Lecture 24

2D Circular Convolution

$$f(x,y) \star h(x,y) = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(m,n)h(x-m, y-n)$$

for $x = 0, 1, 2, \dots, M-1$, and $x = 0, 1, 2, \dots, N-1$

The 2-D convolution theorem is given by

$$f(x,y) \star h(x,y) \Leftrightarrow F(u,v)H(u,v)$$

Convolution in Spatial Domain <=> Multiplication in Frequency Domain

$$f(x,y)h(x,y) \Leftrightarrow F(u,v) \star H(u,v)$$

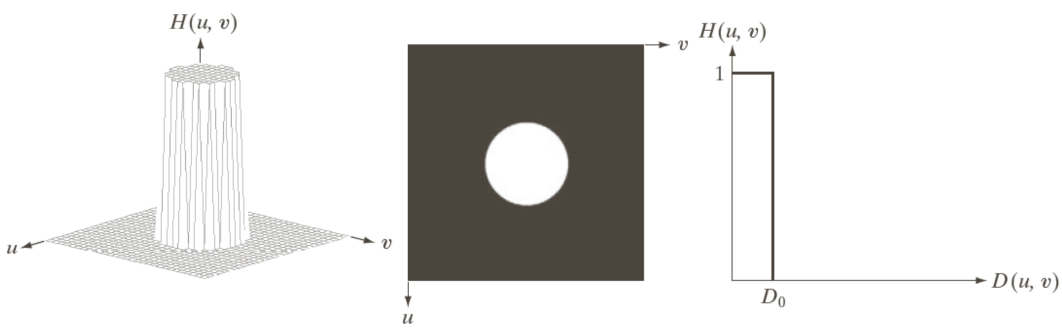
Input Image

$$f(x,y) \xrightarrow{\text{fft2}} F(u,v) \cdot H(u,v) = G(u,v) \xrightarrow{\text{ifft2}} g(x,y)$$

For example,

- Ideal lowpass filter

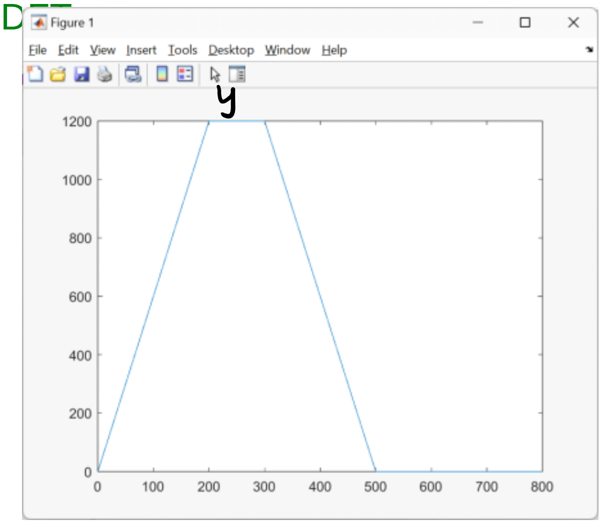
$$H(u,v) = \begin{cases} 1 & if\ D(u,v) \le D_0 \\ 0 & if\ D(u,v) > D_0 \end{cases}$$



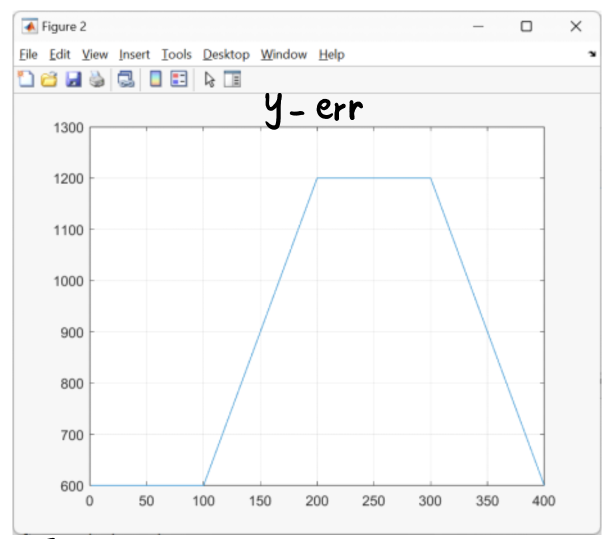
If we elect to compute the spatial convolution using the IDFT of the product of the two transforms, then the periodicity issue must be taken into account.

http://www.ece.uah.edu/~dwpan/course/ee604/code/ch4/fig4_28.m

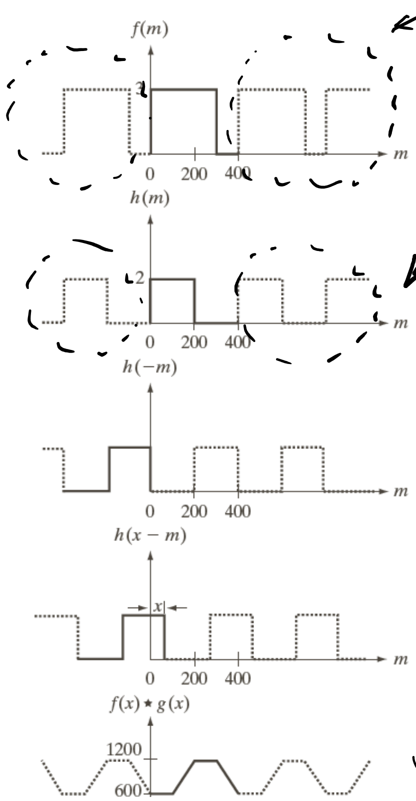
```
% Fig. 4.28
% Wraparound error due to periodicity implied by DFT

% Spatial domain convolution
f = zeros(1, 400);
f(1: 300) = 3;
h = zeros(1, 400);
h(1: 200) = 2;
y = conv(f, h);
plot(y);
```

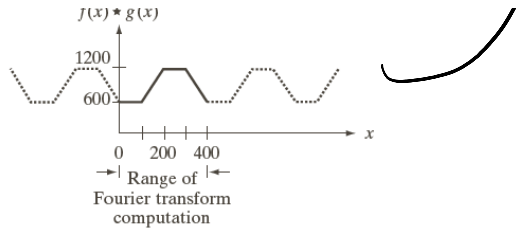



```
% DFT domain filtering with wraparound error
F = fft(f);
H = fft(h);
Y = F .* H;
y_err = ifft(Y);
figure; plot(y_err);
```

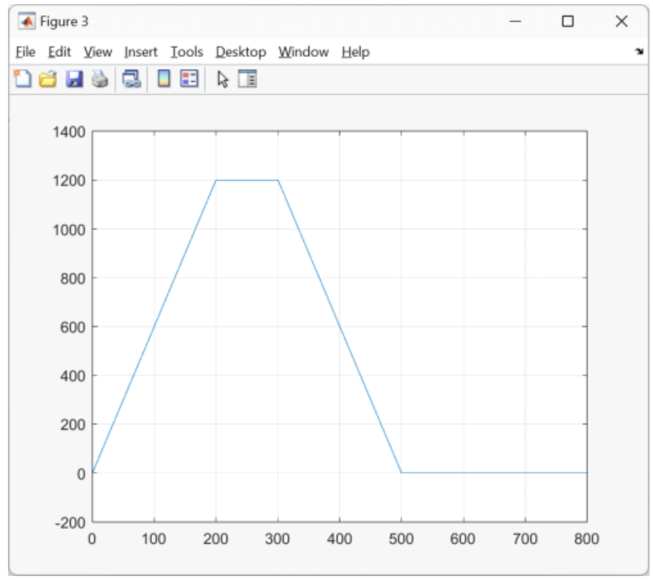


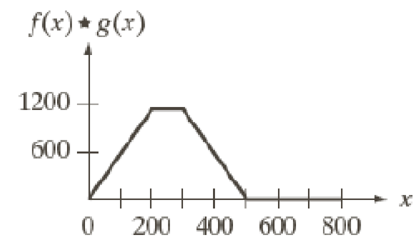
Fix: zero-padding




f(x) ★ g(x)
1200
600
0
200
400
x
Range of
Fourier transform
computation

```
% Zero padding to avoid wraparound error
% P >= A + B - 1 = 400 + 400 - 1 = 799
% Thus append 399 zeros to both f and h
fp = [f, zeros(1, 399)];
hp = [h, zeros(1, 399)];
Fp = fft(fp);
Hp = fft(hp);
Yp = Fp .* Hp;
yp = ifft(Yp);
figure; plot(yp);
```





$$f(x) \star h(x) = \sum_{m=0}^{399} f(x)h(x - m)$$

Zero Padding in 2D Image Filtering

$$f_p(x, y) = \begin{cases} f(x, y) & 0 \le x \le A-1 \quad \text{and} \quad 0 \le y \le B-1 \\ 0 & A \le x \le P \quad \text{or} \quad B \le y \le Q \end{cases}$$

$$h_p(x, y) = \begin{cases} h(x, y) & 0 \le x \le C-1 \quad \text{and} \quad 0 \le y \le D-1 \\ 0 & C \le x \le P \quad \text{or} \quad D \le y \le Q \end{cases}$$

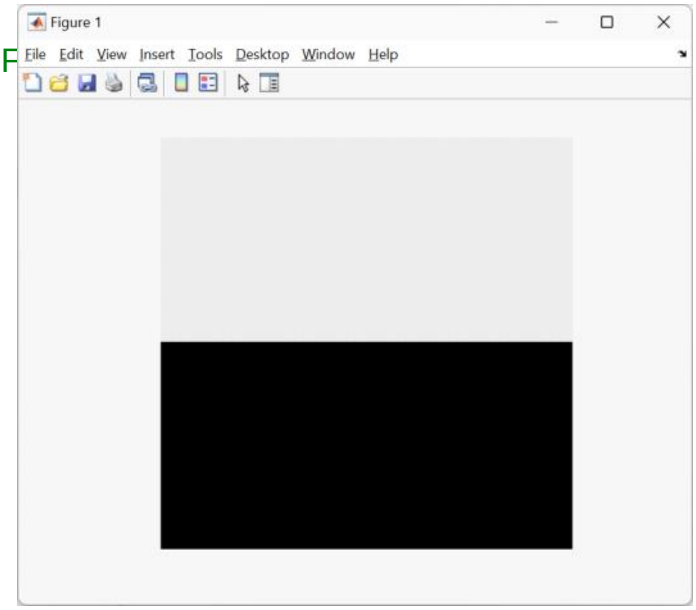
$$P \ge A + C - 1 \qquad\qquad Q \ge B + D - 1$$

- The resulting padded images are of size $P \times Q$.
- As a rule of thumb, DFT algorithms tend to execute faster with arrays of even size, so it is good practice to select P and Q as the smallest even integers that satisfy the preceding equations.
- If the two arrays are of the same size, then P and Q are selected as twice the array size.

http://www.ece.uah.edu/~dwpan/course/ee604/code/ch4/paddedsize.m

```
function PQ = paddedsize(AB, CD, PARAM)
```

```
% Fig. 4.32
% Wraparound error due to periodicity implied by DF
clear all;
close all;
f = imread('Fig0432(a)(square_original).tif');
imshow(f,'initialmagnification','fit');
```



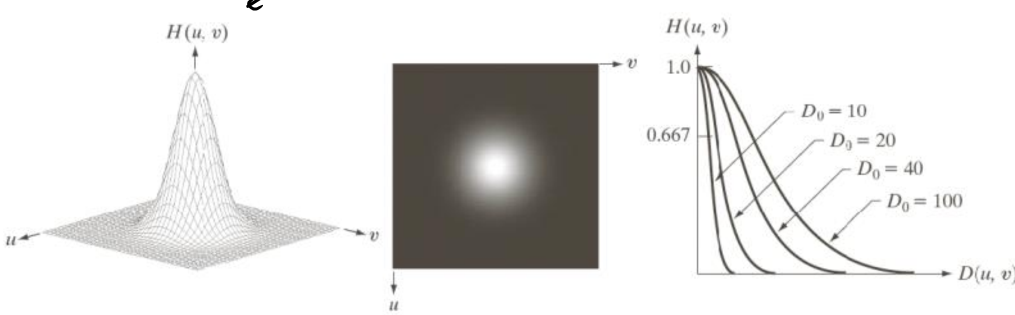

```
>> size(f)
ans =
  768   768

>> PQ = paddedsize(size(f))
PQ =
      1536       1536
```

- Gaussian lowpass filter

$$H(u,v) = e^{-\frac{D^2(u,v)}{2\sigma^2}}$$

Gaussian Filter Transfer Function

Centered transfer function



a b c

FIGURE 4.47 (a) Perspective plot of a GLPF transfer function. (b) Filter displayed as an image. (c) Filter radial cross sections for various values of D_0.

http://www.ece.uah.edu/~dwpan/course/ee604/code/ch4/dftuv.m

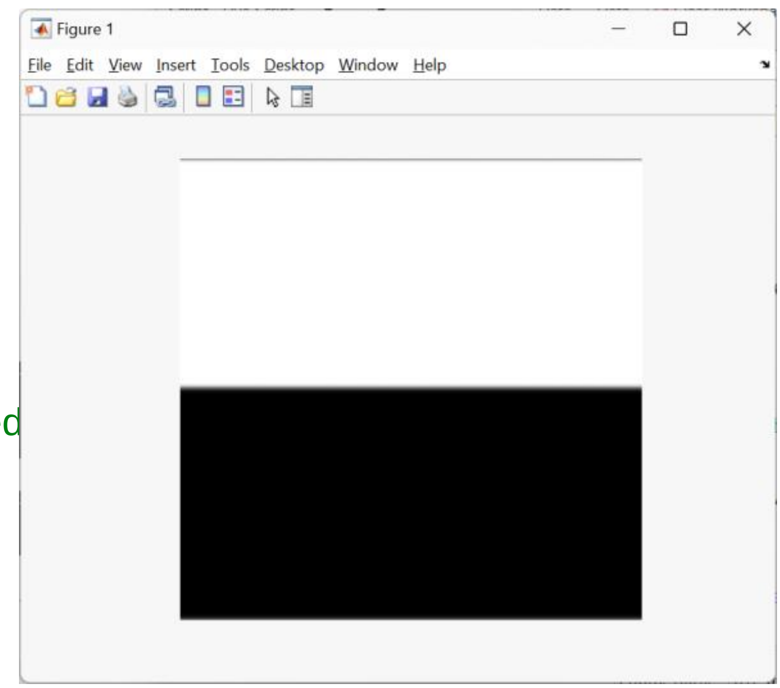
```
function [U, V] = dftuv(M, N)
%DFTUV Computes meshgrid frequency matrices.
%   [U, V] = DFTUV(M, N) computes meshgrid frequency matrices U and
%   V.  U and V are useful for computing frequency-domain filter
%   functions that can be used with DFTFILT.  U and V are both
%   M-by-N.
```

http://www.ece.uah.edu/~dwpan/course/ee604/code/ch4/lpfilter.m

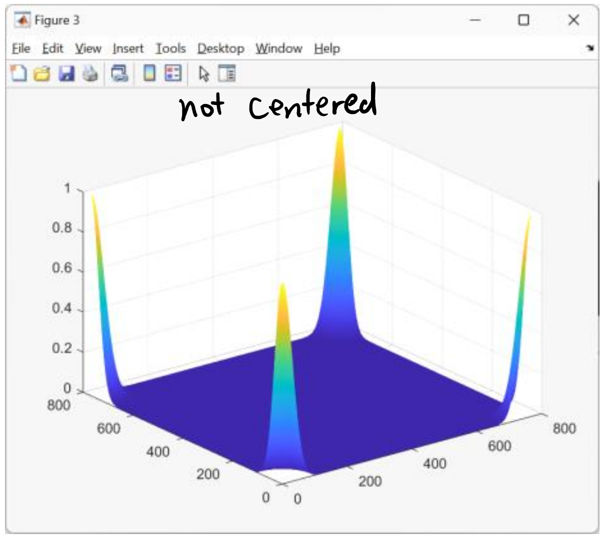
```
function H = lpfilter(type, M, N, D0, n)
%LPFILTER Computes frequency domain lowpass filters
%   H = LPFILTER(TYPE, M, N, D0, n) creates the transfer function of
%   a lowpass filter, H, of the specified TYPE and size (M-by-N).  To
%   view the filter as an image or mesh plot, it should be centered
%   using H = fftshift(H).

%  'gaussian' Gaussian lowpass filter with cutoff (standard deviation)
%              D0.  n need not be supplied.  D0 must be positive.

% Use function dftuv to set up the meshgrid arrays needed for
% computing the required distances.
[U, V] = dftuv(M, N);
% Compute the distances D(U, V).
D = sqrt(U.^2 + V.^2);

% Begin fiter computations.
switch type

case 'gaussian'
   H = exp(-(D.^2)./(2*(D0^2)));
```

σ

```
% Implementation with DFT not centered
F = fft2(f);   ← spectrum not centered
H = lpfilter('gaussian', M, N, sigma);
G = F .* H;
g = real(ifft2(G));
imshow(g, [], 'initialmagnification','fit');
```

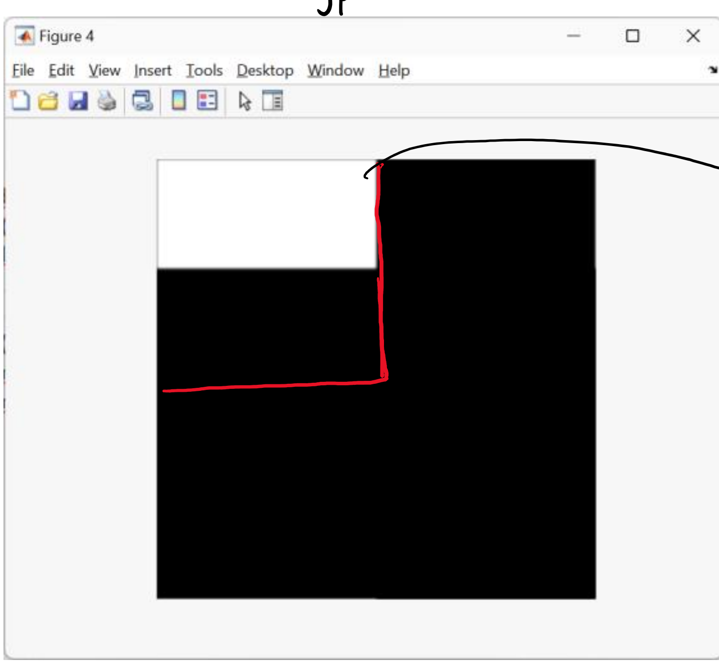


>> figure; mesh(H)




```
% With zero padding
PQ = paddedsize(size(f));
Fp = fft2(f, PQ(1), PQ(2));
Hp = lpfilter('gaussian', PQ(1), PQ(2), 2*sigma);
Gp = Hp .* Fp;
gp = real(ifft2(Gp));
gpc = gp(1:size(f,1), 1:size(f,2));
figure; imshow(gp, [], 'initialmagnification','fit');
figure; imshow(gpc, [], 'initialmagnification','fit');
```

gp



gpc