Convolution via the DFT

Dr. Daniel B. Rowe
Professor of Computational Statistics
Department of Mathematical and Statistical Sciences
Marquette University

Outline

Introduction

The Convolution Theorem

Time Series Convolution via the DFT

Image Convolution via the DFT

Image Convolution Example

Discussion

Homework

Introduction

We previously saw that the Fourier transform is useful for decomposing a time series signal or image into it's constituent temporal or spatial frequencies.

The discrete Fourier transform is also extremely useful for performing convolution.

Convolution via the discrete Fourier transform in frequency space is much faster than convolution in image space.

The Convolution Theorem

In the theory of Fourier transforms, there is a theorem that states that:

Performing convolution of a time series with a kernel in the time domain, is equivalent to pointwise multiplying the Fourier transforms of the time series and the kernel together then inverse Fourier transforming.

Performing convolution of an image with a kernel in the image domain, is equivalent to pointwise multiplying the Fourier transforms of the image and the kernel together then inverse Fourier transforming.

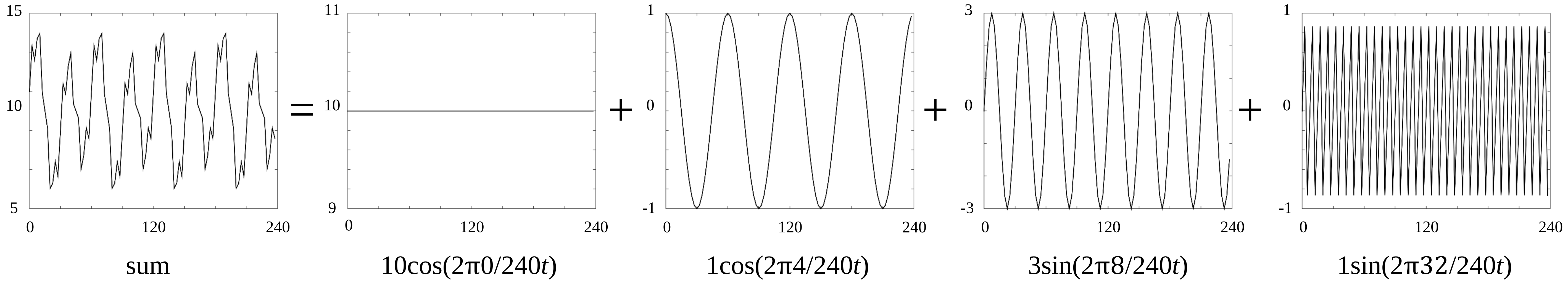
Let's see why and how this is true.

Time Series Convolution via the DFT

Example: Let’s sample the continuous time series (1D function)

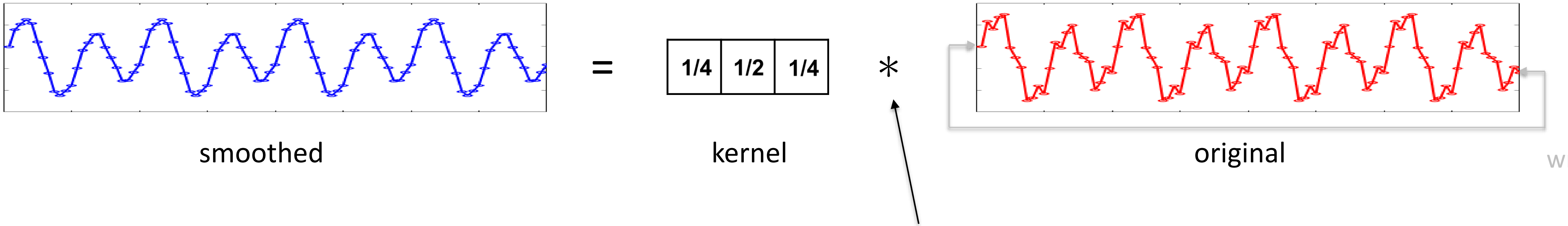
$$y(t)=10\cos\left(2\pi\frac{0}{240}t\right)+\cos\left(2\pi\frac{4}{240}t\right)+3\sin\left(2\pi\frac{8}{240}t\right)+\sin\left(2\pi\frac{32}{240}t\right)$$

at t=$1\Delta t,2\Delta t,3\Delta t,\ldots,n\Delta t$, where n=96 and Δt=2.5s for a total time of 240s.



Time Series Convolution via the DFT

When we performed convolution of a time series with a kernel, we moved the kernel, multiplied, and created a new time point. This was repeated to create an entirely new time series.



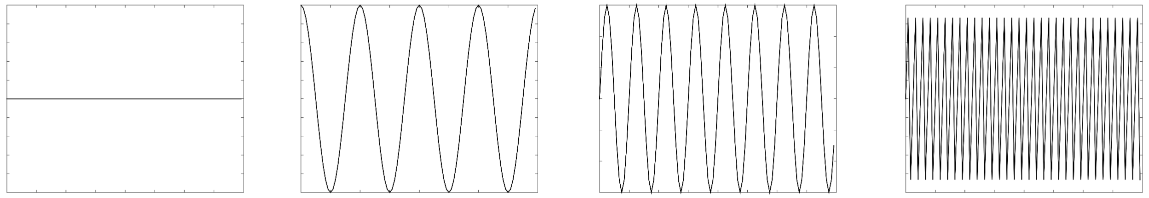
smoothed

kernel

original

wrap

convolution



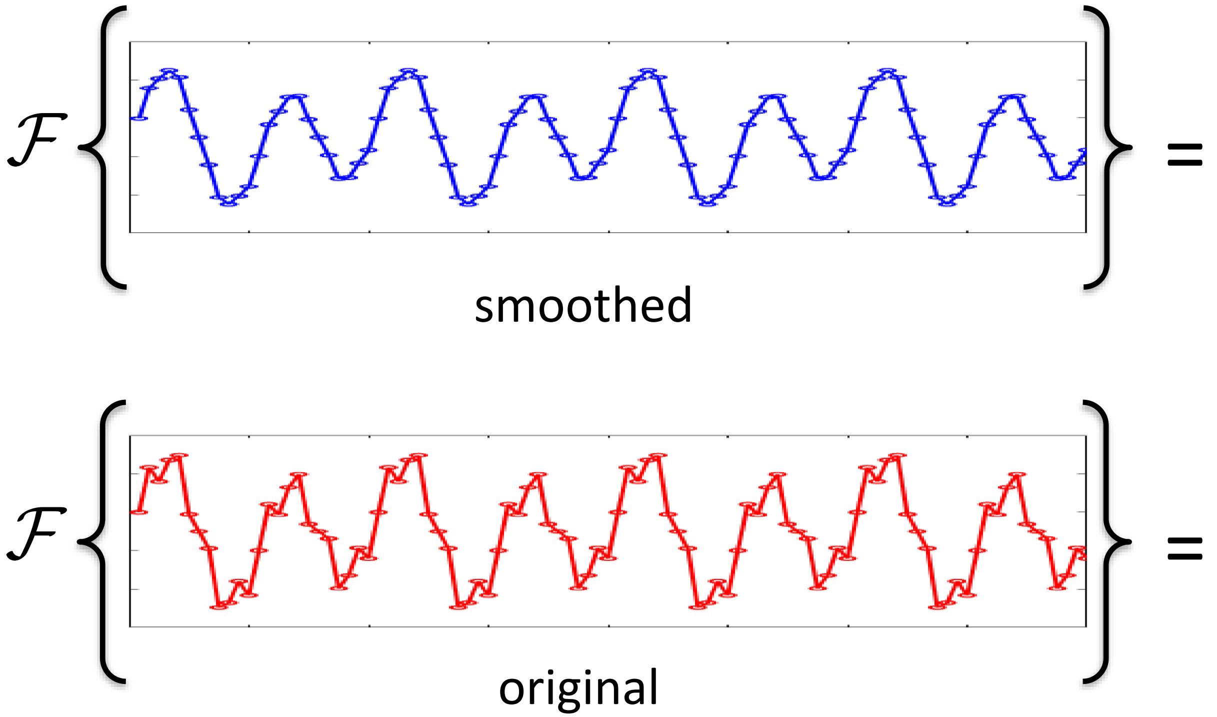
$$y(t) = 10\cos\left(2\pi\frac{0}{240}t\right) + 1\cos\left(2\pi\frac{4}{240}t\right) + 3\sin\left(2\pi\frac{8}{240}t\right) + 1\sin\left(2\pi\frac{32}{240}t\right)$$

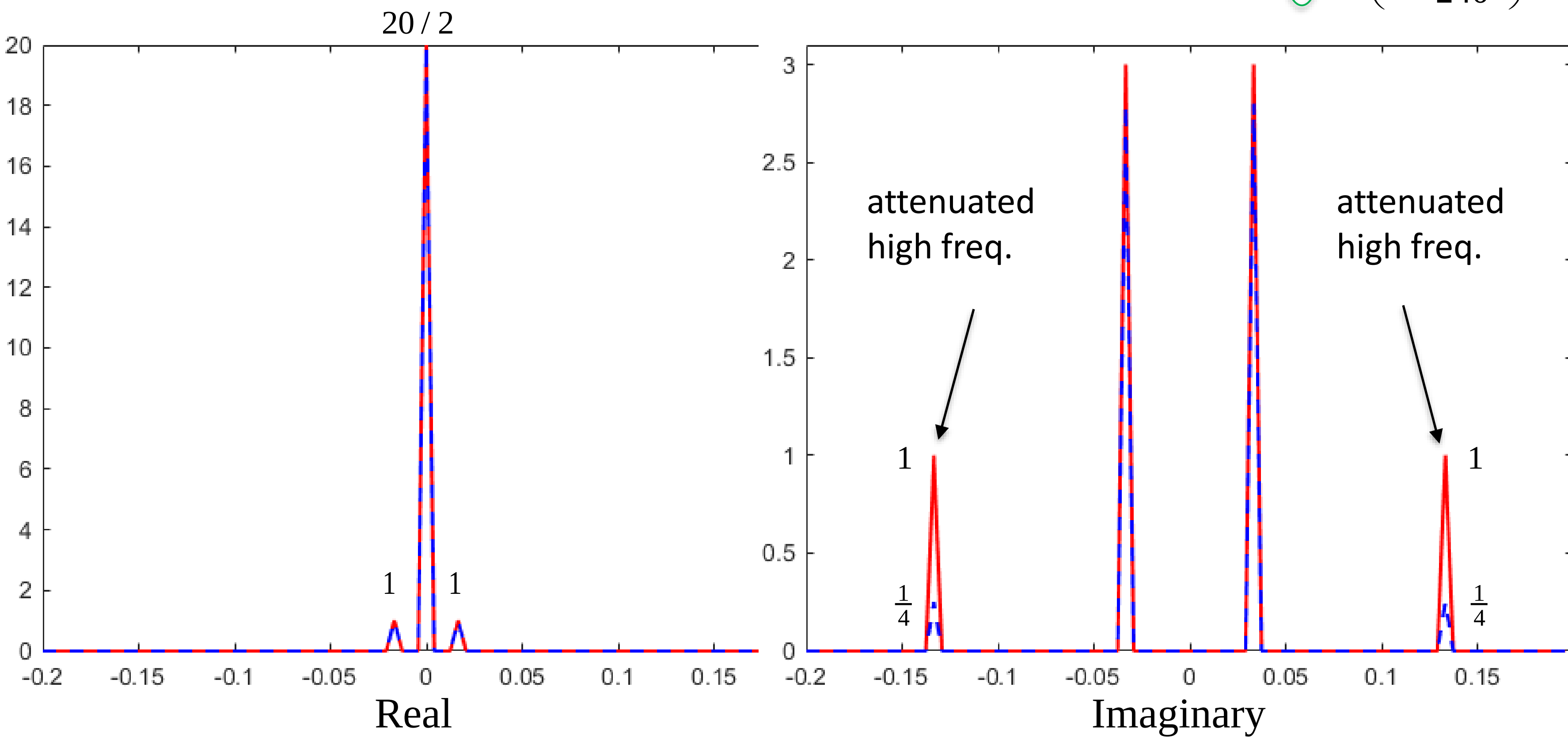
Time Series Convolution via the DFT

We can take the Fourier transform of the smoothed time series and compare it to the Fourier transform of the original input time series to see how the high frequency amplitude coefficients were attenuated.

1/4	**1/2**	**1/4**

$$\sin\left(2\pi\frac{32}{240}t\right)$$





$$y(t) = 10\cos\left(2\pi\frac{0}{240}t\right) + 1\cos\left(2\pi\frac{4}{240}t\right) + 3\sin\left(2\pi\frac{8}{240}t\right) + 1\sin\left(2\pi\frac{32}{240}t\right)$$

*coefficients divided by $n/2$ for display

Time Series Convolution via the DFT

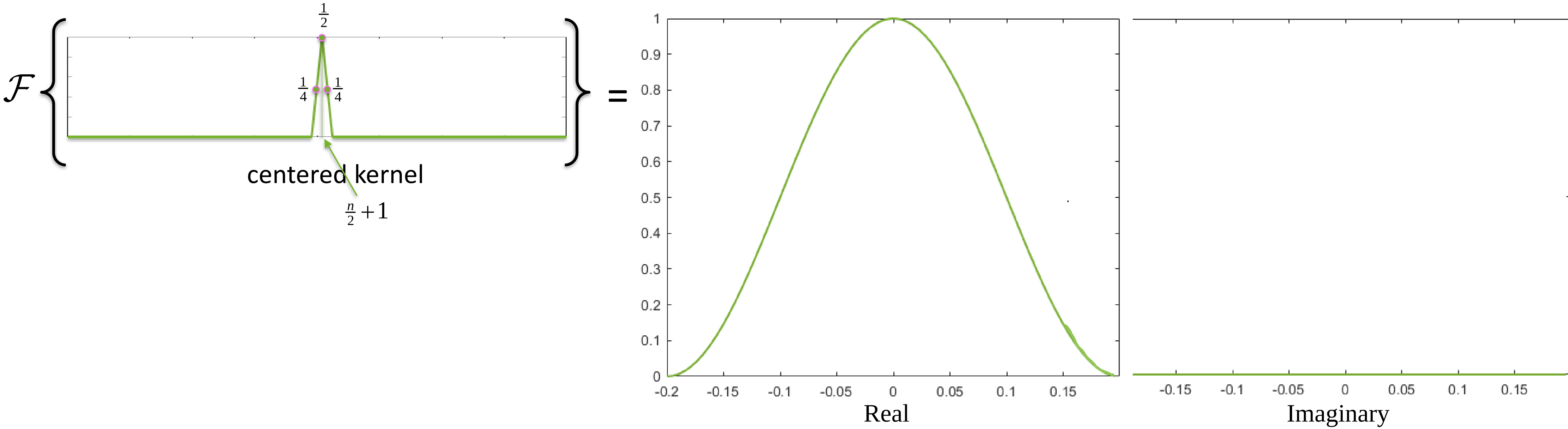
```
n=96; % number of time points
dt=2.5; %seconds between time points
t=(1:n)'; % sampled time points
A1=10; A2=1; A3=3; A4=1; % frequency amplitudes
nu1=0/(dt*n); nu2=4/(dt*n); % cosine frequencies
nu3=8/(dt*n); nu4=32/(dt*n); % sine frequencies
y=A1*cos(2*pi*nu1*(t-1)*dt)...
 +A2*cos(2*pi*nu2*(t-1)*dt)...
 +A3*sin(2*pi*nu3*(t-1)*dt)...
 +A4*sin(2*pi*nu4*(t-1)*dt);
figure;
plot(t,y,'r-o','LineWidth',1.5,'MarkerSize',5)
xlim([0,n]), ylim([5,15])
set(gca,'xtick',(0:n/8:n))
set(gca,'ytick',(5:2:15))

% perform convolution smoothing in time domain
kernel=[1/4,1/2,1/4];
k=length(kernel);
yw=[y(n-(k-1)/2+1:n);y;y(1:(k-1)/2)];
ysm=zeros(n,1);
for j=1:n
    ysm(j,1)=kernel*yw(j:j+(k-1),1);
end
```

```
figure;
plot(t,ysm,'b-o','LineWidth',1.5,'MarkerSize',5)
xlim([0,n]), ylim([5,15])
set(gca,'xtick',(0:n/8:n)
set(gca,'ytick',(5:2:15))

% compute FTs of original and smoothed
dnu=1/(n*dt);
nu=((1:n)'-n/2-1)*dnu;
fty=fftshift(fft(fftshift(y)));
ftysm=fftshift(fft(fftshift(ysm)));
figure;
plot(nu,2*real(fty)/n,'r','LineWidth',1.5)
hold on
plot(nu,2*real(ftysm)/n,'b--','LineWidth',1.5)
xlim([-1/dt/2,1/dt/2]),ylim([0,12])
set(gca,'xtick',round(nu(1:n/8:n),2))
figure;
plot(nu,2*abs(imag(fty))/n,'r','LineWidth',1.5)
hold on
plot(nu,2*abs(imag(ftysm))/n,'b--','LineWidth',1.5)
xlim([-1/dt/2,1/dt/2]),ylim([0,3.1])
set(gca,'xtick',round(nu(1:n/8:n),2))
```

Time Series Convolution via the DFT

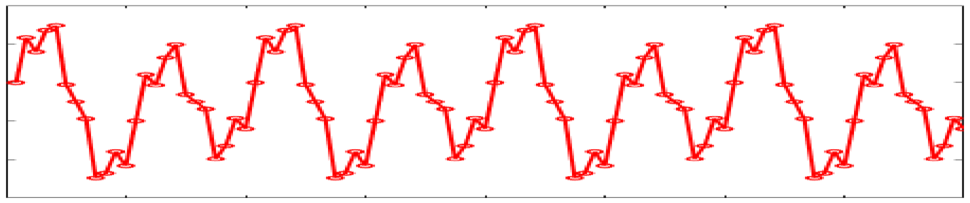
To demonstrate the convolution theorem, we take the discrete Forward Fourier transform of the centered kernel …

1/4	**1/2**	**1/4**

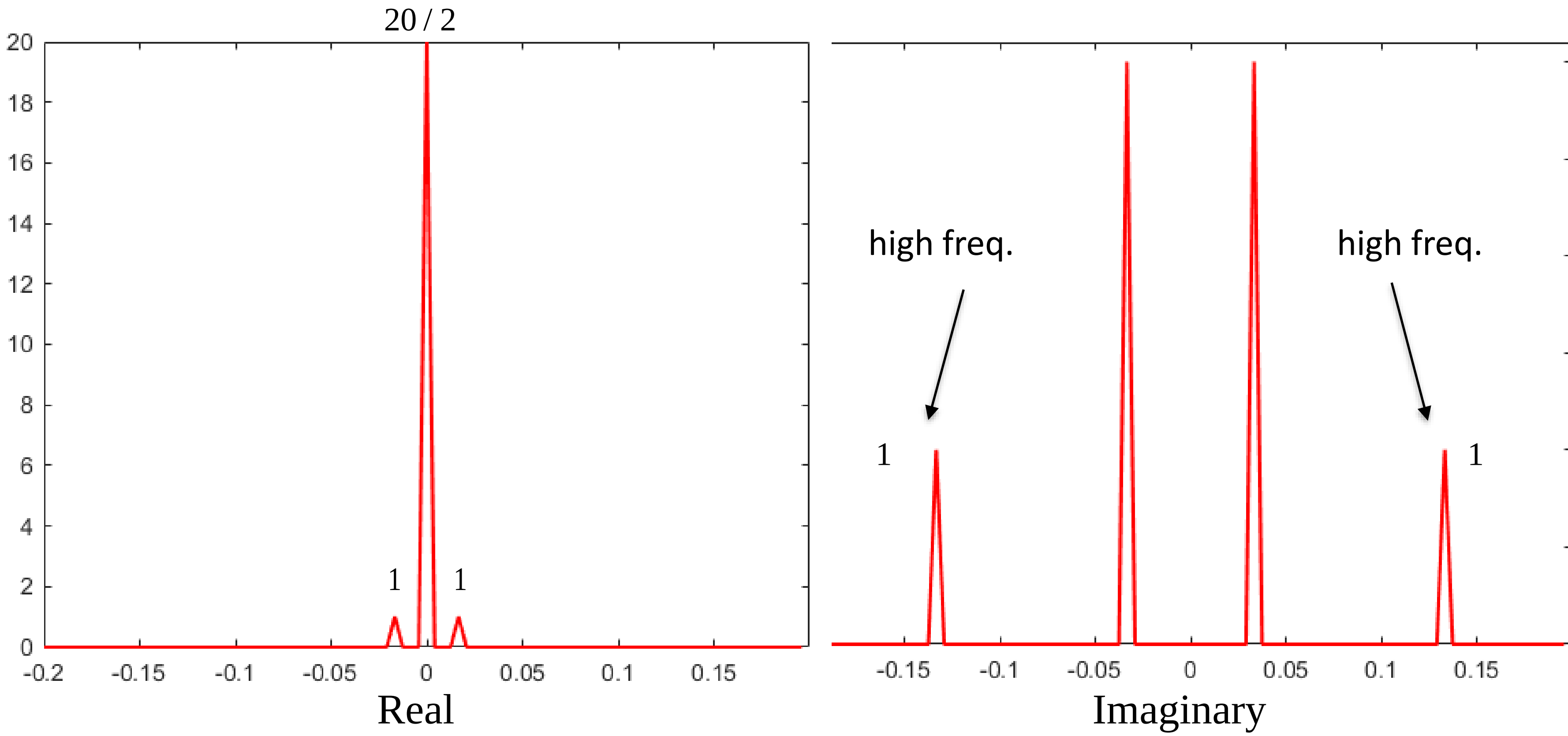


Time Series Convolution via the DFT

… and the forward discrete Fourier transform of the original input time series …

$\mathcal{F}\Bigg\{$  $\Bigg\} =$

original



$$y(t)=10\cos\left(2\pi\frac{0}{240}t\right)+1\cos\left(2\pi\frac{4}{240}t\right)+3\sin\left(2\pi\frac{8}{240}t\right)+1\sin\left(2\pi\frac{32}{240}t\right)$$

*coefficients divided by $n/2$ for display

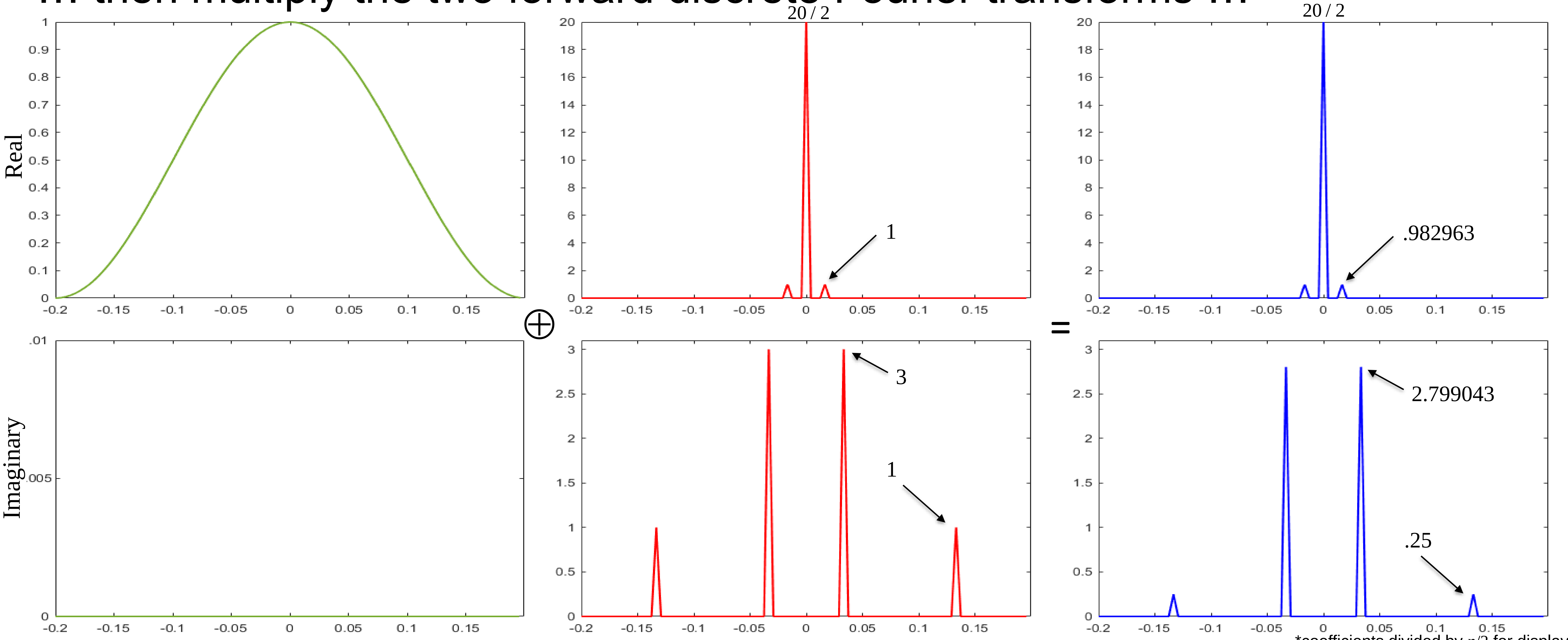
Time Series Convolution via the DFT

$$y(t)=10\cos\left(2\pi\frac{0}{240}t\right)+1\cos\left(2\pi\frac{4}{240}t\right)+3\sin\left(2\pi\frac{8}{240}t\right)+1\sin\left(2\pi\frac{32}{240}t\right)$$

$\oplus$ = direct product, element -wise complex multiplication

"`.*`" in Matlab

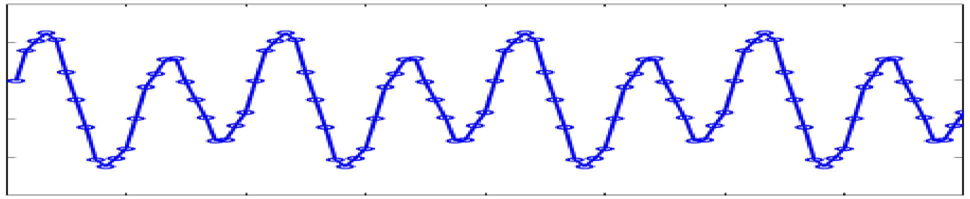
… then multiply the two forward discrete Fourier transforms …



*coefficients divided by $n/2$ for display

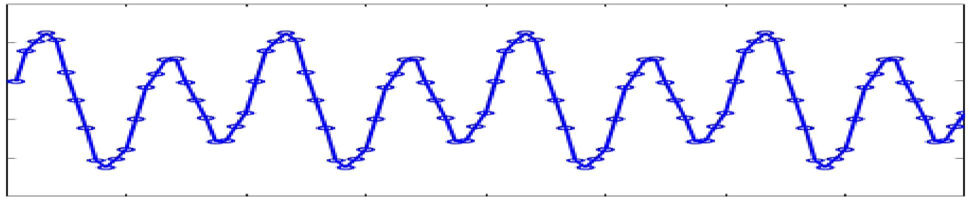
Time Series Convolution via the DFT

… and inverse discrete Fourier transform to obtain the smoothed time series.



FT smoothed

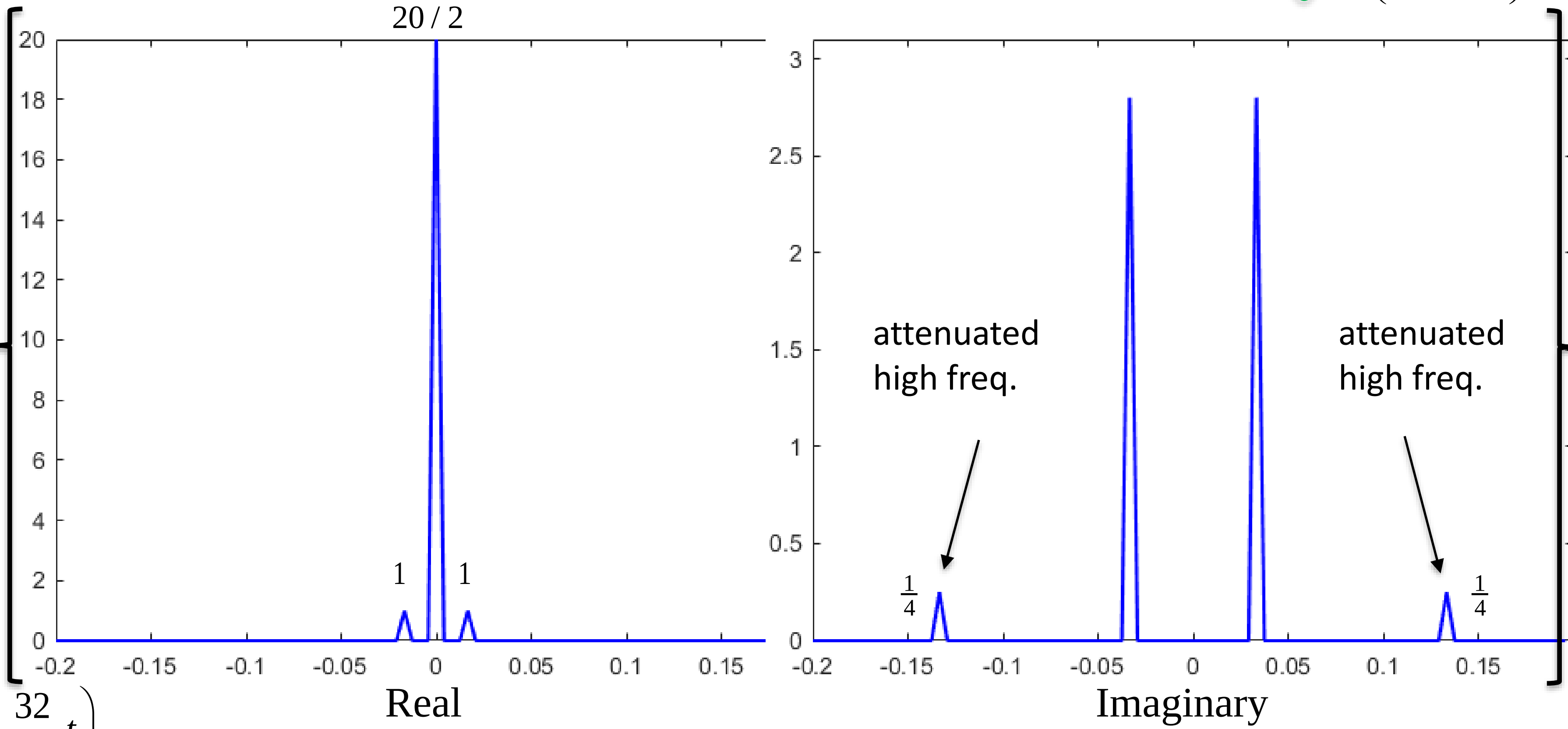
identical



Convolution smoothed

$= \mathcal{F}^{-1}$

$\sin\left(2\pi\frac{32}{240}t\right)$



$$y(t)=10\cos\left(2\pi\frac{0}{240}t\right)+.98\cos\left(2\pi\frac{4}{240}t\right)+2.8\sin\left(2\pi\frac{8}{240}t\right)+.25\sin\left(2\pi\frac{32}{240}t\right)$$

*coefficients divided by $n/2$ for display

Time Series Convolution via the DFT

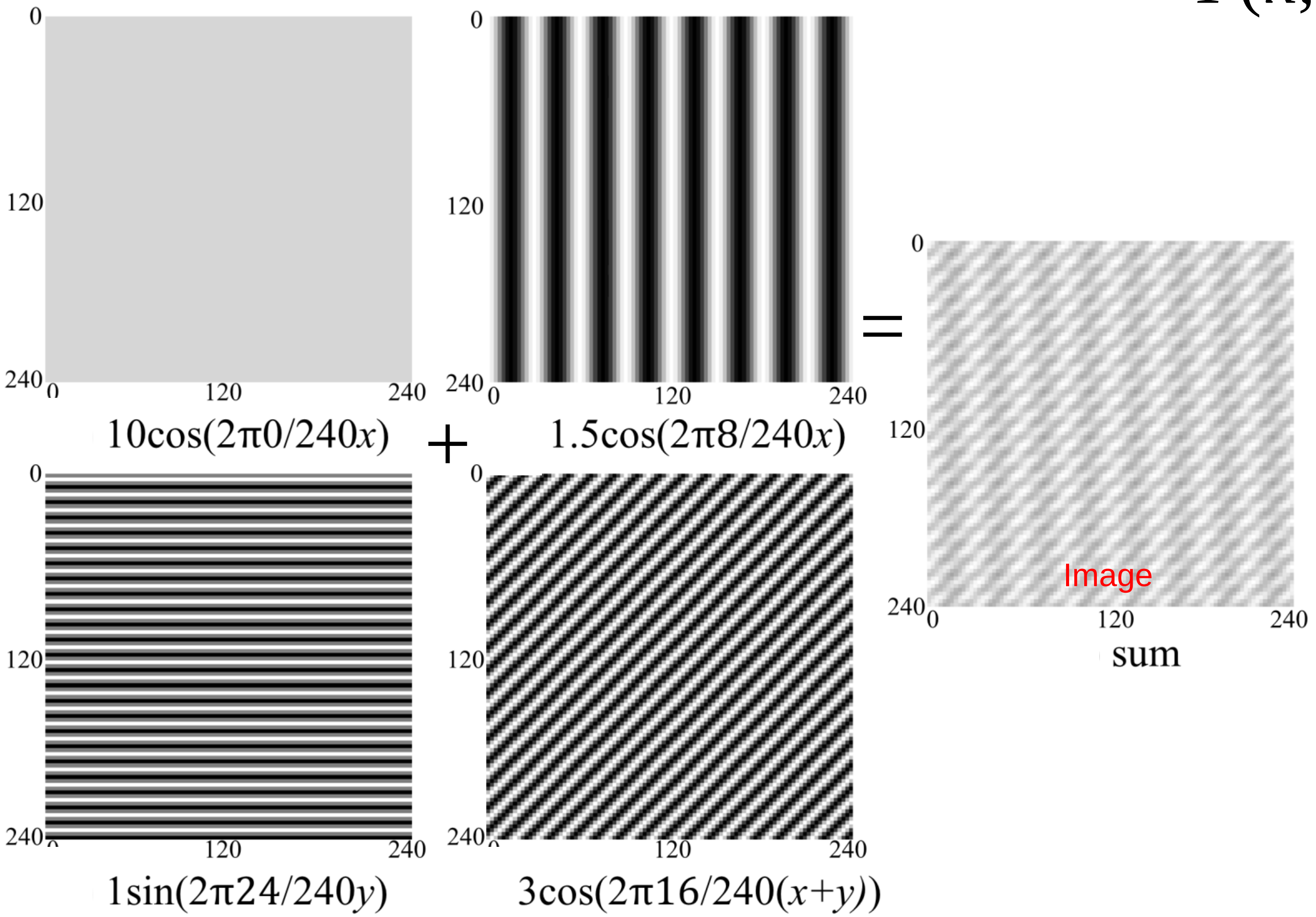
```
% convolution via FFT
kerncenter=[zeros(n/2-(k-1)/2,1);kernel';zeros(n/2-(k-1)/2-1,1)];
fty=fftshift(fft(fftshift(y)));
ftk=fftshift(fft(fftshift(kerncenter)));
figure;
plot(nu,real(ftk),'color',[0.4660,0.6740,0.1880],'LineWidth',1.5)
xlim([-1/dt/2,1/dt/2]),ylim([0,1])
set(gca,'xtick',round(nu(1:n/8:n),2))
figure;
plot(nu,abs(imag(ftk)),'color',[0.4660,0.6740,0.1880],'LineWidth',1.5)
xlim([-1/dt/2,1/dt/2]),ylim([0,.01])
set(gca,'xtick',round(nu(1:n/8:n),2))
set(gca,'ytick',[0,.005,.01])

ftyTftk=fty.*ftk;
ysmFT=fftshift(ifft(fftshift(ftyTftk)));
figure;
plot(nu,2*real(ftyTftk)/n,'b-','LineWidth',1.5)
xlim([-1/dt/2,1/dt/2]),ylim([0,20])
set(gca,'xtick',round(nu(1:n/8:n),2))
figure;
plot(nu,2*abs(imag(ftyTftk))/n,'b-','LineWidth',1.5)
xlim([-1/dt/2,1/dt/2]),ylim([0,3.1])
set(gca,'xtick',round(nu(1:n/8:n),2))
```

```
figure;
plot(t,ysm,'r','LineWidth',1.5)
hold on
plot(t,ysmFT,'b:','LineWidth',1.5,'MarkerSize',5)
xlim([0,n]), ylim([5,15])
set(gca,'xtick',(0:n/8:n)),set(gca,'ytick',(5:2:15))
```

Image Convolution via the DFT

Example: Let’s sample the continuous image scene (2D function)

$$Y(x,y)=10\cos\left(2\pi\frac{0}{240}x\right)+\frac{3}{2}\cos\left(2\pi\frac{8}{240}x\right)$$
$$+\sin\left(2\pi\frac{24}{240}y\right)+\cos\left(2\pi\frac{16}{240}(x+y)\right)$$



10cos(2π0/240x)

1.5cos(2π8/240x)

1sin(2π24/240y)

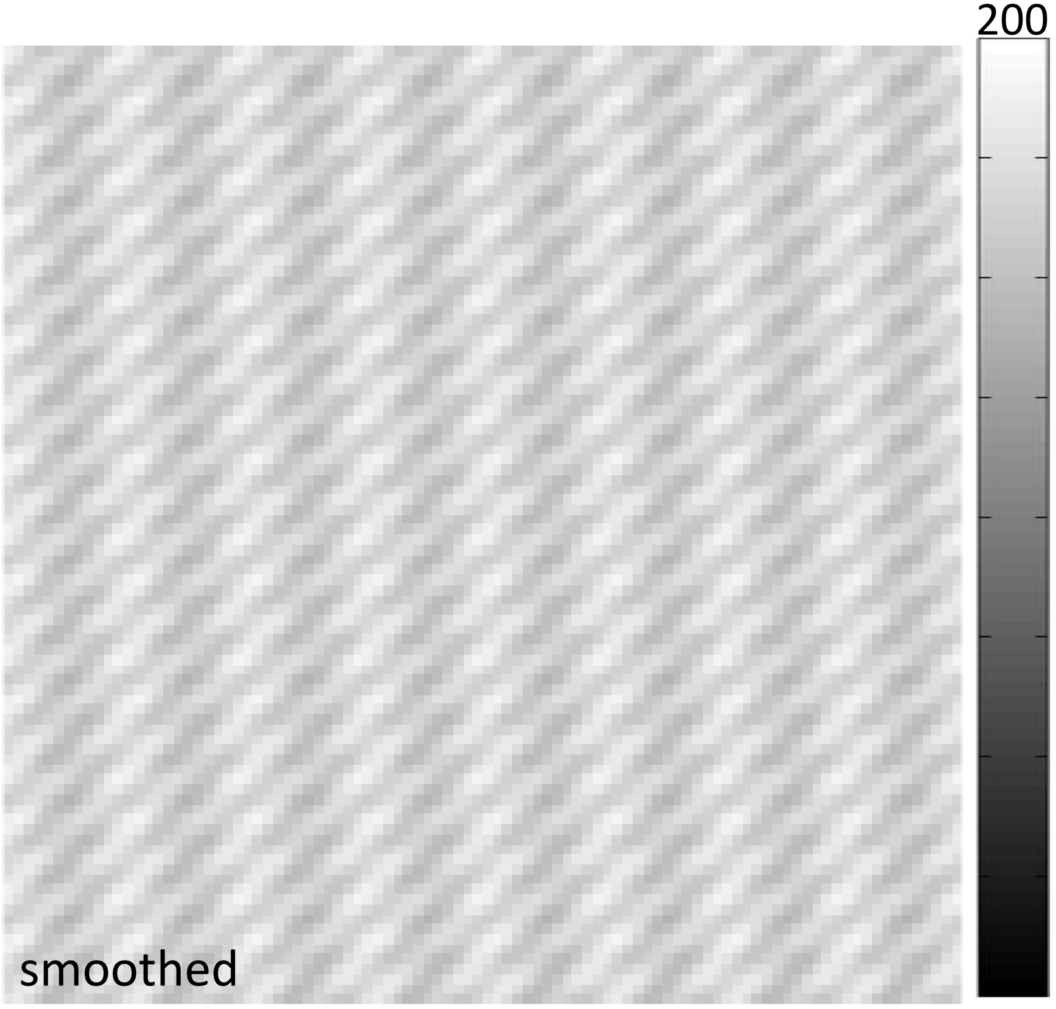
3cos(2π16/240(x+y))

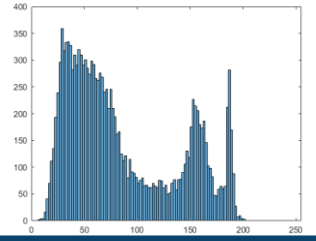
sum

Image Convolution via the DFT

$$Y(x,y)=10\cos\left(2\pi\frac{0}{240}x\right)+\frac{3}{2}\cos\left(2\pi\frac{8}{240}x\right)+\sin\left(2\pi\frac{24}{240}y\right)+\cos\left(2\pi\frac{16}{240}(x+y)\right)$$

We can use convolution in image space to compute a local mean image.





$\bar{x}_5$

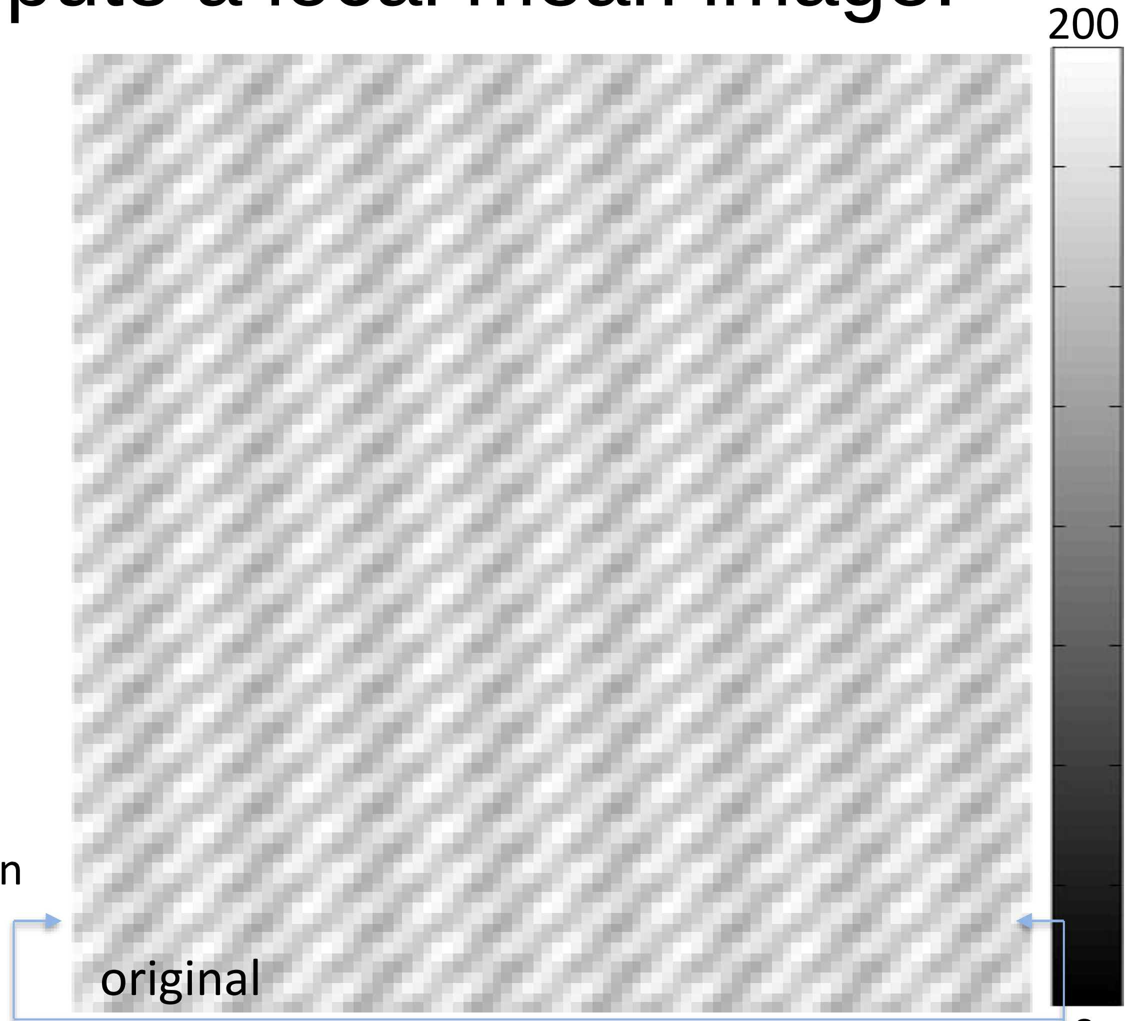
=

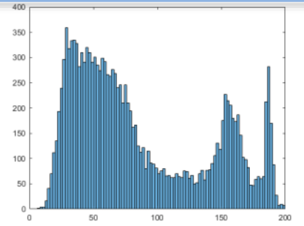
0	1/5	0
1/5	1/5	1/5
0	1/5	0

Kernel

$*$

convolution



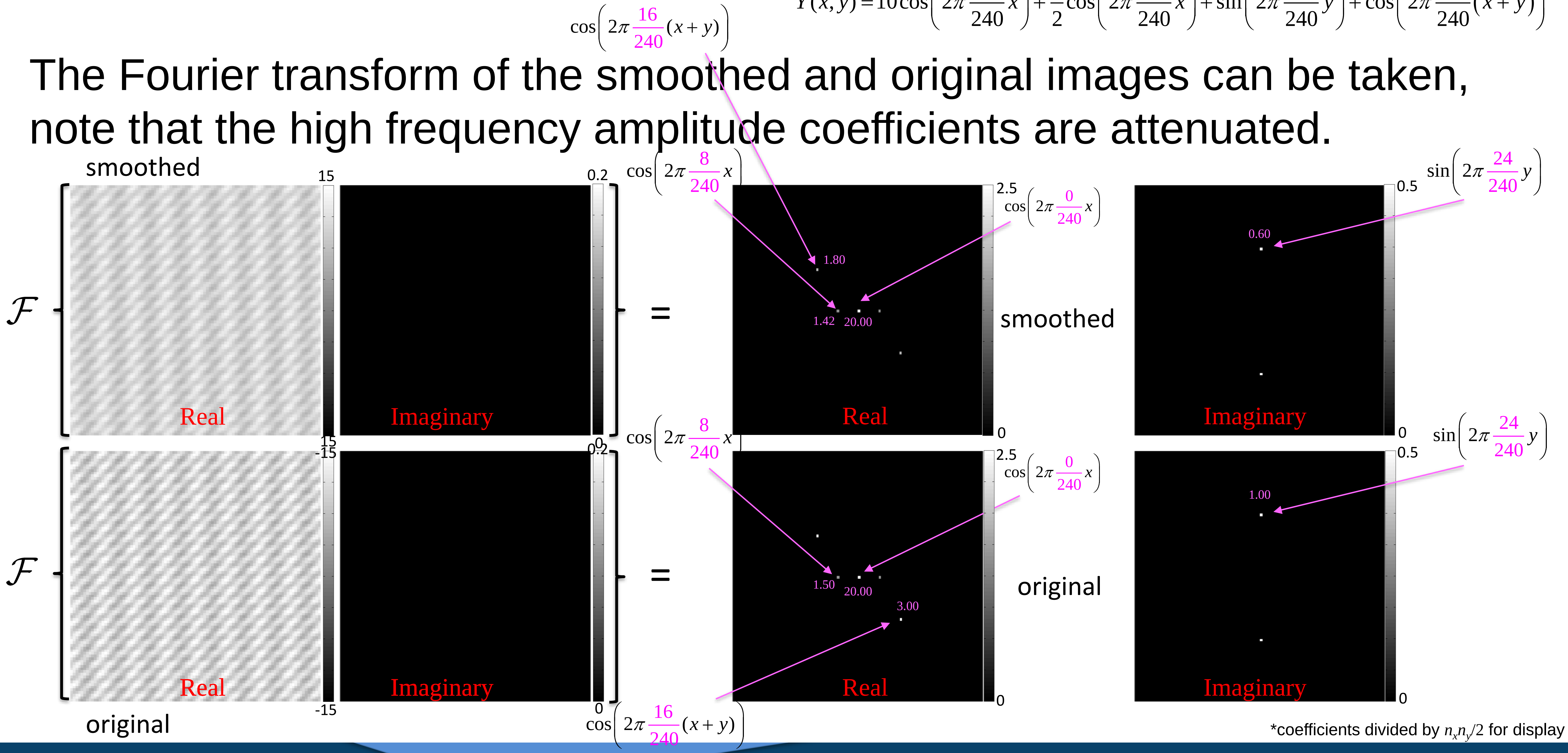


x_i

Time Series Convolution via the DFT

$$Y(x,y)=10\cos\left(2\pi\frac{0}{240}x\right)+\frac{3}{2}\cos\left(2\pi\frac{8}{240}x\right)+\sin\left(2\pi\frac{24}{240}y\right)+\cos\left(2\pi\frac{16}{240}(x+y)\right)$$

The Fourier transform of the smoothed and original images can be taken, note that the high frequency amplitude coefficients are attenuated.



*coefficients divided by $n_x n_y/2$ for display

Image Convolution via the DFT

```
ny=96; nx=ny;

FOVy=240; FOVx=240; deltay=FOVy/ny; deltax=FOVx/nx;
deltakx=1/ny/deltax; deltaky=1/nx/deltay;

A1=10; A2=1.5; A3=1; A4=3;
nu1=0/240; nu2=8/240; nu3=24/240; nu4=16/240;
Y=zeros(ny,nx);
for k=1:ny
    for j=1:nx
        Y(k,j)= A1*cos(2*pi*nu1*(j-1)*deltax)...
        + A2*cos(2*pi*nu2*(j-1)*deltax)...
        + A3*sin(2*pi*nu3*(k-1)*deltay)...
        + A4*cos(2*pi*(nu4*(j-1)*deltax+nu4*(k-1)*deltay));
    end
end
figure;
imagesc(Y,[-15,15])
axis image, axis off, colormap(gray)
figure;
imagesc(abs(imag(Y)),[0,1])
axis image, axis off, colormap(gray)
```

Image Convolution via the DFT

```
kernel=[0,1,0;...
        1,1,1;...
        0,1,0]/5;
[a,b]=size(kernel);

% appends border pixels for wrap-around
YW=[Y(ny-(a-1)/2+1:ny,nx-(b-1)/2+1:nx),Y(ny-(a-1)/2+1:ny,1:nx),Y(ny-(a-1)/2+1:ny,1:(b-1)/2);...
    Y(1:ny         ,nx-(b-1)/2+1:nx)  ,Y(1:ny           ,1:nx),Y(1:ny           ,1:(b-1)/2);...
    Y(1:(a-1)/2    ,nx-(b-1)/2+1:nx)  ,Y(1:(a-1)/2      ,1:nx),Y(1:(a-1)/2      ,1:(b-1)/2)];
% perform convolution
Ysm=zeros(ny,nx);
for j=1:ny
    for i=1:nx
        Ysm(j,i)=sum(sum(kernel.*YW(j:j+a-1,i:i+b-1)));
    end
end
figure;
imagesc(Ysm,[-15,15])
colormap(gray), axis image, axis off
figure;
imagesc(abs(imag(Ysm)),[0,1])
colormap(gray), axis image, axis off
```

Image Convolution via the DFT

```
% Fourier transform of smoothed image
ftYsm=fftshift(fft2(fftshift(Ysm)));
figure;
imagesc(2*real(ftYsm)/(nx*ny),[0,2.5])
colormap(gray), axis image, axis off
figure;
imagesc(2*abs(imag(ftYsm))/(nx*ny),[0,.5])
colormap(gray), axis image, axis off
% Fourier transform of original image
ftY=fftshift(fft2(fftshift(Y)));
figure;
imagesc(2*real(ftY)/(nx*ny),[0,2.5])
axis image, axis off, colormap(gray)
figure;
imagesc(2*abs(imag(ftY))/(nx*ny),[0,.5])
axis image, axis off, colormap(gray)
% inverse Fourier transform back
Yhat=fftshift(ifft2(fftshift(ftY)));
figure;
imagesc(real(Yhat),[-15,15])
axis image, axis off, colormap(gray)
figure;
imagesc(imag(Yhat),[-15,15])
axis image, axis off, colormap(gray)
```

Image Convolution via the DFT

0	1/5	0
1/5	1/5	1/5
0	1/5	0

$$Y(x,y)=10\cos\left(2\pi\frac{0}{240}x\right)+\frac{3}{2}\cos\left(2\pi\frac{8}{240}x\right)+\sin\left(2\pi\frac{24}{240}y\right)+\cos\left(2\pi\frac{16}{240}(x+y)\right)$$

To demonstrate the convolution theorem, we take the discrete Forward Fourier transform of the centered kernel and image …

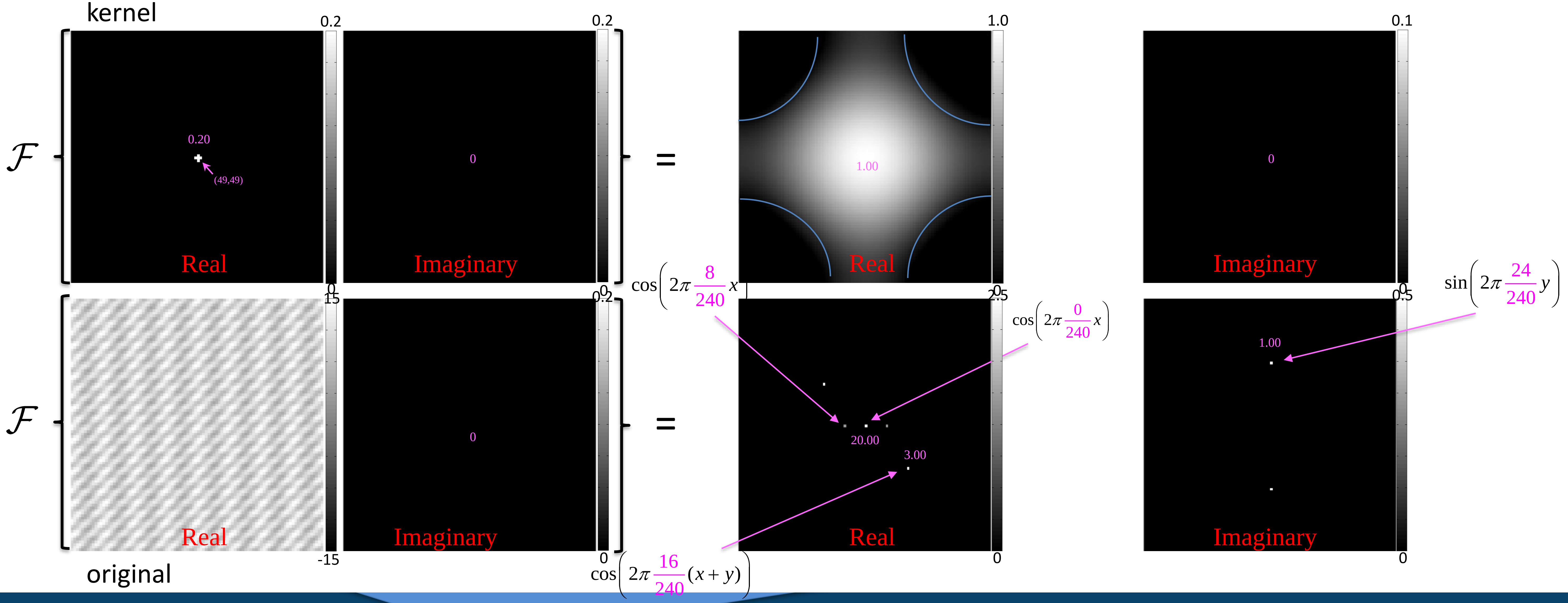


Image Convolution via the DFT

⊕ = direct product, element
-wise multiplication

“.*” in Matlab

… then multiply the two forward discrete Fourier transform together …

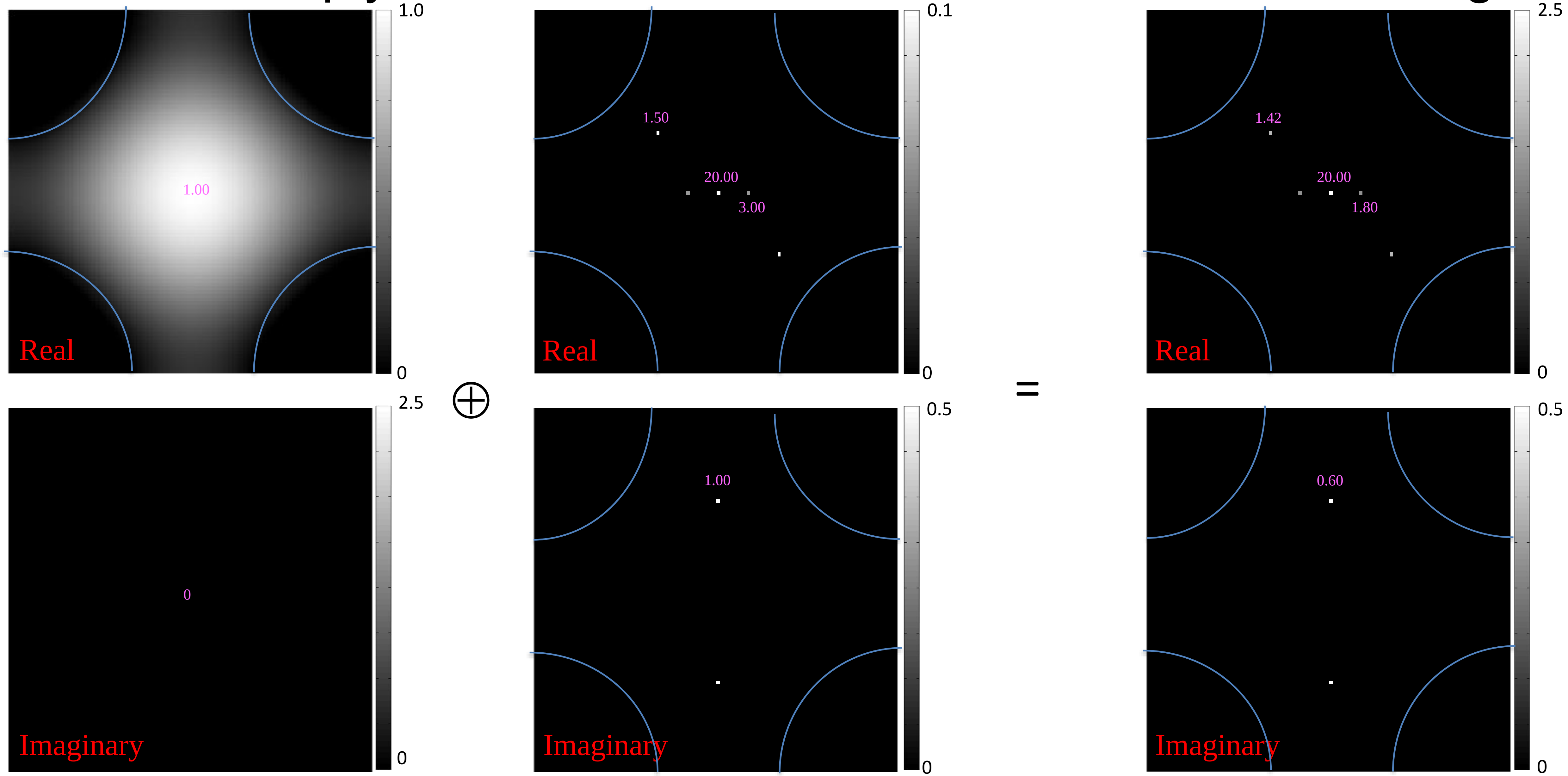


Image Convolution via the DFT

$$Y(x,y)=10\cos\left(2\pi\frac{0}{240}x\right)+\frac{3}{2}\cos\left(2\pi\frac{8}{240}x\right)+\sin\left(2\pi\frac{24}{240}y\right)+\cos\left(2\pi\frac{16}{240}(x+y)\right)$$

… and inverse discrete Fourier transform to get our smoothed image.

This process yields the same result as convolution of the image with a kernel in image assuming wrap around!

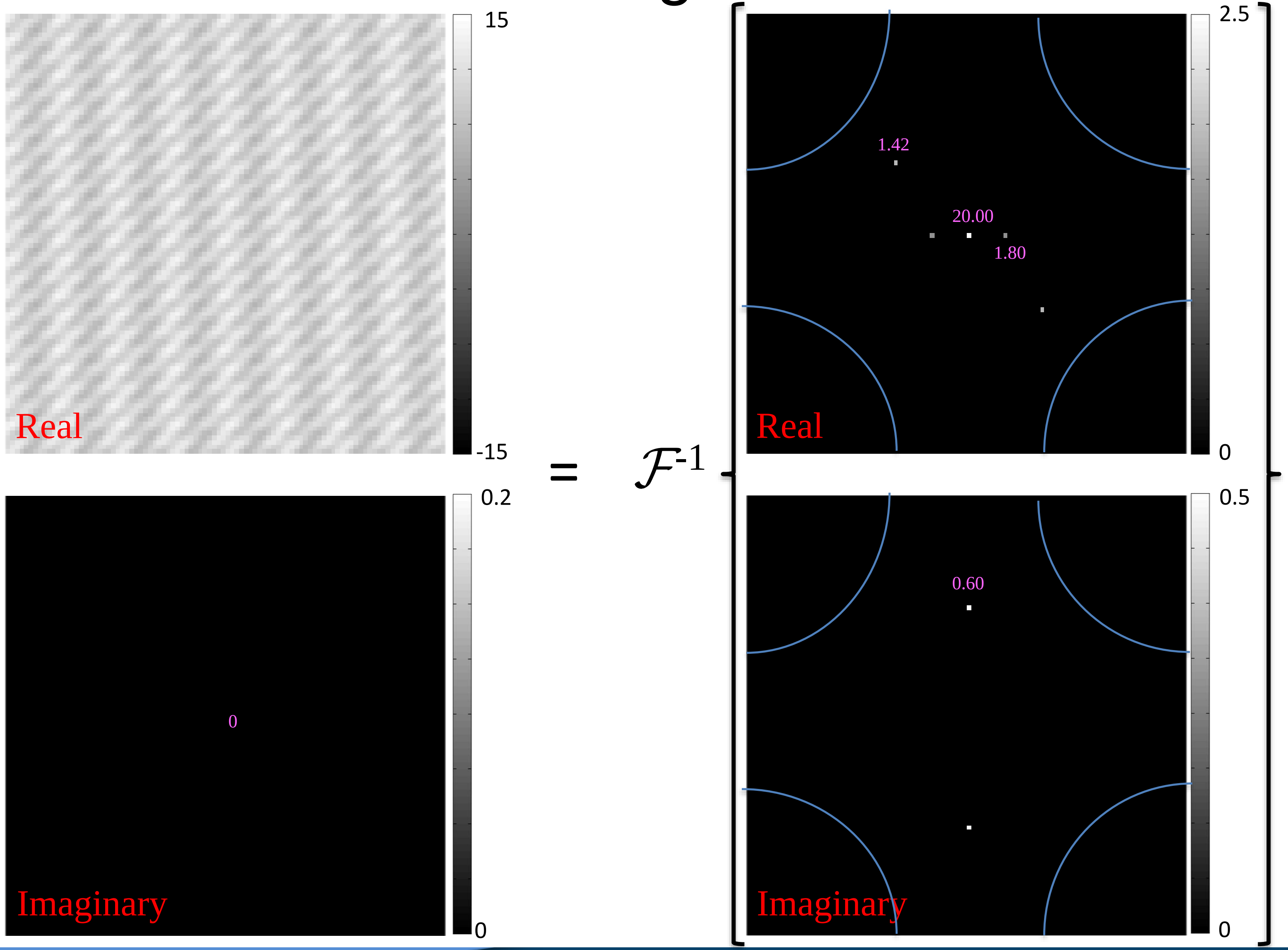


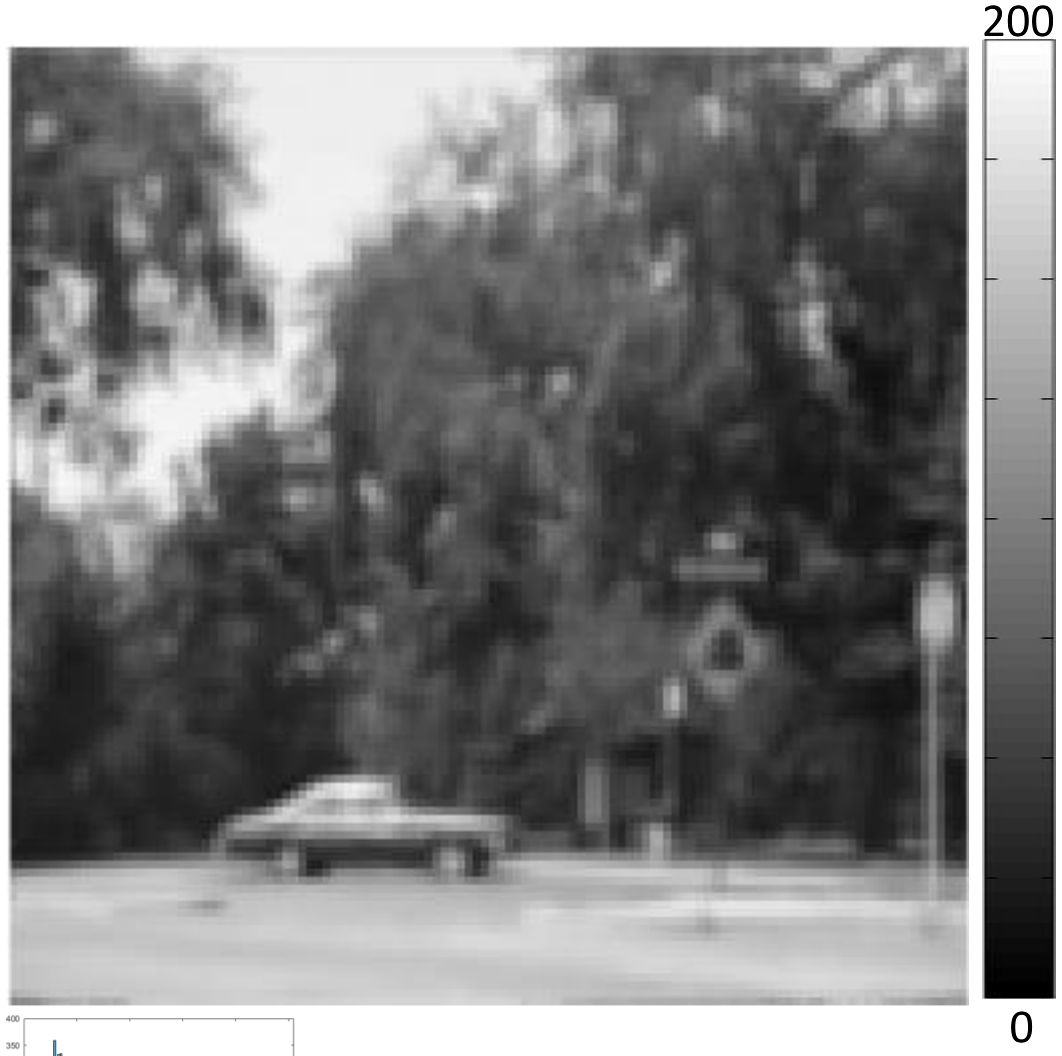
Image Convolution via the DFT

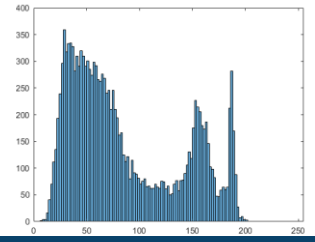
```
kMat(ny/2-(a-1)/2+1:ny/2+(a-1)/2+1,nx/2-(a-1)/2+1:nx/2+(a-1)/2+1)=kernel;
figure;
imagesc(real(kMat),[0,.2])
colormap(gray), axis image, axis off
figure;
imagesc(imag(kMat),[0,.2])
colormap(gray), axis image, axis off
ftkMat=fftshift(fft2(fftshift(kMat)));
figure;
imagesc(real(ftkMat),[0,1])
axis image, axis off, colormap(gray)
print(gcf,'-dtiffn','-r100','ftkMatR')
figure;
imagesc(abs(imag(ftkMat)),[0,0.1])
axis image, axis off, colormap(gray)
print(gcf,'-dtiffn','-r100','ftkMatI')
ftkMatftY=ftkMat.*ftY;
figure;
imagesc(2*real(ftkMatftY)/(nx*ny),[0,2.5])
axis image, axis off, colormap(gray)
figure;
imagesc(2*abs(imag(ftkMatftY))/(nx*ny),[0,.5])
axis image, axis off, colormap(gray)
```

```
YsmFT=fftshift(ifft2(fftshift(ftkMatftY)));
figure;
imagesc(YsmFT,[-15,15])
colormap(gray), axis image, axis off
print(gcf,'-dtiffn','-r100','YsmFTR')
figure;
imagesc(abs(imag(YsmFT)),[0,1])
colormap(gray), axis image, axis off
print(gcf,'-dtiffn','-r100','YsmFTI')
```

Image Convolution Example

We can use convolution in image space to compute a local weighted mean.





$\bar{x}_5$

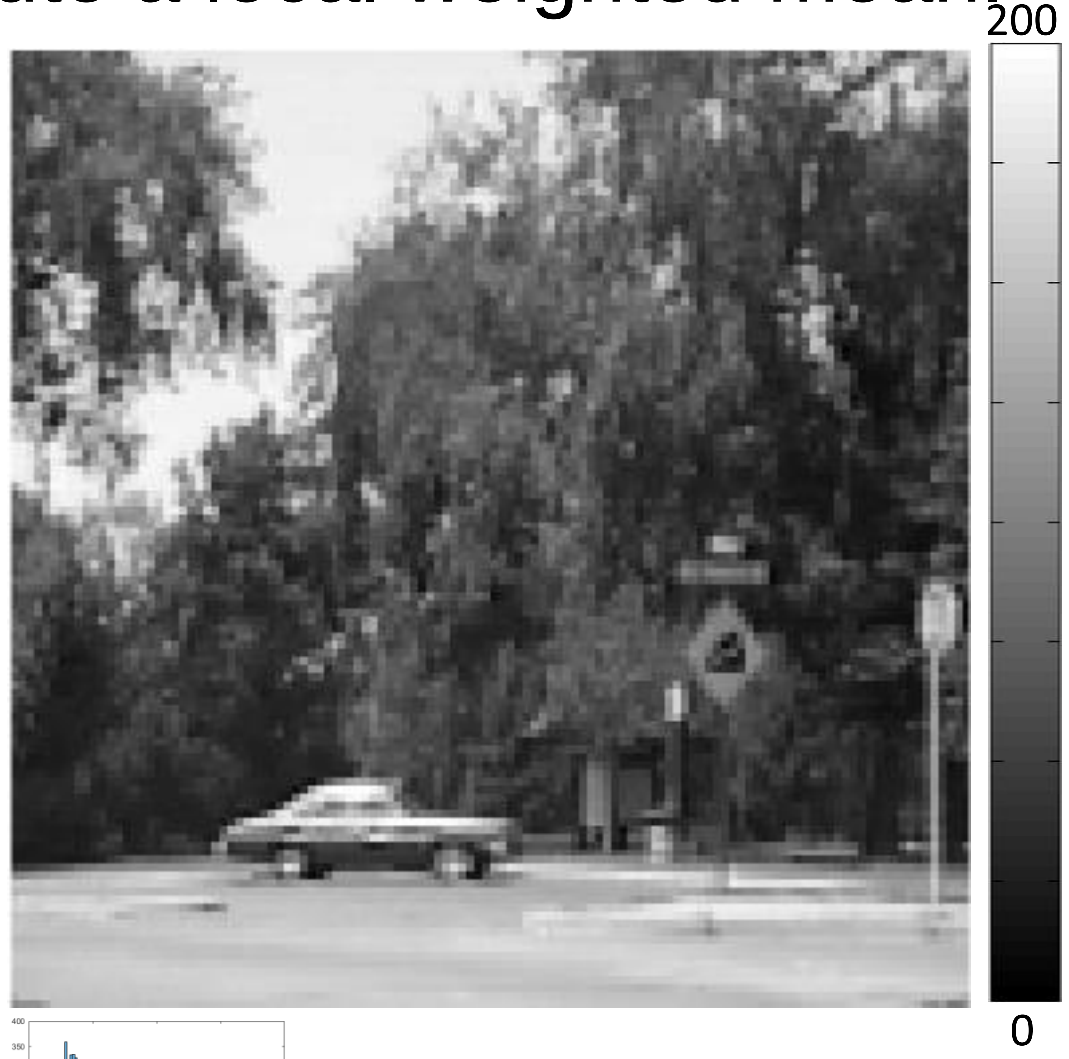
=

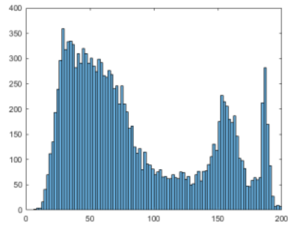
0	1/5	0
1/5	1/5	1/5
0	1/5	0

Kernel

$*$

convolution



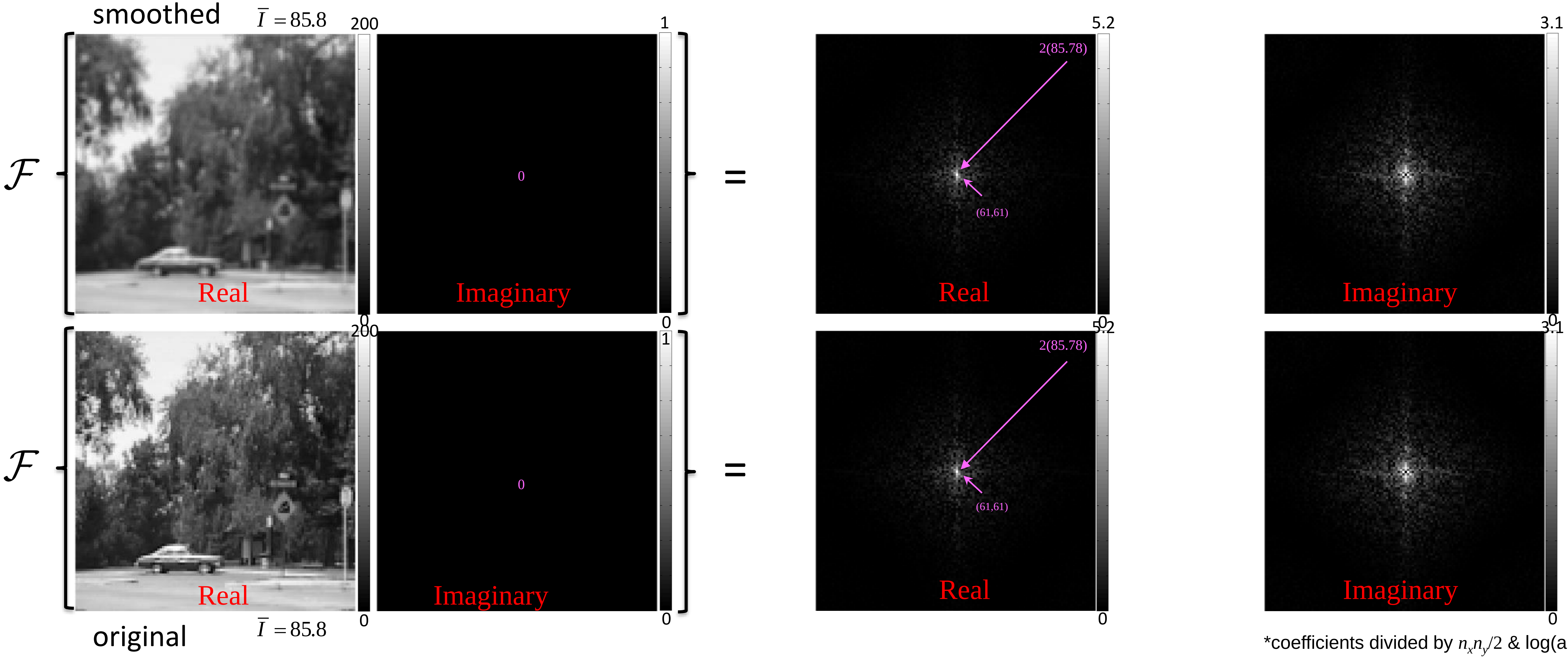


x_i

Image Convolution Example

120×120 image

We can Fourier transform of the smoothed image and compare it to that of the original input image to see high frequency coefficient attenuation.



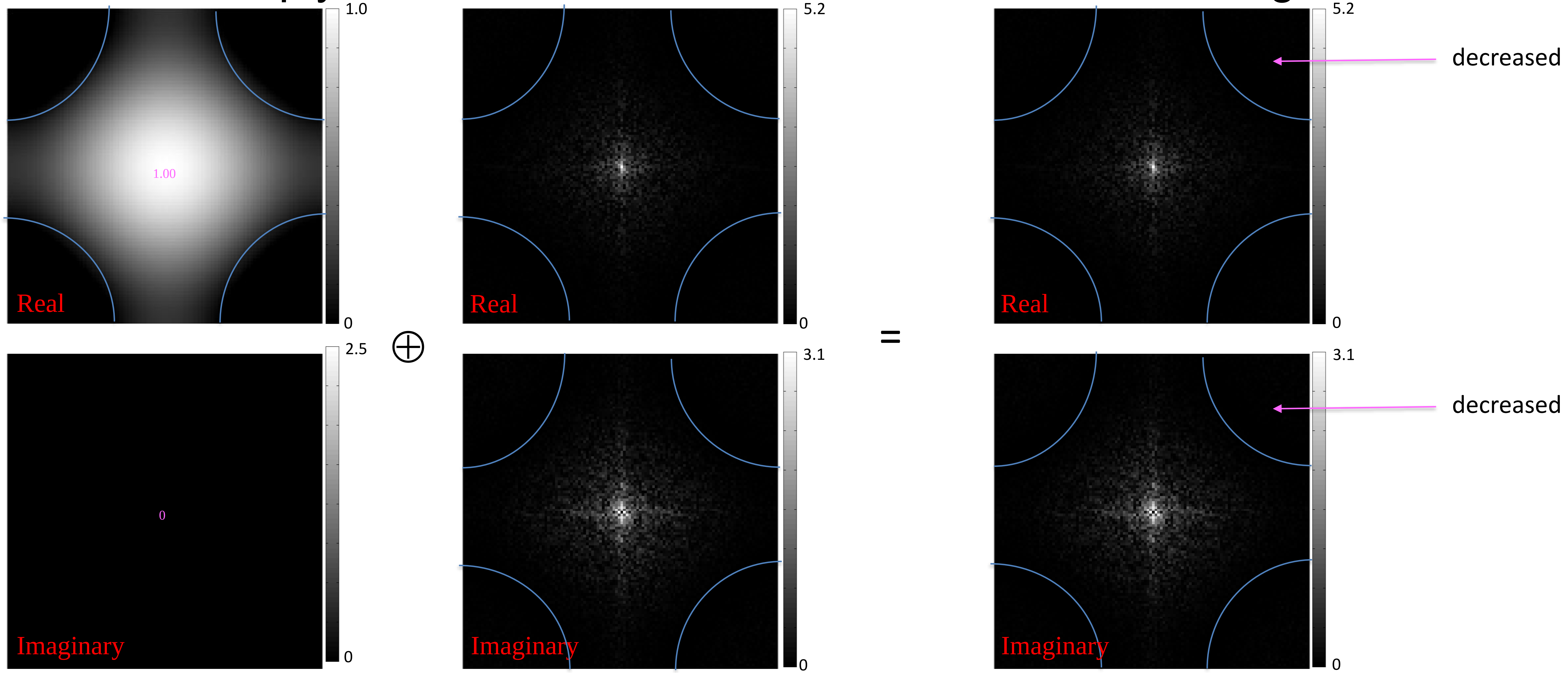
*coefficients divided by $n_x n_y/2$ & log(abs(f)+1) for display

Image Convolution Example

⊕ = direct product, element-wise complex multiplication

".*" in Matlab

… then multiply the two forward discrete Fourier transform together …



decreased

decreased

Image Convolution Example

… and inverse discrete Fourier transform to get our smoothed image.

This process yields the same result as convolution of the image with a kernel in image space assuming wrap around!

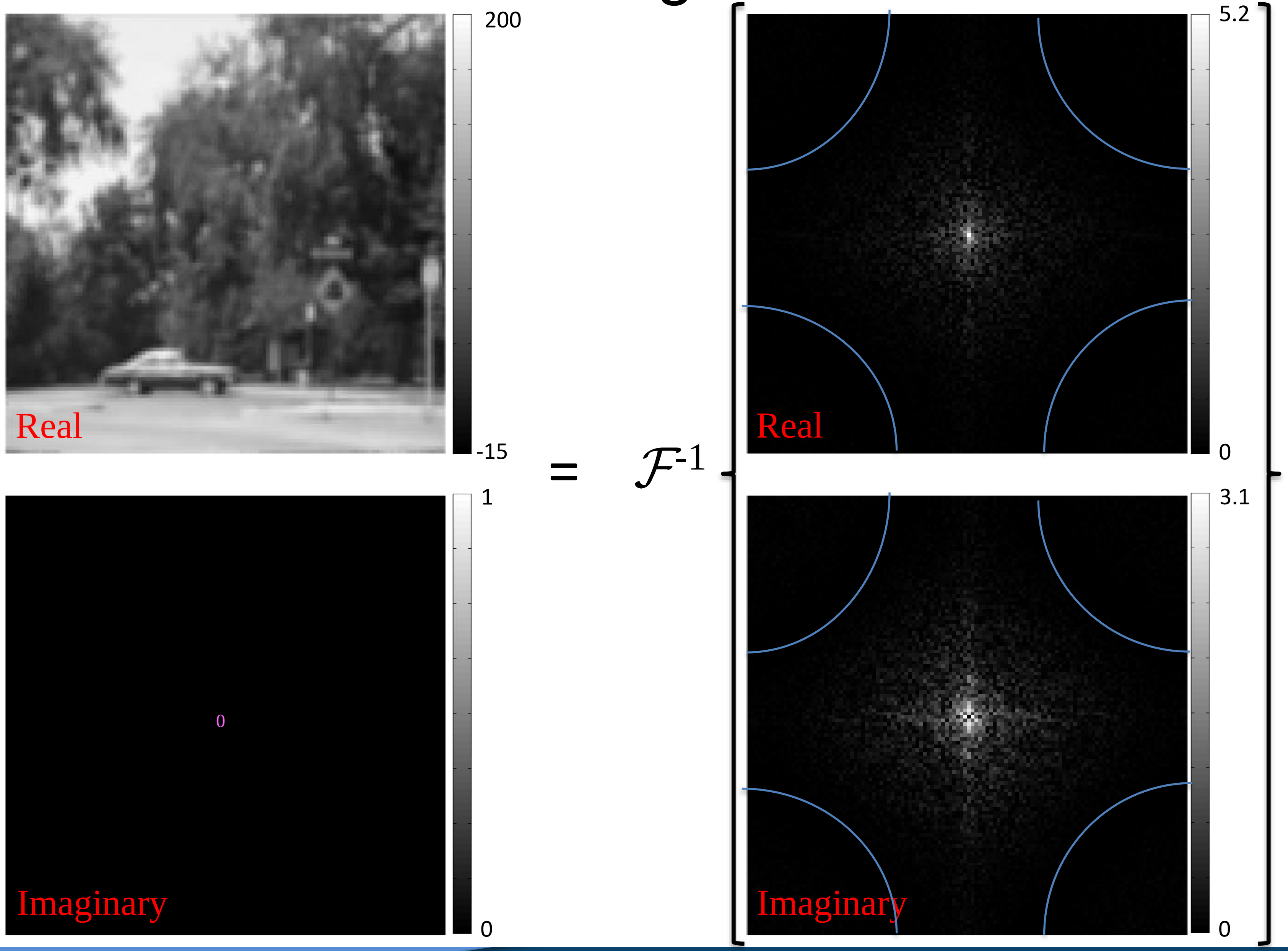


Image Convolution Example

```
% form kernel for convolution
kernel=[0,1,0;1,1,1;0,1,0]/5;
[a,b]=size(kernel);

load cardata.txt
[n,p]=size(cardata);
nx=sqrt(n); ny=nx;
I=double(reshape(cardata,[ny,nx])');
figure;,imagesc(I,[0,200])
colormap(gray), axis image, axis off
figure;,imagesc(imag(I),[0,1])
colormap(gray), axis image, axis off

% appends border pixels for wrap-around
IW=[I(ny-(a-1)/2+1:ny,nx-(b-1)/2+1:nx)  ,I(ny-(a-1)/2+1:ny,1:nx),I(ny-(a-1)/2+1:ny,1:(b-1)/2);...
    I(1:ny         ,nx-(b-1)/2+1:nx),I(1:ny           ,1:nx),I(1:ny           ,1:(b-1)/2);...
    I(1:(a-1)/2    ,nx-(b-1)/2+1:nx),I(1:(a-1)/2      ,1:nx),I(1:(a-1)/2      ,1:(b-1)/2)];
% perform convolution
Ism=zeros(ny,nx);
for j=1:ny
    for i=1:nx
        Ism(j,i)=sum(sum(kernel.*IW(j:j+a-1,i:i+b-1)));
    end
end
```

```
figure;,imagesc(Ism,[0,200])
colormap(gray), axis image, axis off
figure;,imagesc(imag(Ism),[0,1])
colormap(gray), axis image, axis off
figure;,histogram(I(:),(0:2:255))
xlim([0,255]), ylim([0,400])
```

Image Convolution Example

```
% fourier transform of smoothed image
ftIsm=fftshift(fft2(fftshift(Ism)));
maxftIsmreal=max(max(2*real(ftIsm)/(nx*ny)));
maxftIsmimag=max(max(2*imag(ftIsm)/(nx*ny)));
figure;
imagesc(log(2*abs(real(ftIsm))/n+1),[0,log(maxftIsmreal+1)])
colormap(gray), axis image, axis off
figure;
imagesc(log(2*abs(imag(ftIsm))/n+1),[0,log(maxftIsmimag+1)])
colormap(gray), axis image, axis off

ftI=fftshift(fft2(fftshift(I)));
figure;
imagesc(log(2*abs(real(ftIsm))/n+1),[0,log(maxftIsmreal+1)])
colormap(gray), axis image, axis off
figure;
imagesc(log(2*abs(imag(ftIsm))/n+1),[0,log(maxftIsmimag+1)])
colormap(gray), axis image, axis off
% check zero frequency
[mean(I(:)),real(ftI(61,61)/(nx*ny))]

kMat=zeros(ny,nx);
kMat(ny/2-(a-1)/2+1:ny/2+(a-1)/2+1,nx/2-(a-1)/2+1:nx/2+(a-1)/2+1)=kernel;
```

```
figure;
imagesc(real(kMat),[0,.2])
colormap(gray), axis image, axis off
figure;
imagesc(imag(kMat),[0,.2])
colormap(gray), axis image, axis off

ftkMat=fftshift(fft2(fftshift(kMat)));
figure;
imagesc(real(ftkMat),[0,1])
axis image, axis off, colormap(gray)
figure;
imagesc(abs(imag(ftkMat)),[0,0.1])
axis image, axis off, colormap(gray)
```

Image Convolution Example

```
figure;
imagesc(log(abs(2*real(ftI)/(nx*ny))+1),[0,log(maxft
Ismreal+1)])
axis image, axis off, colormap(gray)
figure;
imagesc(log(2*abs(imag(ftI))/(nx*ny)+1),[0,log(maxft
Ismimag+1)])
axis image, axis off, colormap(gray)

ftkMatftI=ftkMat.*ftI;
figure;
imagesc(log(abs(2*real(ftkMatftI)/(nx*ny))+1),[0,log
(maxftIsmreal+1)])
axis image, axis off, colormap(gray)
figure;
imagesc(log(2*abs(imag(ftkMatftI))/(nx*ny)+1),[0,log
(maxftIsmimag+1)])
axis image, axis off, colormap(gray)
```

```
IsmFT=fftshift(ifft2(fftshift(ftkMatftI)));
figure;
imagesc(IsmFT,[0,200])
colormap(gray), axis image, axis off
figure;
imagesc(abs(imag(IsmFT)),[0,1])
colormap(gray), axis image, axis off
```

Discussion

We just saw how the DFT can be used to perform convolution.

With the DFT, we can smooth or sharpen images much faster and quickly find objects in an image with template matching.

It is important to understand how the Fourier transform works and be able to critically reason its operation characteristics.

Discussion

Questions?

Homework 9

1. Make up your own time series with sines and cosines.
   (a) forward discrete Fourier transform it to discover your frequencies. Make plots of time series and Fourier coefficients.
   (b) Set the coefficients for one frequency to zero and inverse discrete Fourier transform. Comment.

2. Select a kernel and apply it to your own time series by moving across. Compute the forward discrete Fourier transform of your centered kernel and your time series, pointwise multiply, inverse discrete Fourier transform. Make intermediate plots and images.

3. Repeat #2 but for an image.

Homework 9

4*. Add a very strong left-right plane wave to you image.
Compute the forward discrete Fourier transform of your new image.
Find the coefficients for your frequency. Make sure the amplitude matches. Set this frequency pair to zero and inverse Fourier transform.

5*. Make up your own homework problem to present to the class.

*For students in MSSC 5770.