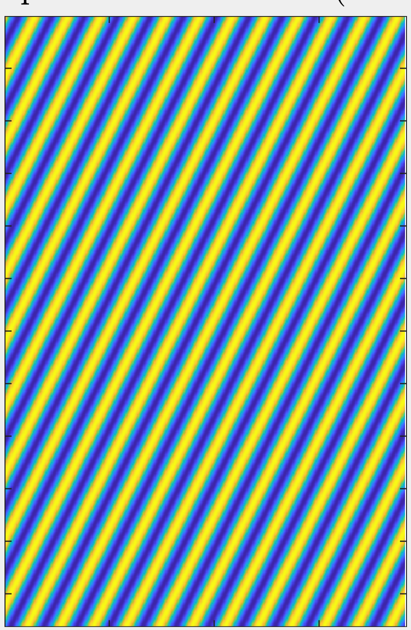
**Example:** let us introduce a periodic perturbation on the red component of an RGB image

```
f=imread('McCartney.jpg');
[m,n,q]=size(f); [X,Y]=meshgrid(1:n,1:m);
p=60*cos(.2*X+.1*Y); imagesc(p); axis tight; axis equal
pf=f; pf(:,:,1)=pf(:,:,1)+uint8(p); imagesc(pf); axis
equal
fred=uint8(zeros(size(f)));  fred(:,:,1)=f(:,:,1);
imagesc(fred); axis equal; axis tight
```

RGB image

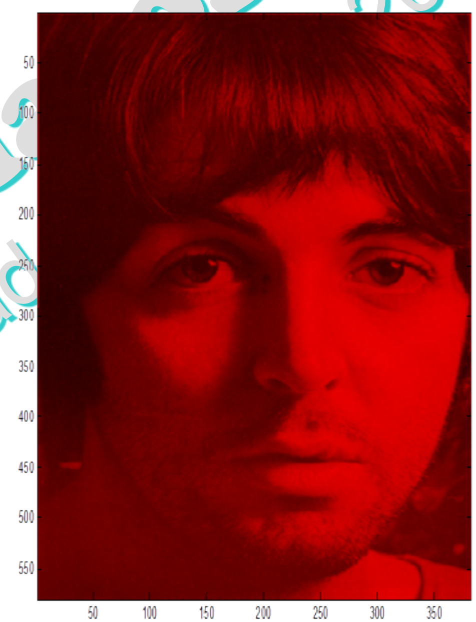
perturbation

Periodic perturbation $60 \times \cos(0.2x + 0.1y)$

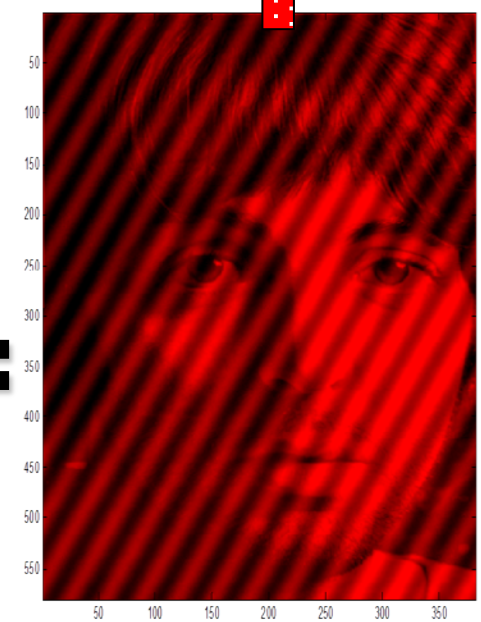


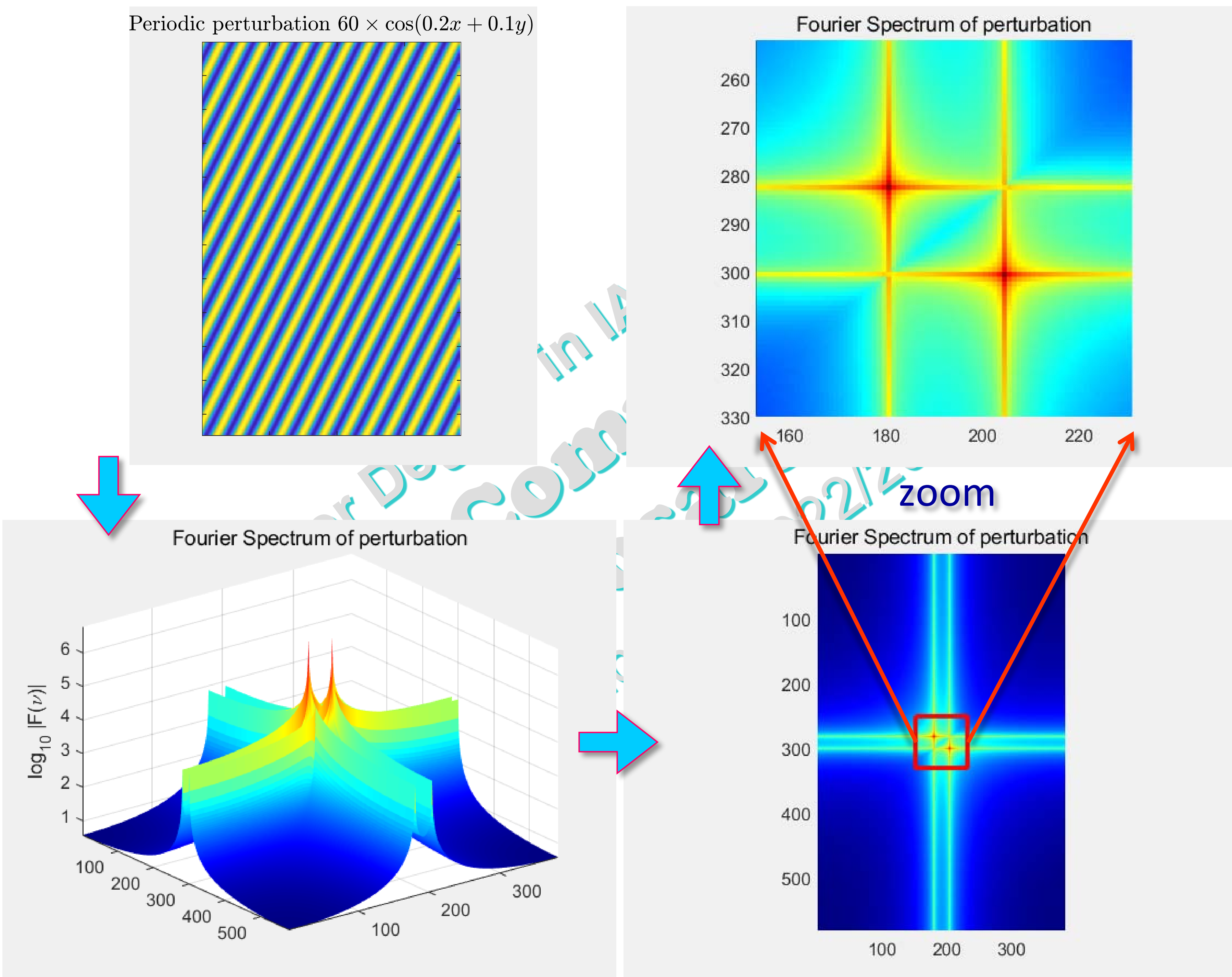
\+

red component



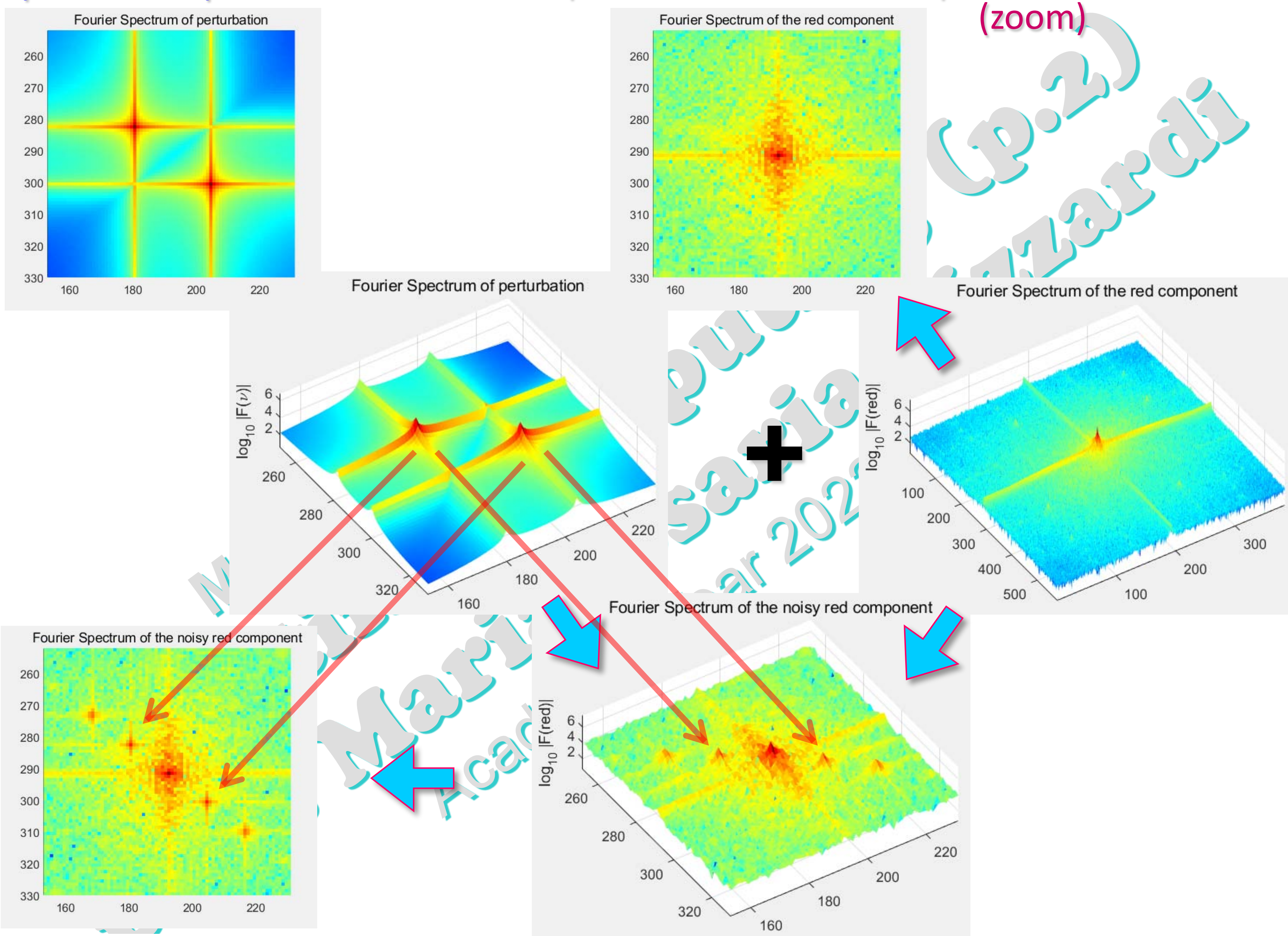
=




Periodic perturbation $60 \times \cos(0.2x + 0.1y)$
Fourier Spectrum of perturbation
260
270
280
290
300
310
320
330
160
180
200
220
zoom
Fourier Spectrum of perturbation
$\log_{10} |F(\nu)|$
6
5
4
3
2
1
100
200
300
400
500
100
200
300
Fourier Spectrum of perturbation
100
200
300
400
500
100
200
300

## Spectrum of perturbation

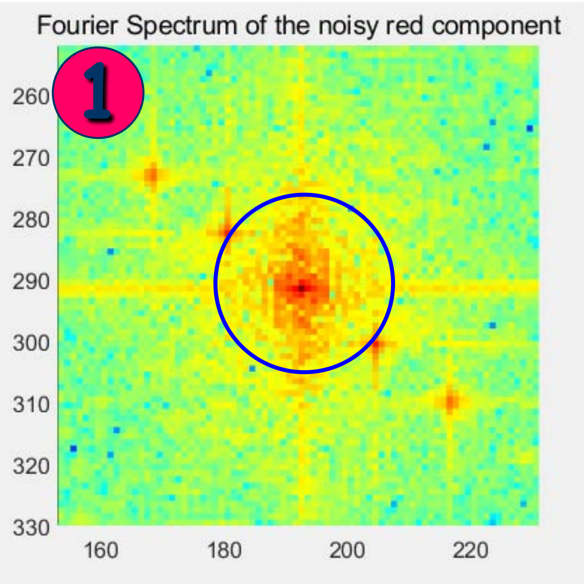
## Spectrum of the red component (zoom)

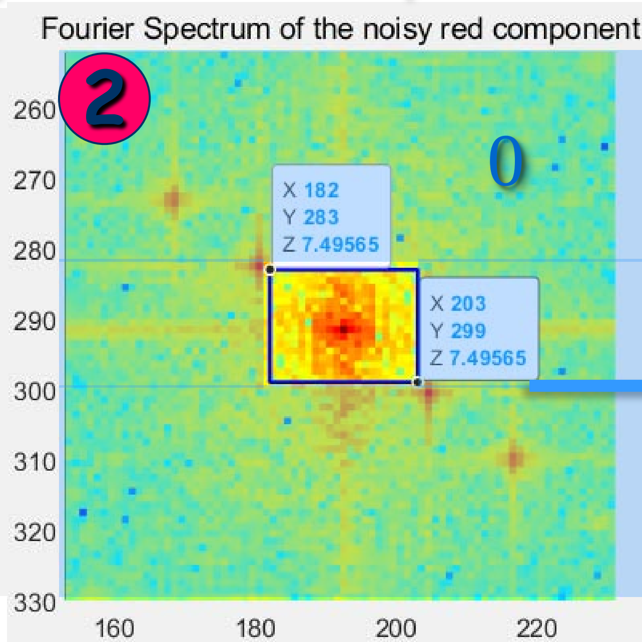


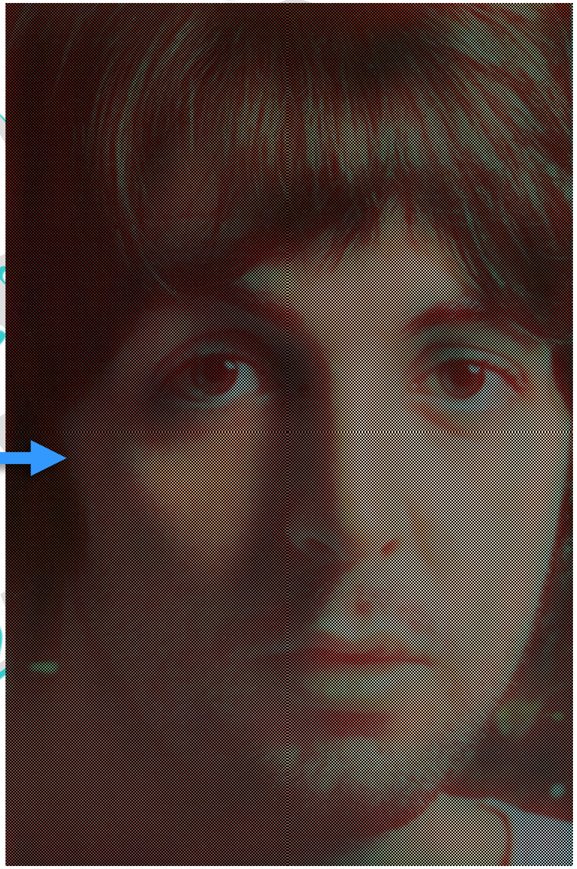

Spectrum of the perturbed red component

# To remove perturbations we can filter the FT in frequencies ...

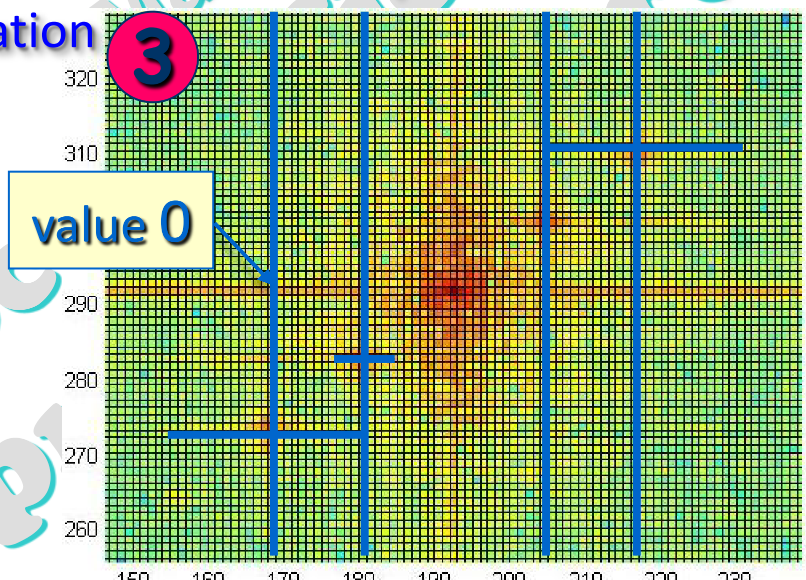
mantain only the central part of the FT: low-pass filter



resets to zero only the frequencies corresponding to the perturbation




**Exercise**

Implement in MATLAB such filters, and compare their results