Obrazové Transformácie

Cvičenia z Počítačového Videnia

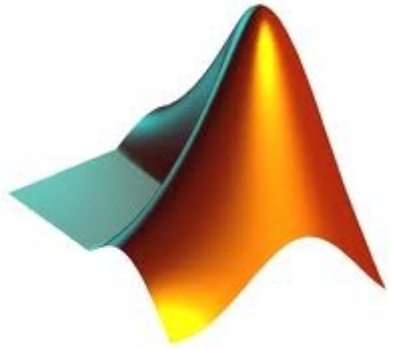
Zuzana Haladová
Júlia Kučerová

Obrazové transformácie

Zovšeobecnená Houghova transformácia

Rýchla Fourierova transformácia

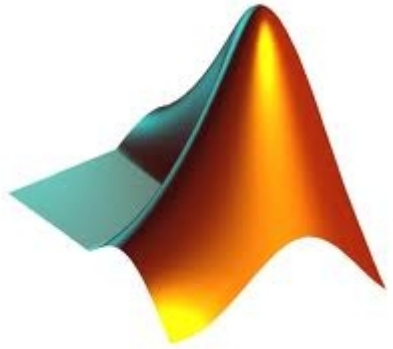
Diskrétna kosínusová transformácia



Hough

Houghova transformácia- ak chceme detekovať oblasti zo známym tvarom hranice

- Vieme detekovať priamky aj krivky, ak je známe ich analytické vyjadrenie
- Metóda je robustná aj pre zašumené objekty



Hough

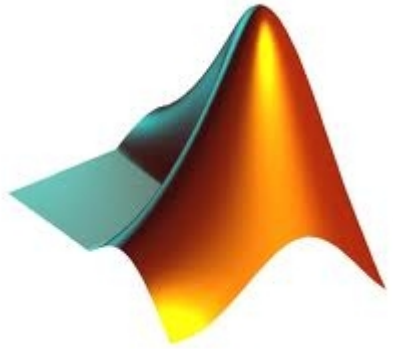
Houghova transformácia pre priamky

- Priamku vieme vyjadriť

$$y = mx + b$$

- Houghova trans. reprezentuje priamku pomocou b, m resp. r, θ pričom
- r je najmenšia vzdialenosť od počiatku k priamke

- θ je uhol medzi x a priamkou



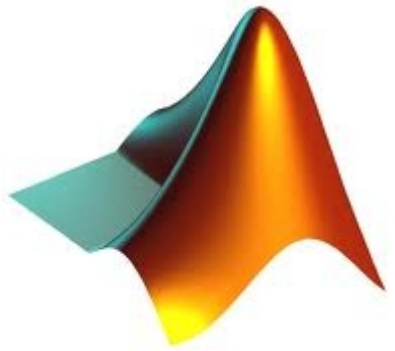
Hough

Vyjadrenie priamky pomocou r a θ je:

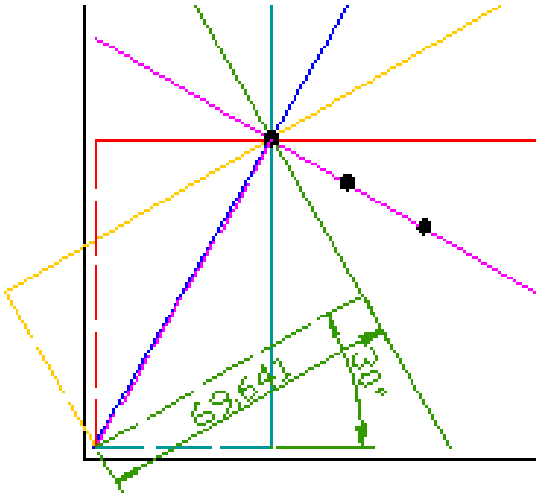
- HT pracuje na bin. obrázku

nasledovn $y = (-\frac{\cos\Theta}{\sin\Theta})x + (\frac{r}{sin\Theta})$

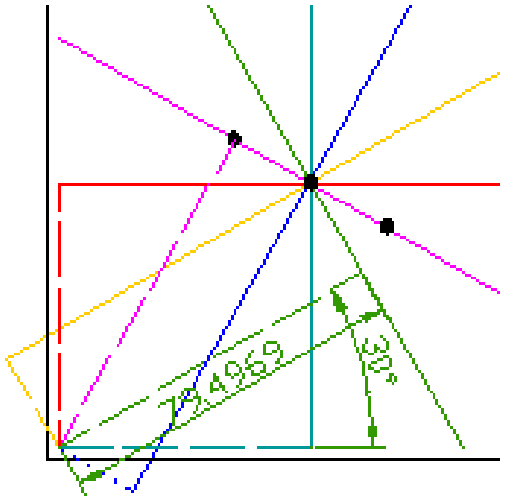
- Pre každý bod obrázku a pre každú priamku ním prechádzajúcu vypočíta r a θ tejto priamky.



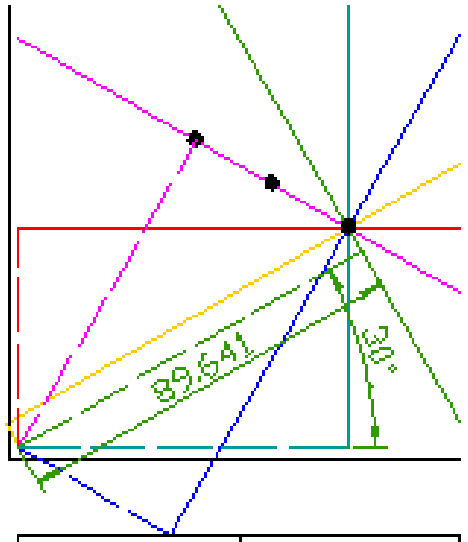
Hough



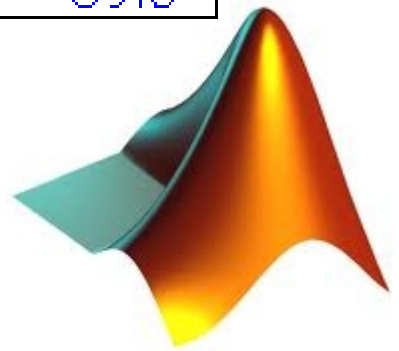
Angle	Dist.
0	40
30	69.6
60	81.2
90	70
120	40.6
150	0.4



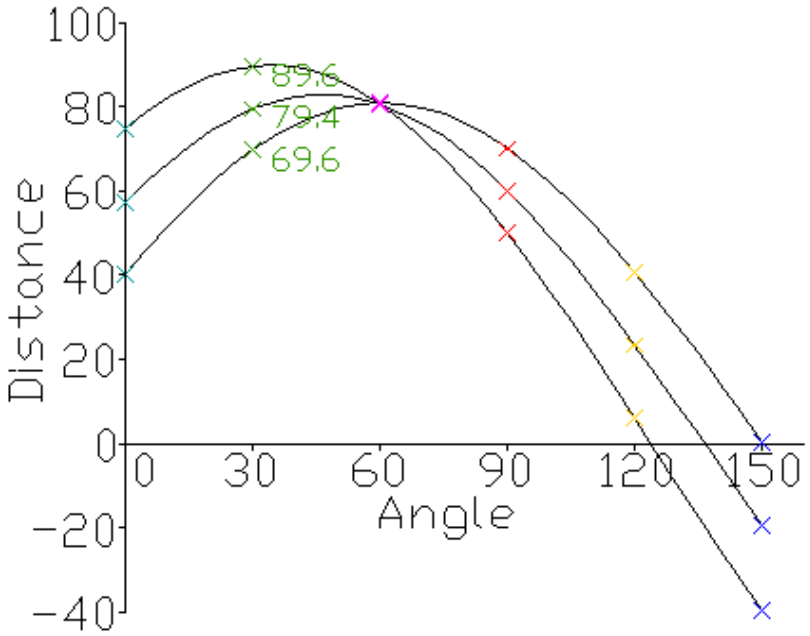
Angle	Dist.
0	57.1
30	79.5
60	80.5
90	60
120	23.4
150	-19.5

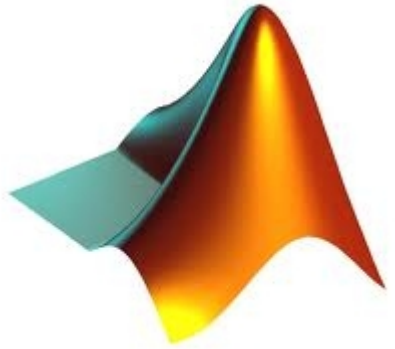


Angle	Dist.
0	74.6
30	89.6
60	80.6
90	50
120	6.0
150	-39.6

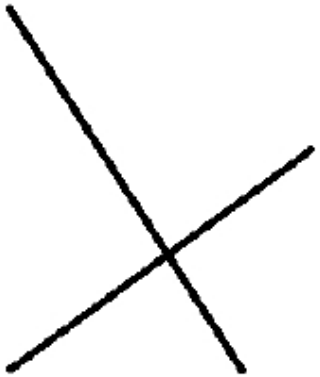


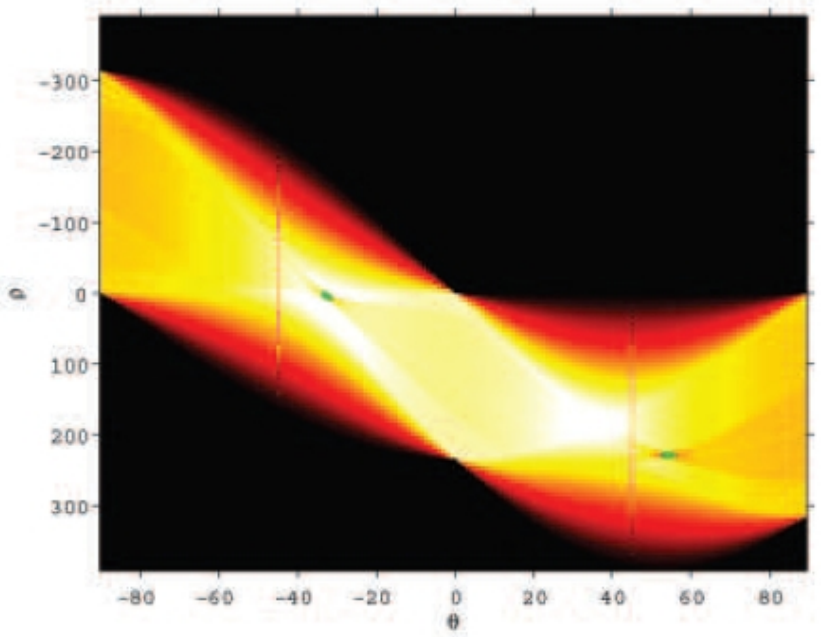
Hough

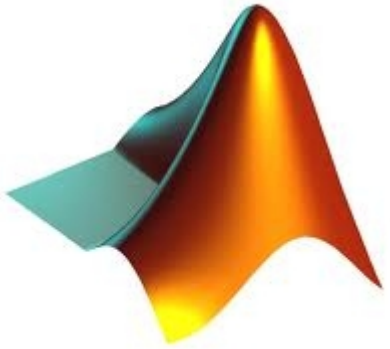

100
80
60
40
20
0
-20
-40
Distance
0
30
60
90
120
150
Angle
89.6
79.4
69.6



Hough



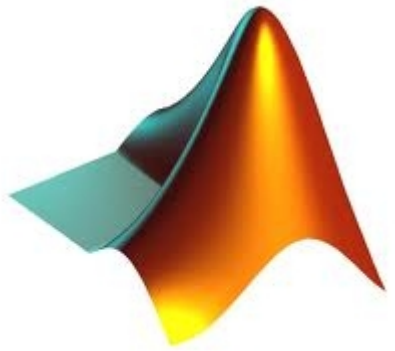




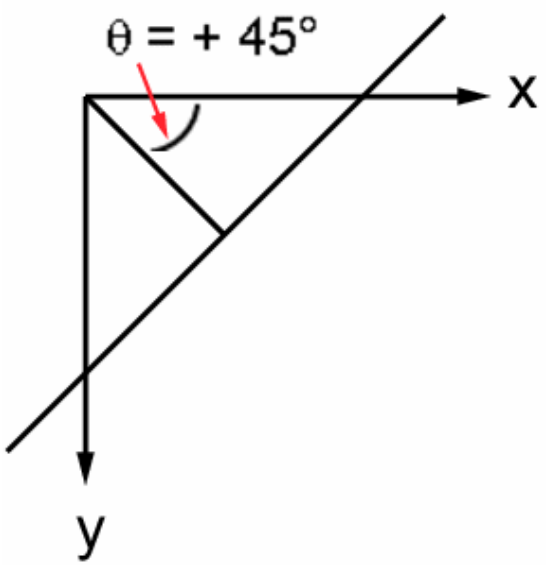
Hough

V MATLABE (Image Processing Toolbox)

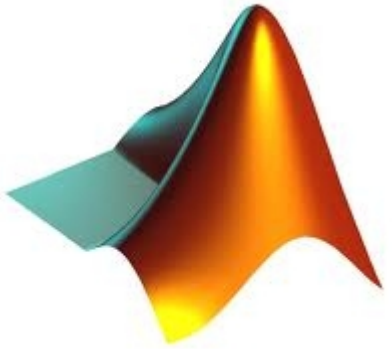
```
I=imread('1.jpg'); IG =rgb2gray(I);
IE=edge(IG,'Canny');
[H,T,R]=hough(IE);
imshow(imadjust(mat2gray(H)),'XData',T,'YData',
R); colormap(hot);
```



Hough



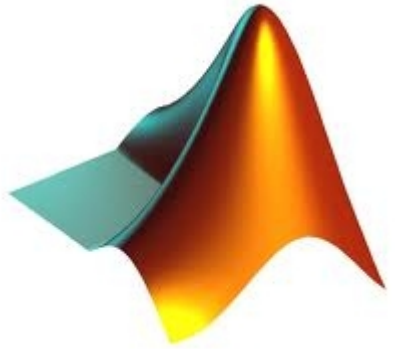
rho = x*cos(theta) + y*sin(theta)



Hough

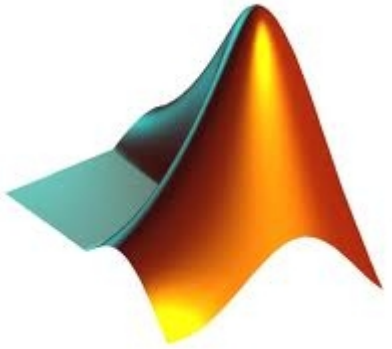
[H,T,R]=hough(IE);

- H – matica Houghovej transformácie
- T je pole hodnôt θ a R je pole r nad ktorými je vyjadrená H
- Riadky a stĺpce H zodpovedajú r a θ
- hodnota na pozícii (1,2) zodpovedá počtu bodov ležiacich na priamke z 1. r a 2. θ z polí R a T



Hough

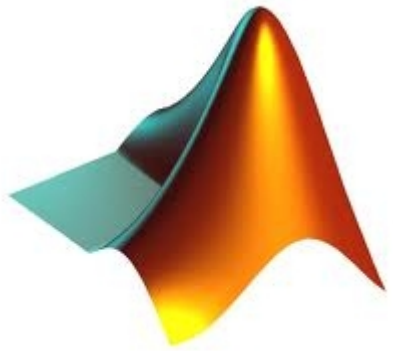
[H,T,R]=hough(IE);
- peaks = houghpeaks(H, numpeaks)
- peaks je pole Q x 2 kde Q <0,numpeaks>
a obsahuje x,y hodnoty pre peaks v H



Hough

lines = houghlines(BW, theta, rho, peaks)

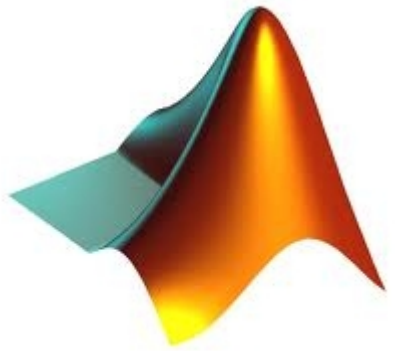
- lines je štruktúrované pole ktoré pre každú nájdenú úsečku obsahuje:
- point1 x,y hodnoty konca úsečky
- point2 x,y hodnoty konca úsečky
- theta uhol v stupňoch (z matice H)
- rho hodnota(z matice H)



Hough

lines = houghlines(IE, theta, rho, peaks)

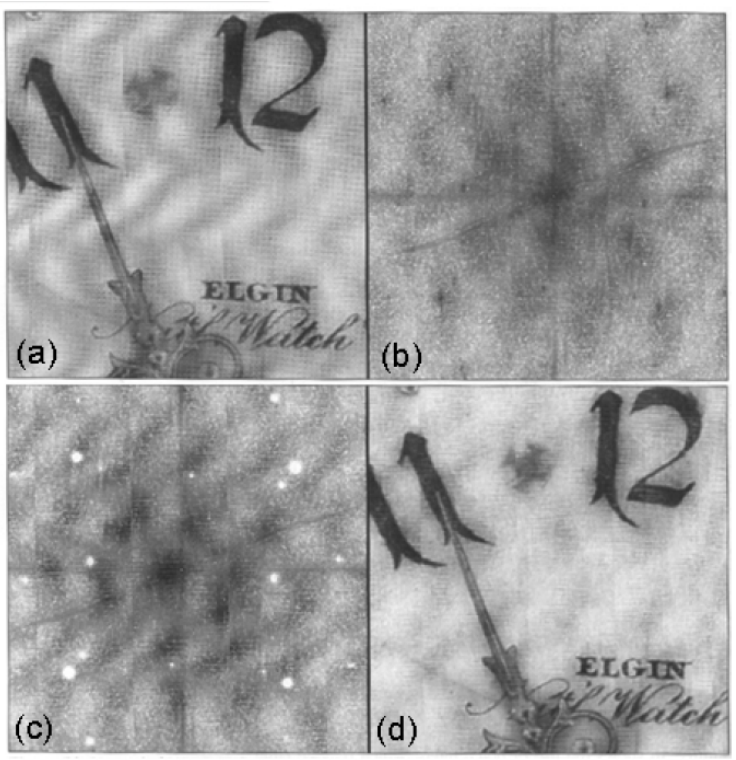
- figure, imshow(IE), hold on
- for k = 1:length(lines)
- xy = [lines(k).point1; lines(k).point2];
- plot(xy(:,1),xy(:,2),'Color','green');
- end

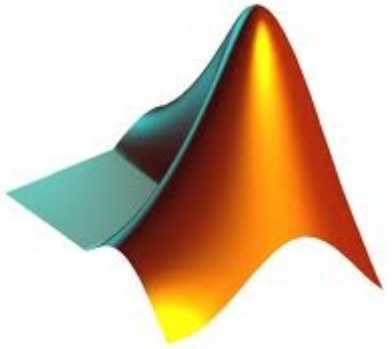


FFT

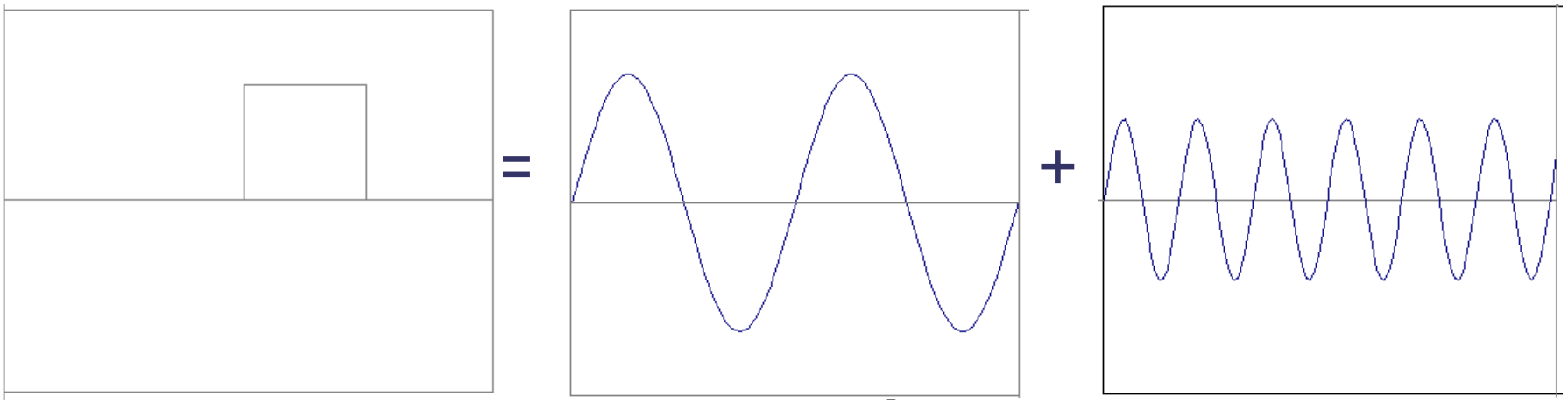
Akákoľvek funkcia f(x) môže byť vyjadrená ako vážený súčet sínusov a kosínusov

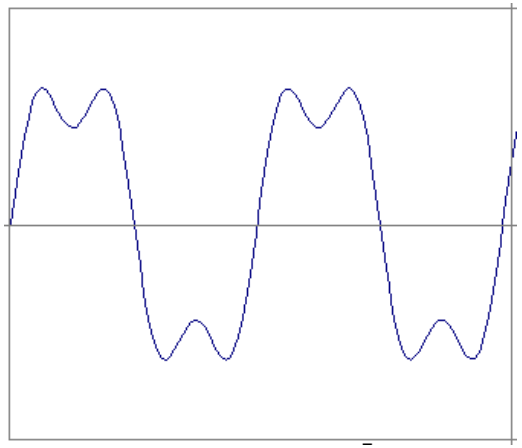
Využitie:
- lowpass filter
- highpass filter
- odstraňovanie artefaktov z obrazu

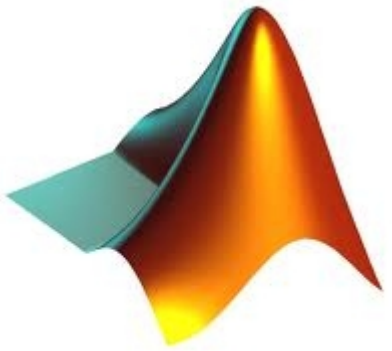




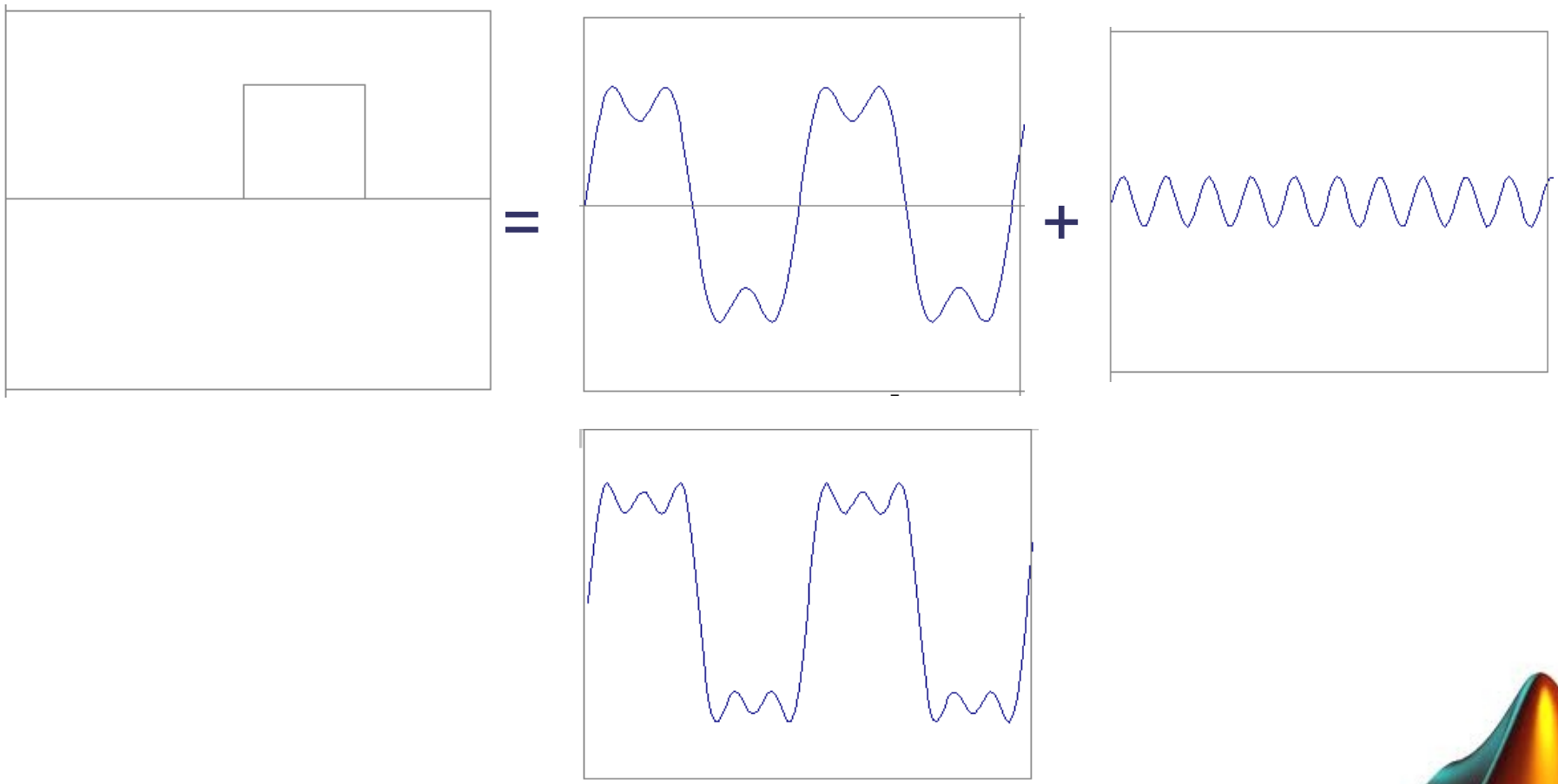
FFT

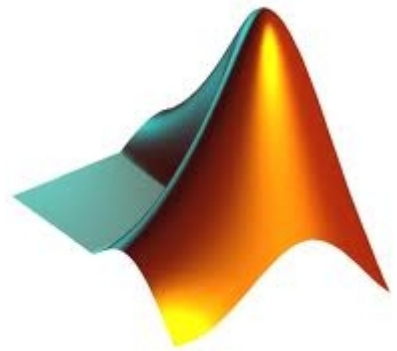




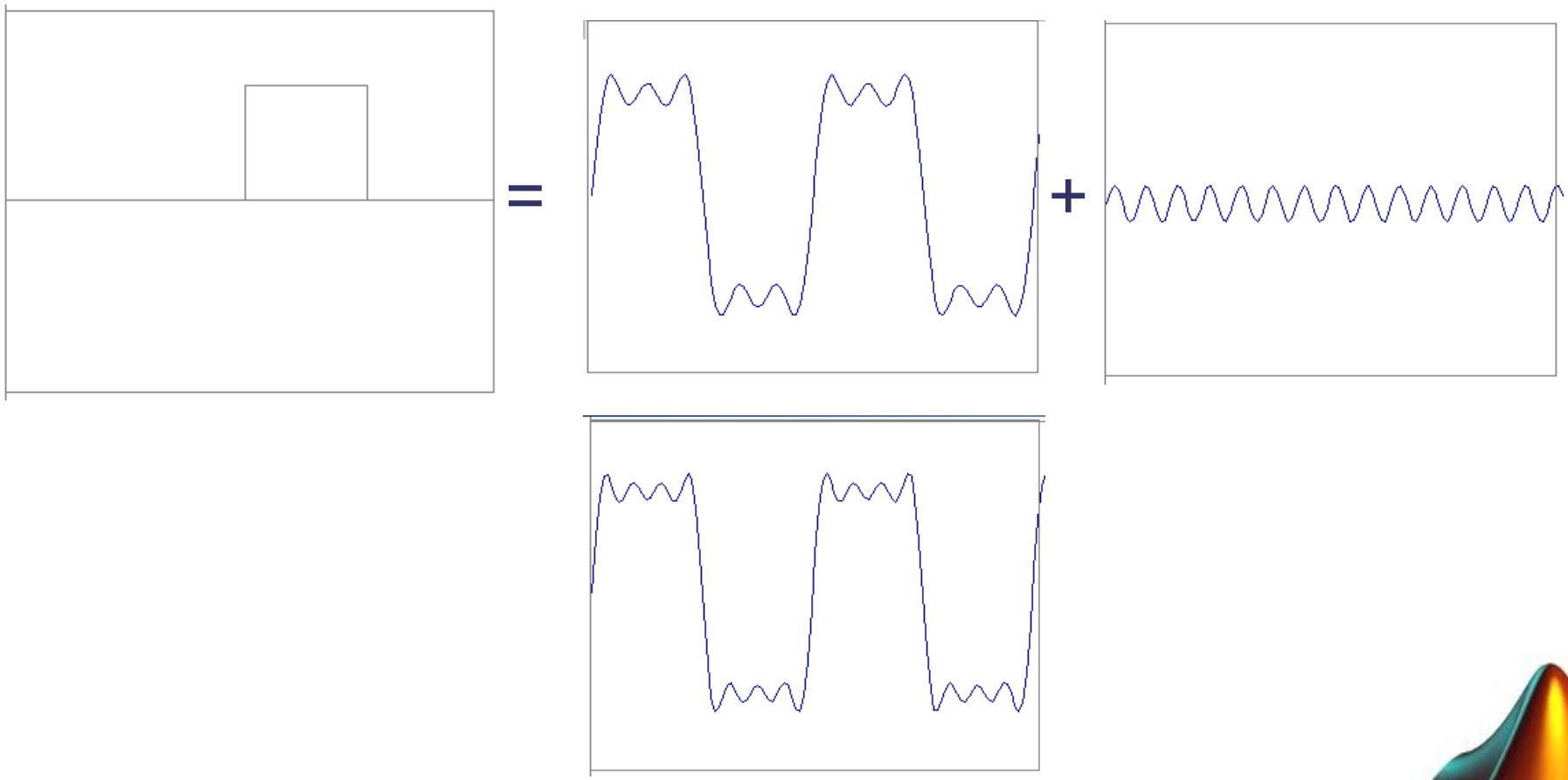


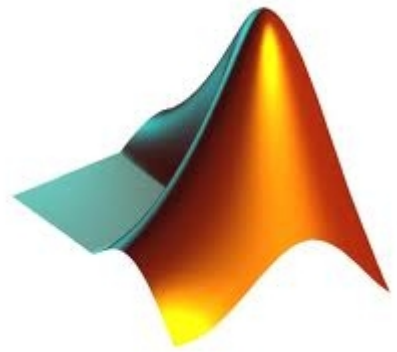
FFT



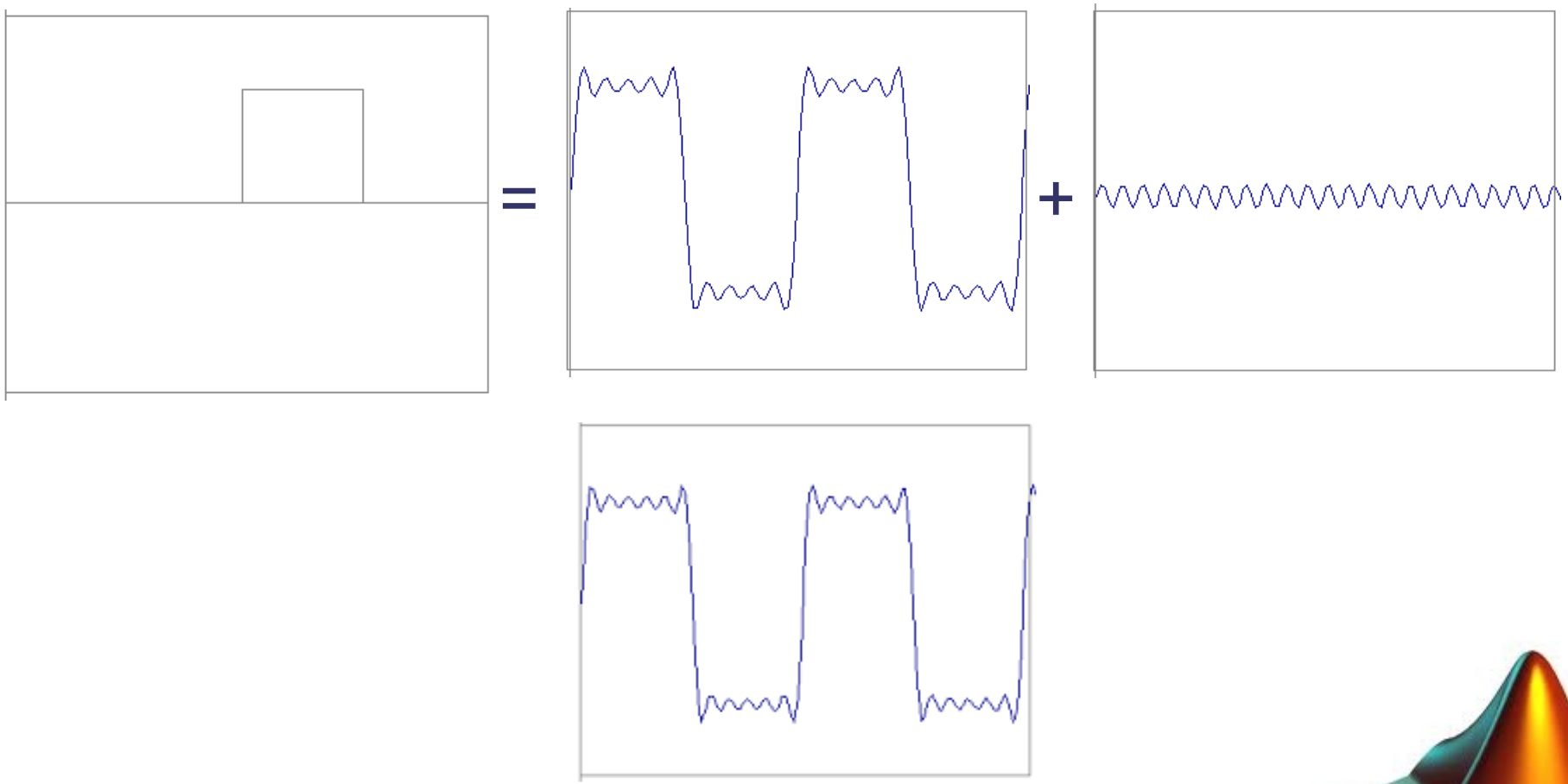


FFT

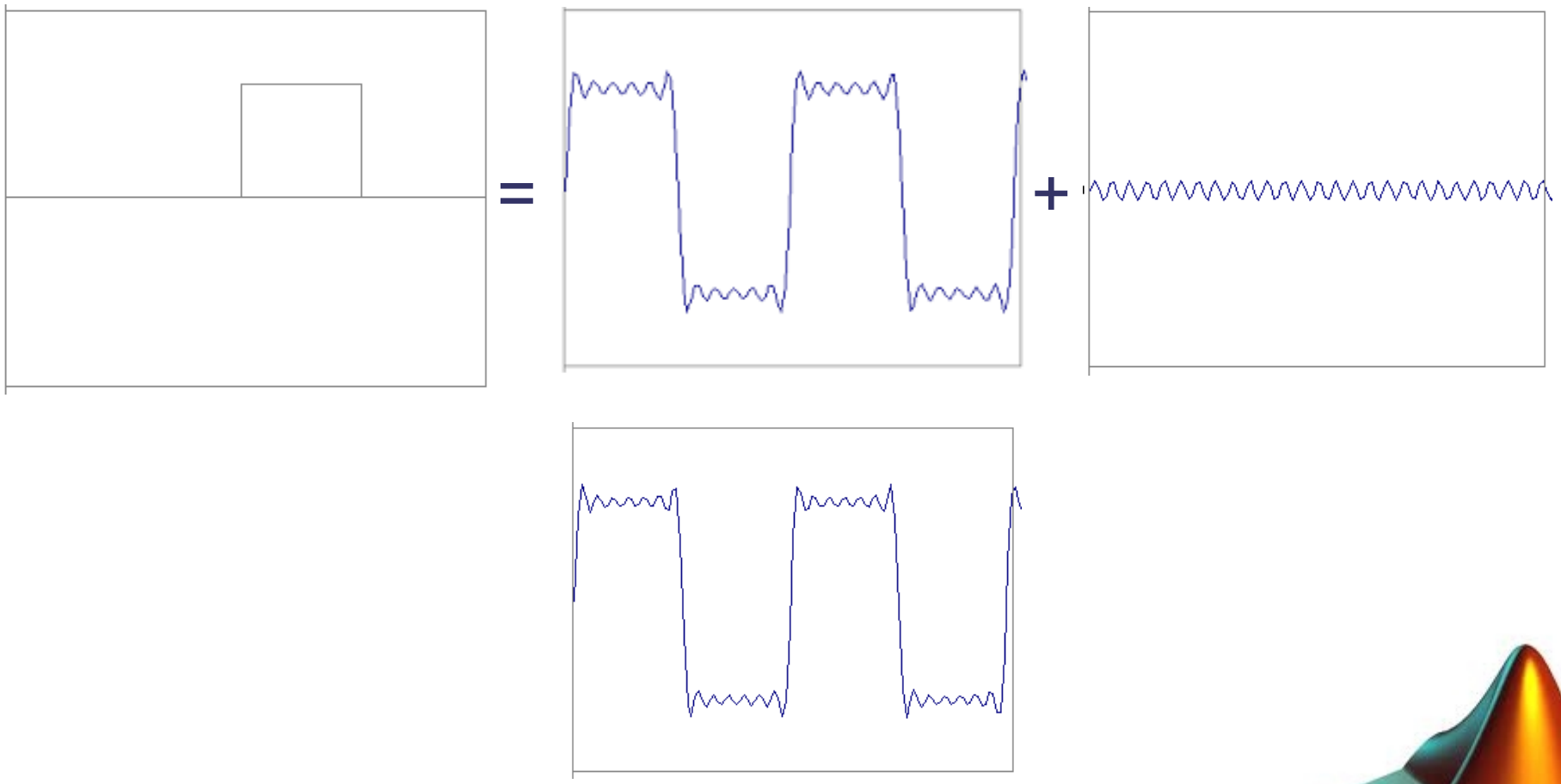


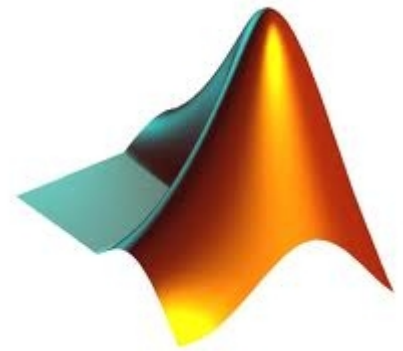


FFT

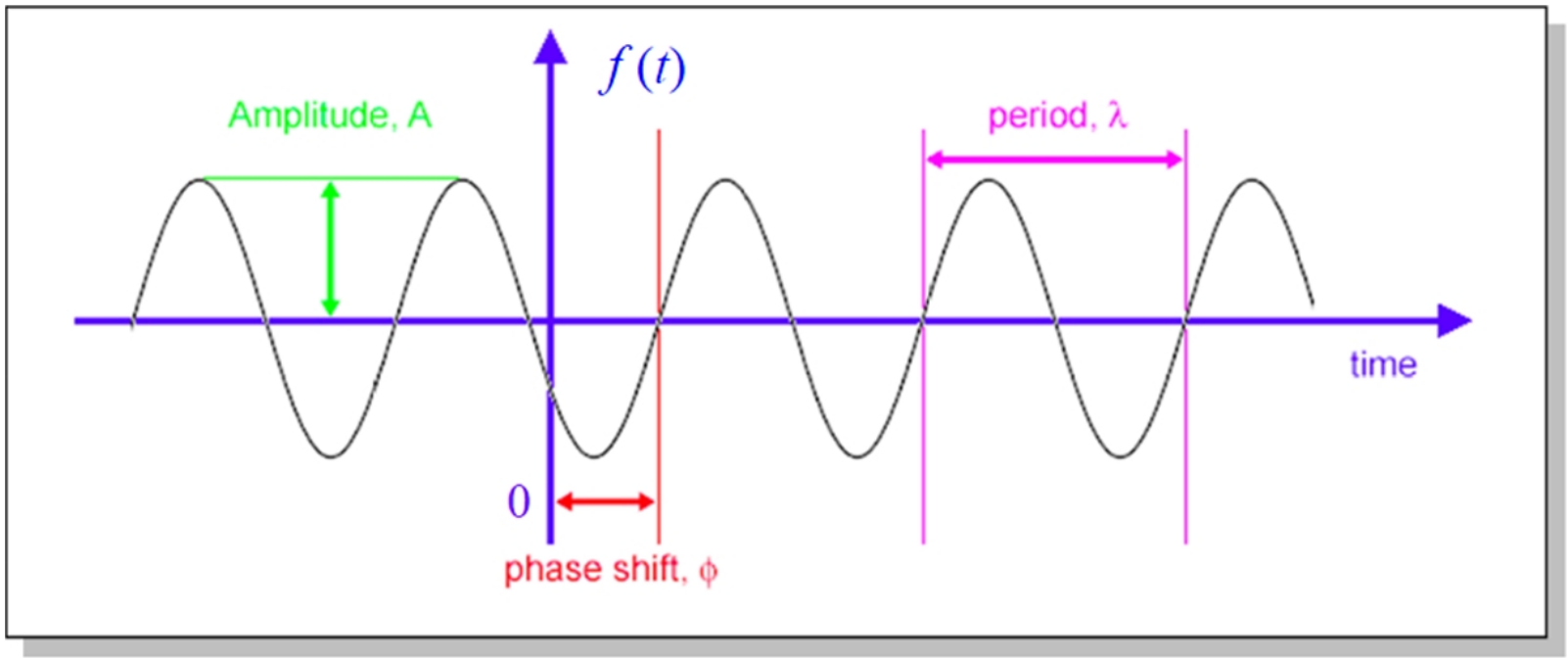


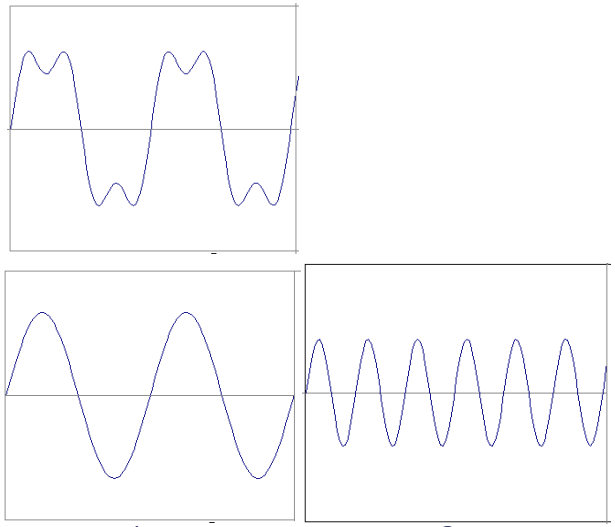
FFT

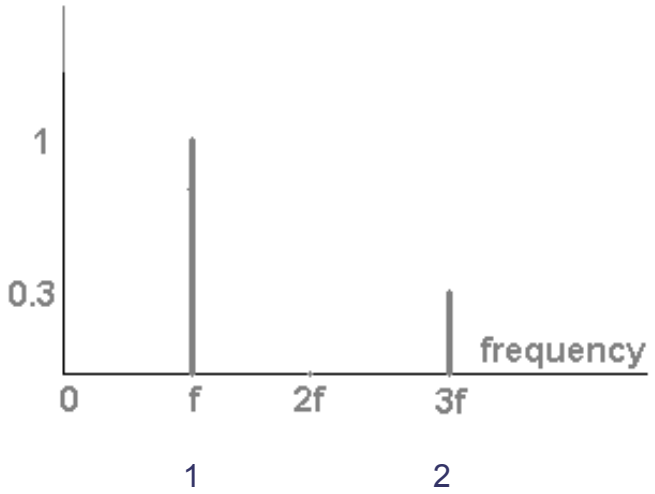


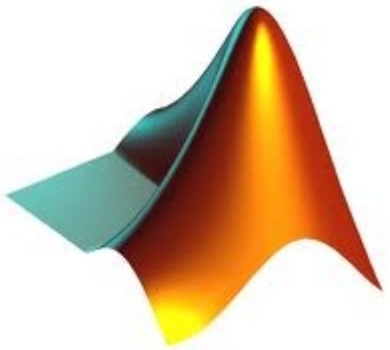


FFT


f (t)
Amplitude, A
period, λ
time
0
phase shift, ϕ

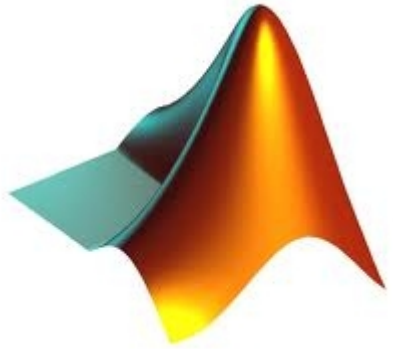

1
2


1
0.3
frequency
0
f
2f
3f
1
2



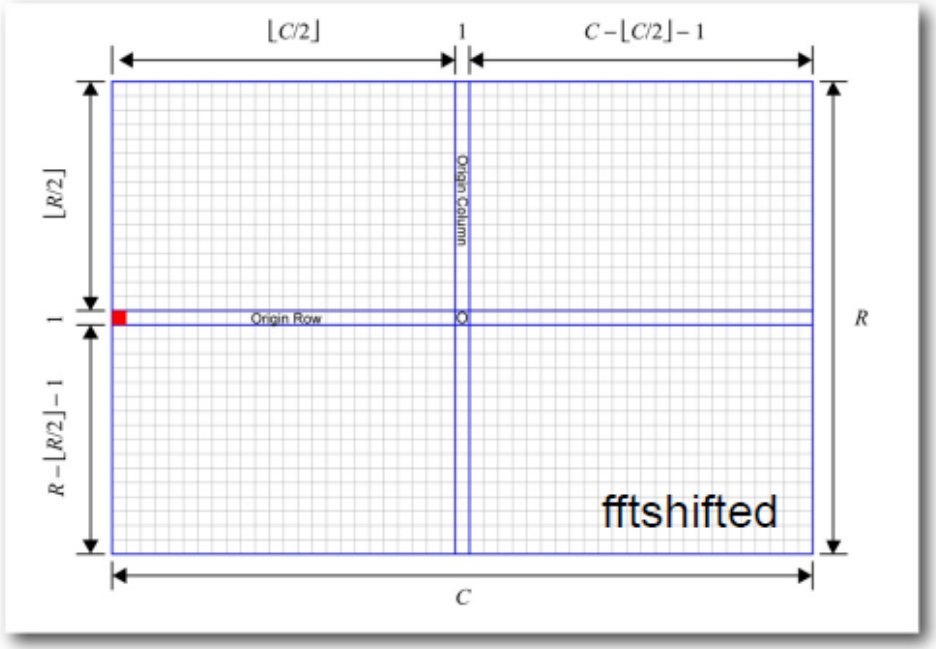
FFT – Skladanie signálu MATLAB

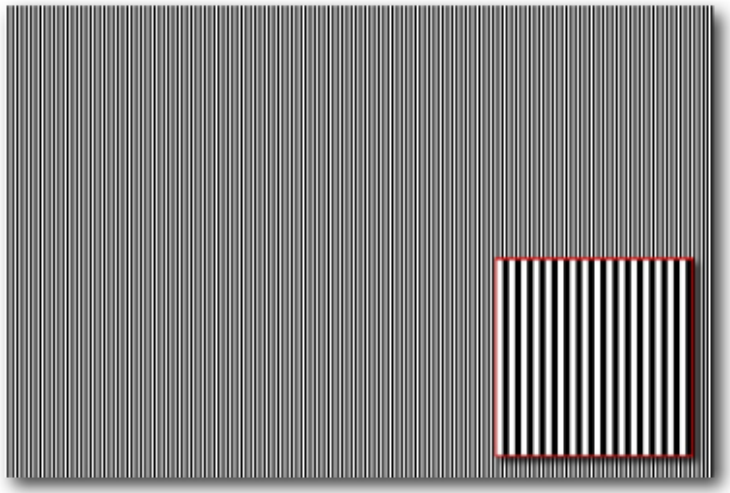
```
Fs = 1000;                    % Sampling frequency
T = 1/Fs;                     % Sample time
L = 1000;                     % Length of signal
t = (0:L-1)*T;                % Time vector
y1 = 0.9*sin(2*pi*10*t);
y2 = 0.3*sin(2*pi*30*t);
y3 = y1 +y2;
figure, plot(Fs*t(1:200),y1(1:200));
figure, plot(Fs*t(1:200),y2(1:200));
figure, plot(Fs*t(1:200),y3(1:200));
```



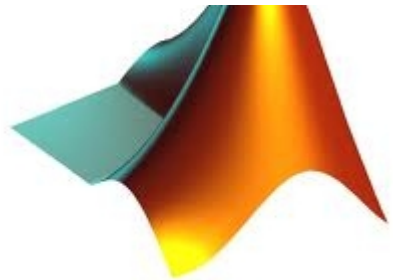
FFT

"horizontal" is the wavefront direction.



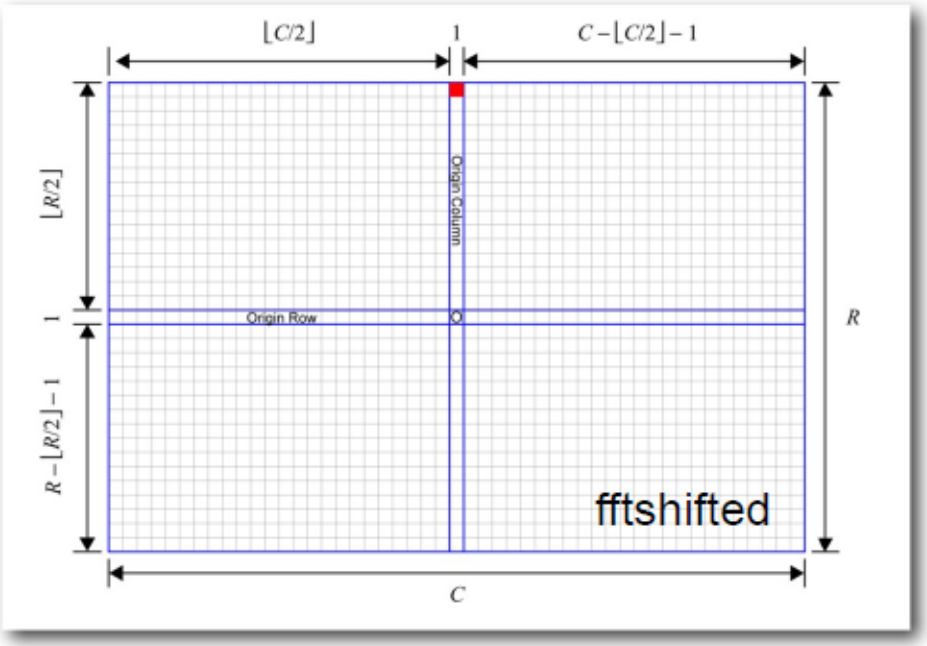


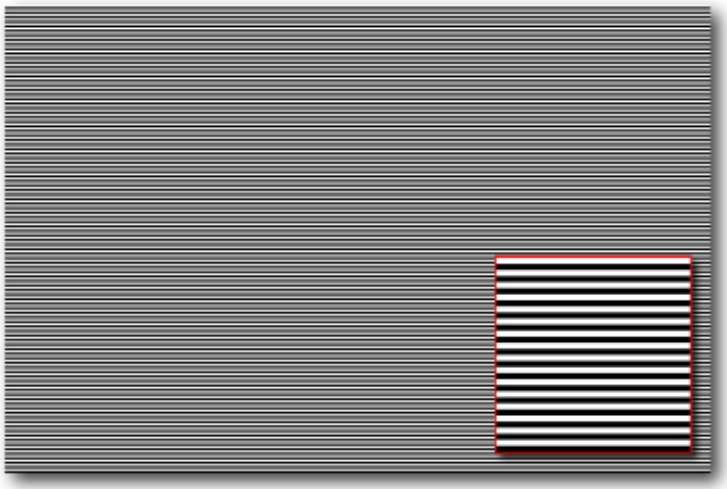
highest-possible-frequency horizontal sinusoid (C is even)



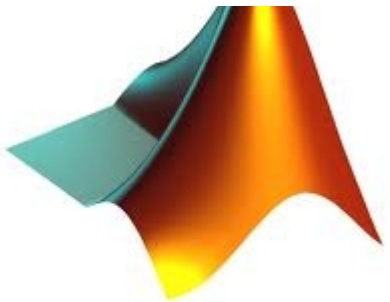
FFT

"vertical" is the wavefront direction.



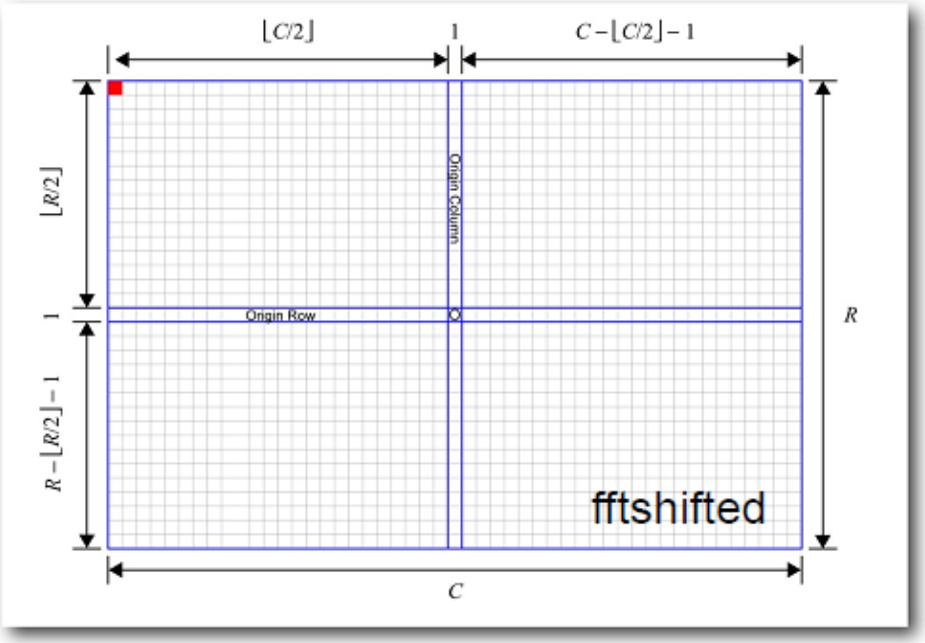


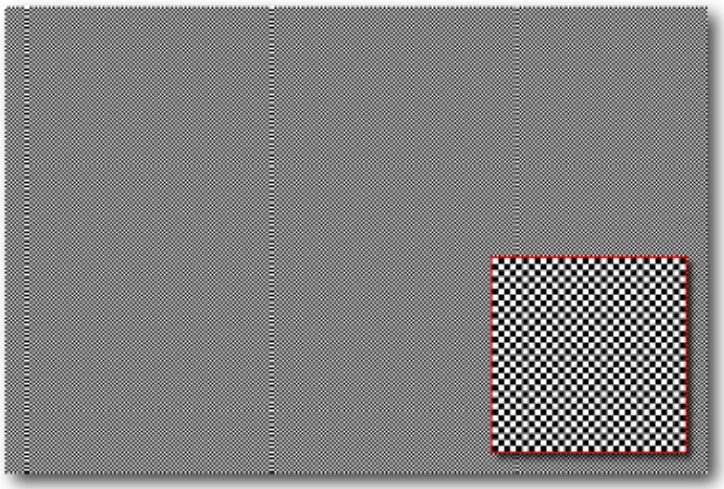
highest-possible-frequency vertical sinusoid (R is even)



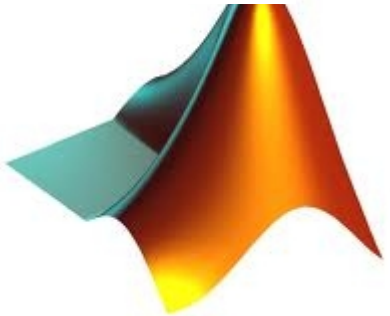
FFT

a checker-board pattern.



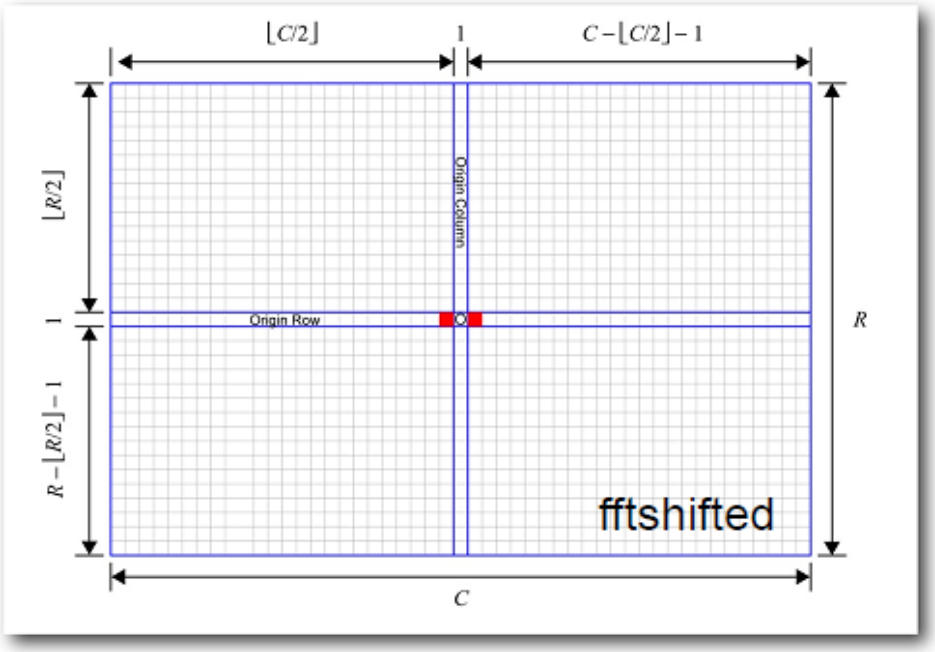


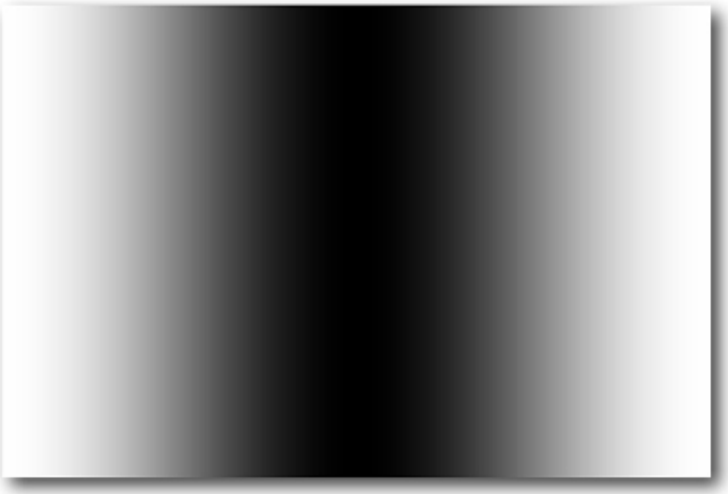
highest-possible-freq horizontal+vertical sinusoid (R & C even)



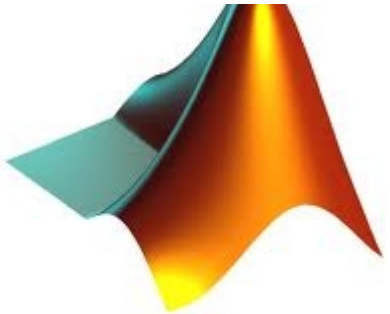
FFT

"horizontal" is the wavefront direction.



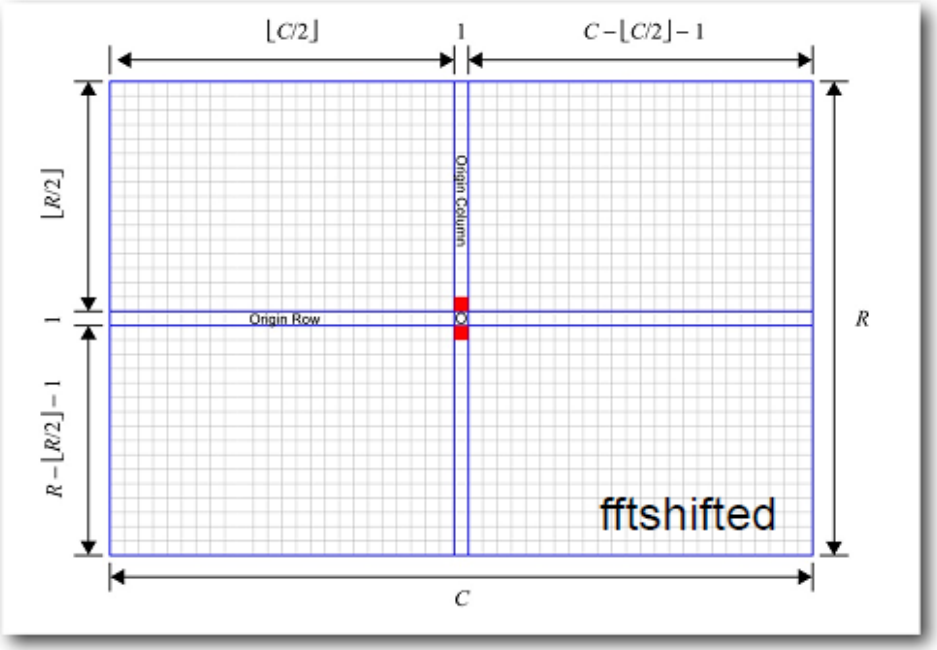


lowest-possible-frequency horizontal sinusoid

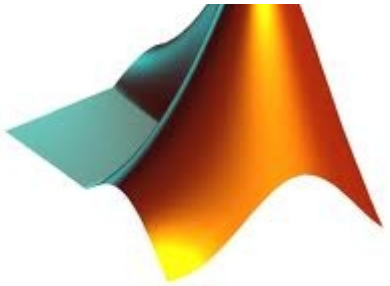


FFT

"vertical" is the wavefront direction.

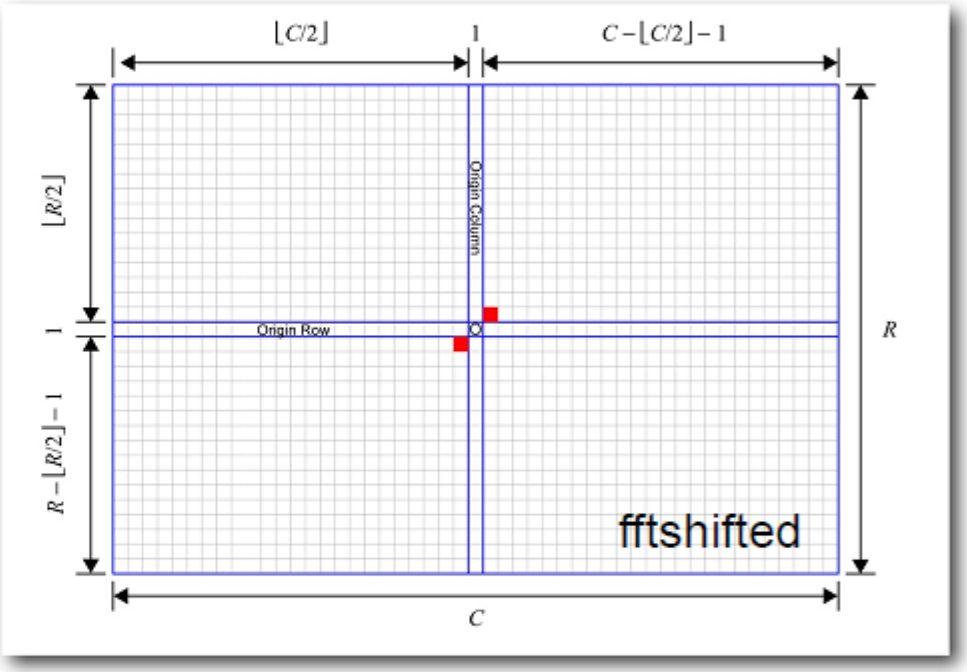


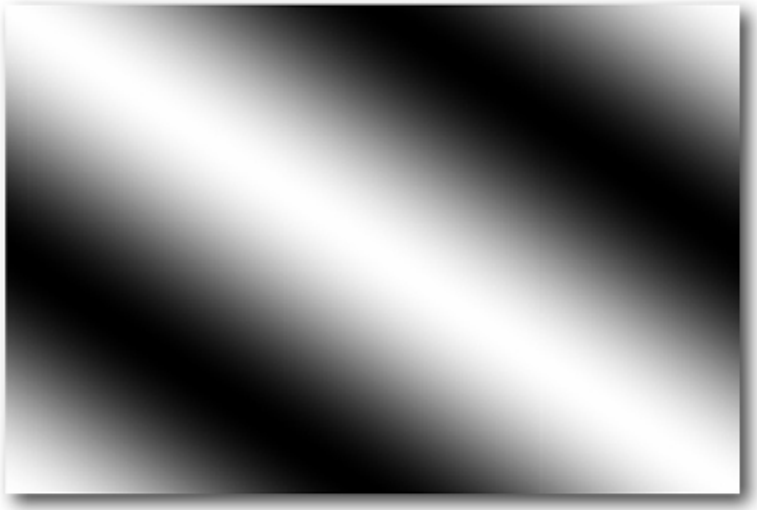
lowest-possible-frequency vertical sinusoid



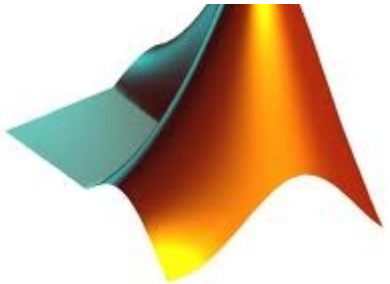
FFT

"negative diagonal" is the wavefront direction.

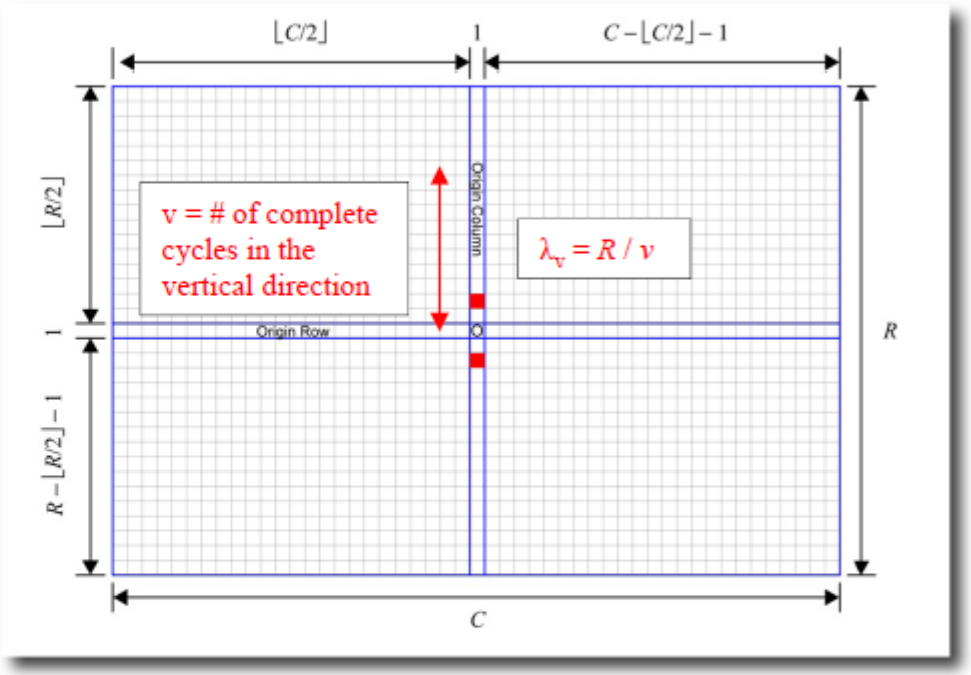


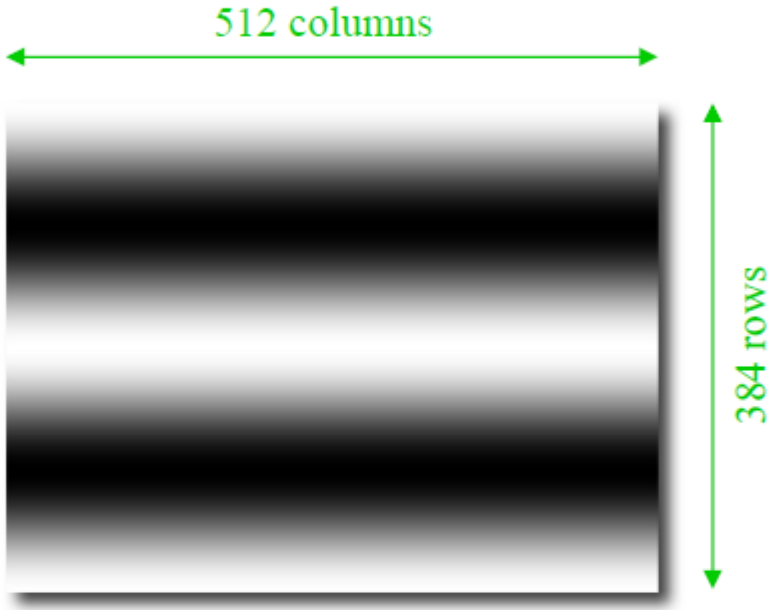


lowest-possible-frequency negative diagonal sinusoid

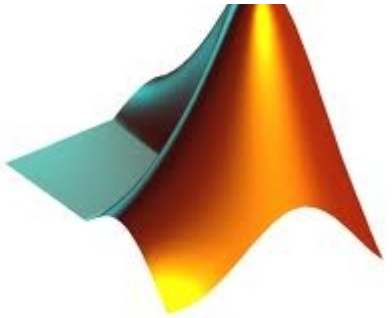


FFT

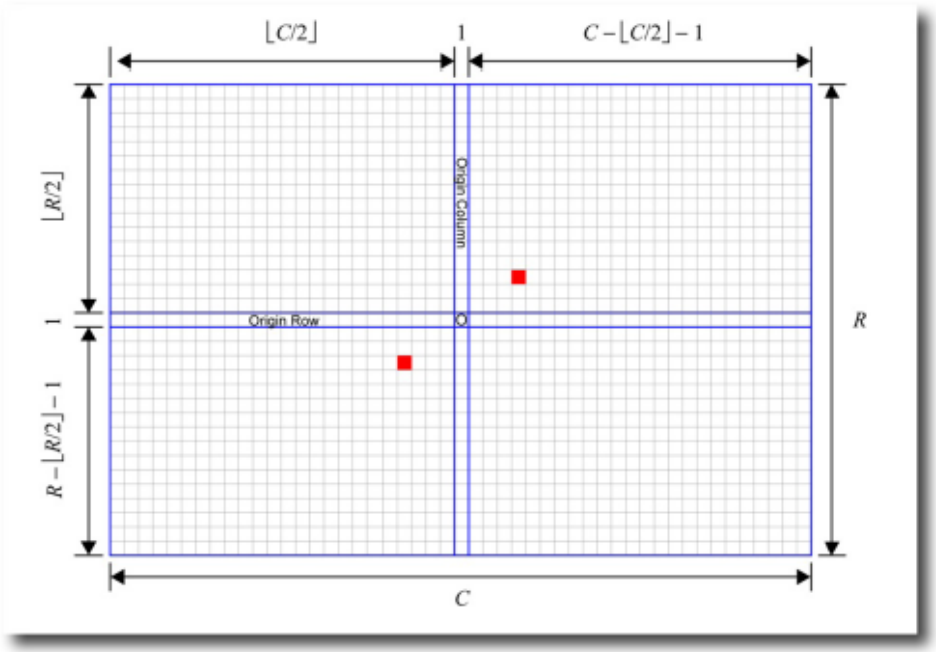


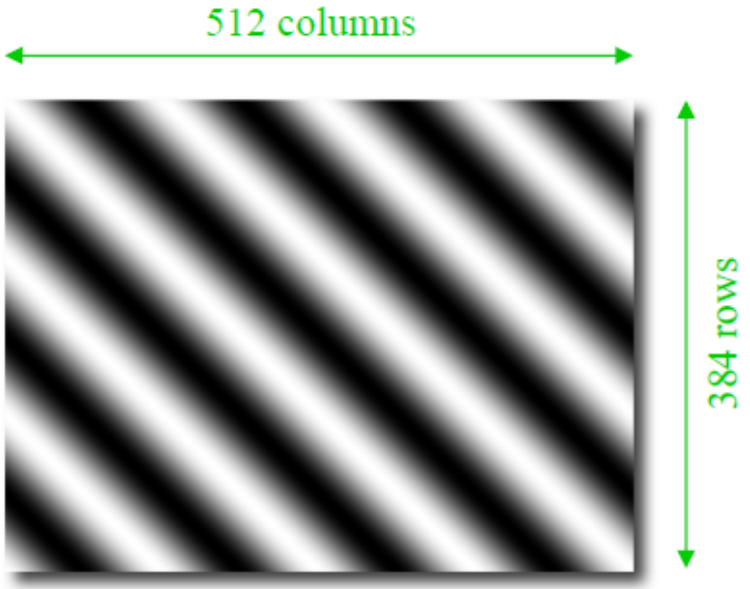


frequencies: $(u,v) = (0,2)$; wavelength: $\lambda_v = 192$

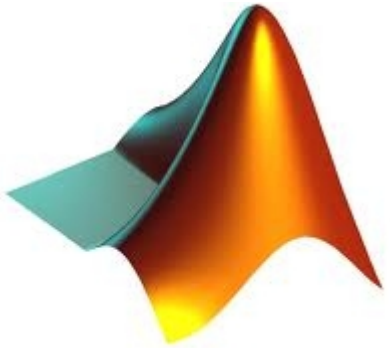


FFT





frequencies: $(u,v) = (4,3)$; wavelengths: $(\lambda_u, \lambda_v) = (128,128)$



FFT

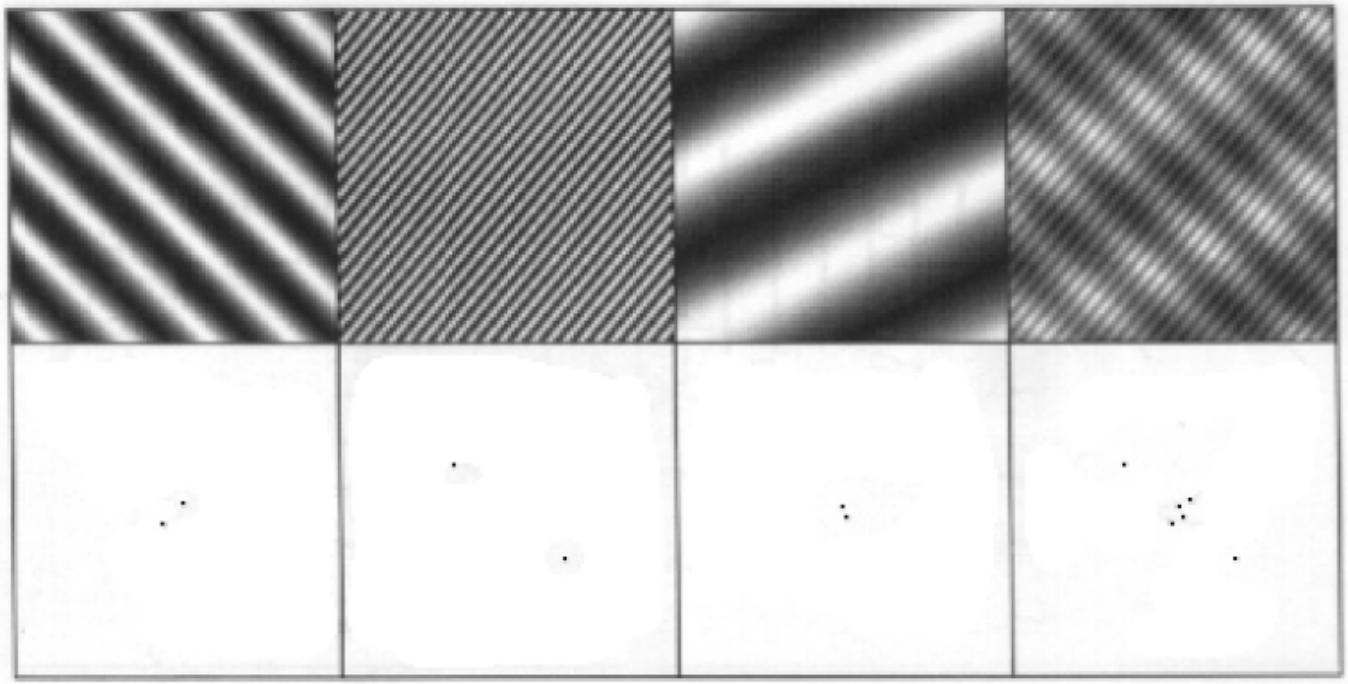
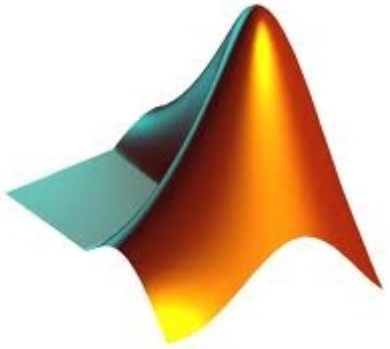
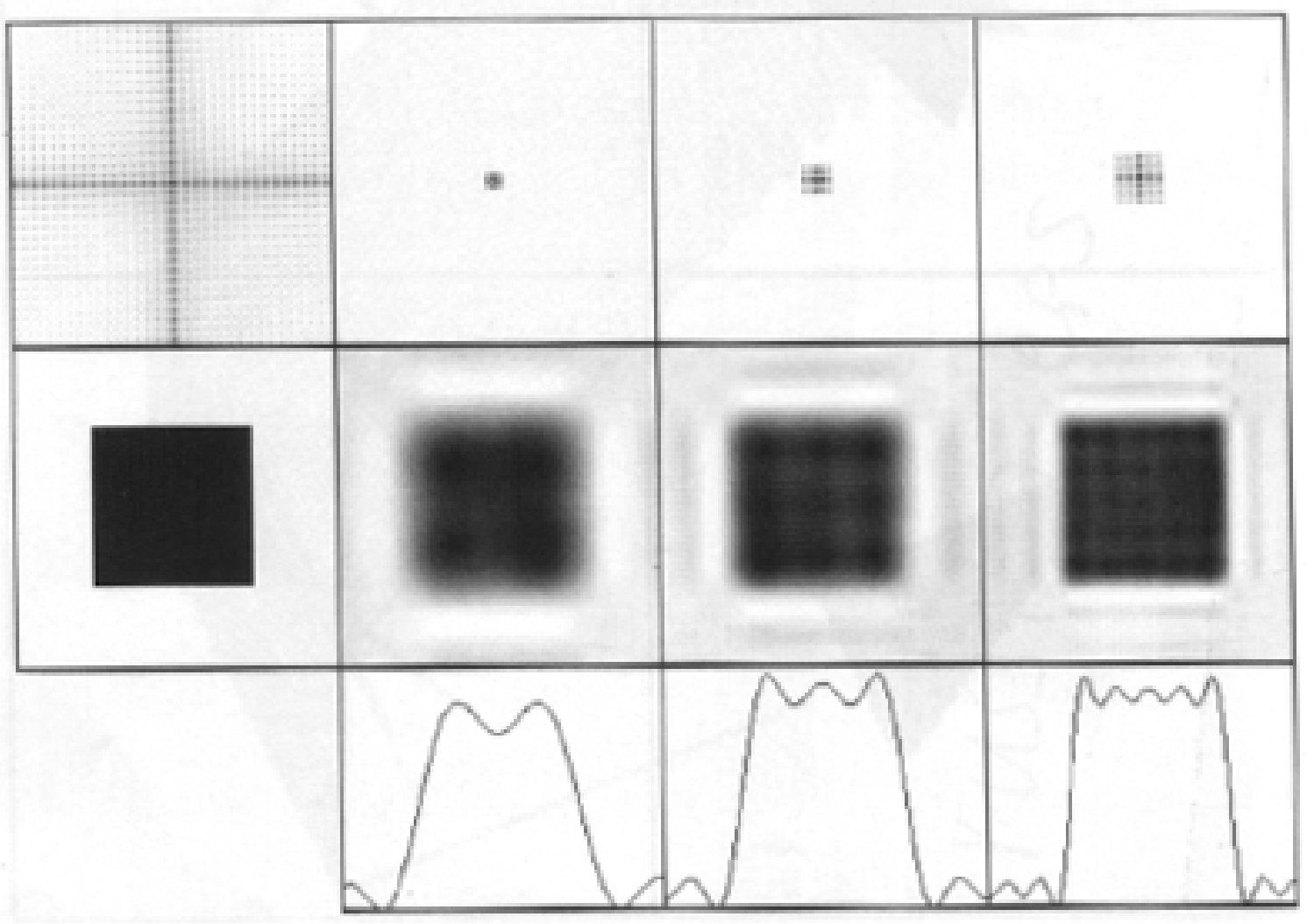
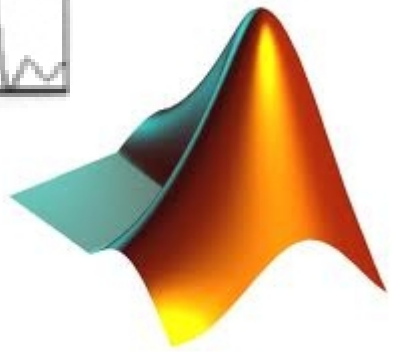


Figure 2: Images with perfectly sinusoidal variations in brightness: The first three images are represented by two dots. You can easily see that the position and orientation of those dots have something to do with what the original image looks like. The 4th image is the sum of the first three.
(Taken from p.177 of [1].)



FFT

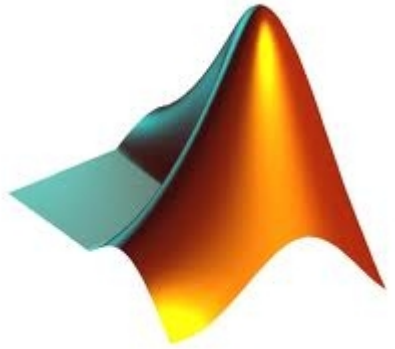




FFT MATLAB

2-D Fast Fourier transform

```
PS = fftshift(log(abs(fft2(I))+1));
M = max(PS(:));
image(uint8(255*(PS/M)));
```

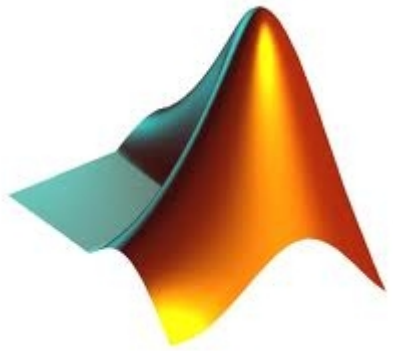


FFT ImageJ

- Image Processing and Analysis in Java

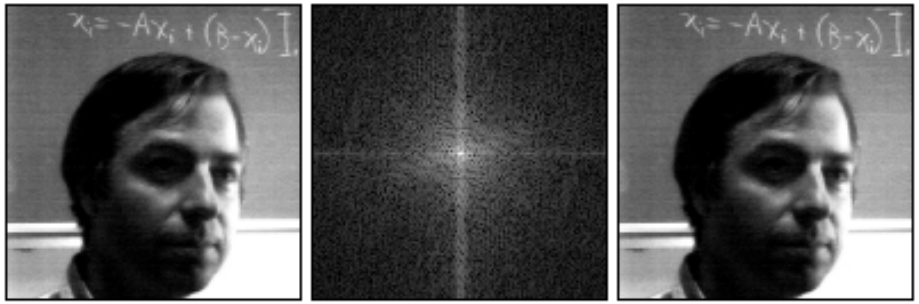
- Vytvorený v Jave – umožňuje ho spustiť na všetkých OS
- Open Source

- http://rsbweb.nih.gov/ij/

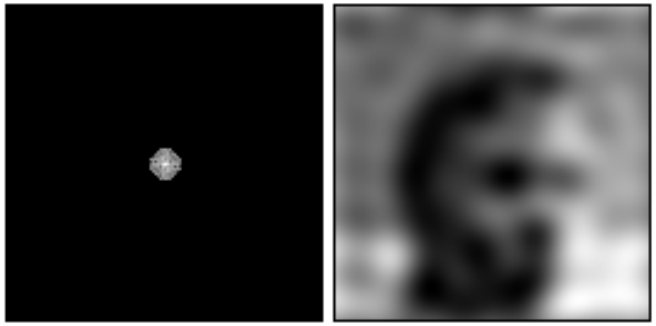


FFT ImageJ

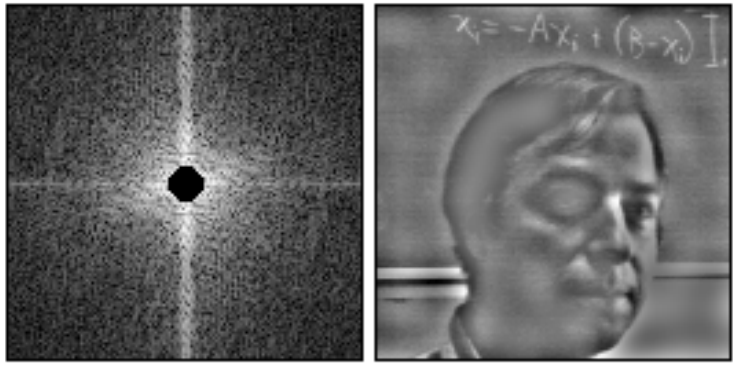
Brightness Image Fourier Transform Inverse Transformed



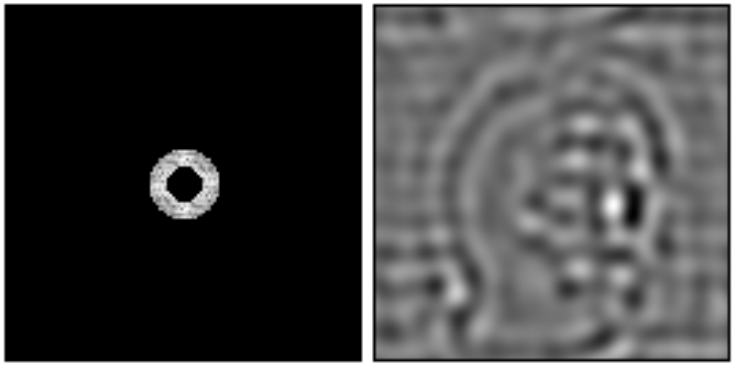
Low-Pass Filtered Inverse Transformed

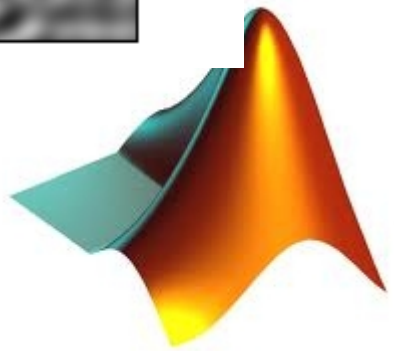


High-Pass Filtered Inverse Transformed



Band-Pass Filtered Inverse Transformed





DCT

- pre typický obrázok so spojitými prechodmi farieb je väčšina vizuálne najdôležitejších informácií o obrázku koncentrovaná v malom počte kosínusových funkcií nízkych frekvencií
- frekvenčné koeficienty sú reálne

A=dct2(B);
B=idct2(A);

http://www.mathworks.com/help/toolbox/images/ref/dct2.html