### Question 1



**Question 2**

function g = MotionBlurInFrequencyDomain(f, h)

% g = MotionBlurInFrequencyDomain(f, h)

% 'f' is original image, 'h' is kernel of motion blur and 'g' is degraded image

% All inputs and output is in spatial domain but your process has to be in frequency domain

% implematation here

% Take the 2D Fourier Transforms of the image and kernel

F = fft2(f);

H = fft2(h, size(f, 1), size(f, 2));

% Perform element-wise multiplication in the frequency domain

G = F .\* H;

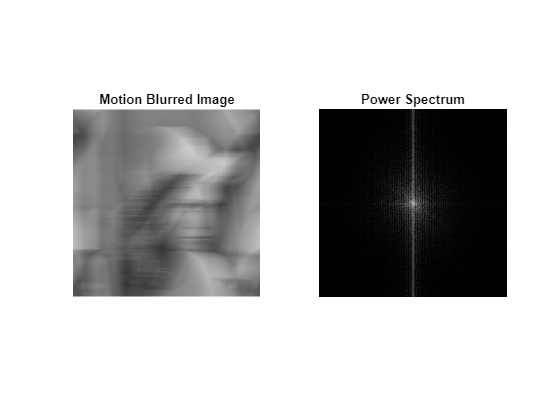
% Take the inverse Fourier Transform to get the degraded image

g = ifft2(G);

% Ensure the output is real (due to potential numerical errors)

g = real(g);

end



### Question 3

function f\_hat = InverseFilterInFrequencyDomain(g, h, epsilon)

% f\_hat = InverseFilterInFrequencyDomain(g, h, epsilon)

% 'g' is degraded image, 'h' is kernel of motion blur

% All inputs and output is in spatial domain but your process has to be in frequency domain

% implematation here

G = fft2(g);

H = fft2(h, size(g, 1), size(g, 2));

R = ones(size(H));

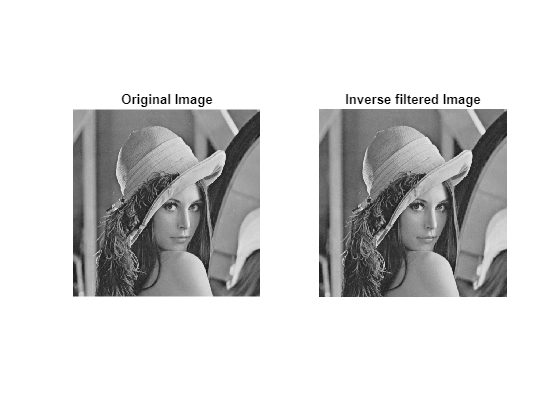
R(abs(H) > epsilon) = 1./H(abs(H) > epsilon);

F\_hat = R .\* G;

f\_hat = ifft2(F\_hat);

f\_hat = real(f\_hat);

end



### Question 4

function snr = ComputeSpectrumSNR(f, n)

% abs(N(u,v)).^2 / abs(N(u,v)).^2

% please copy from PDF file

F = fftshift(fft2(f));

N = fftshift(fft2(n));

S\_n = sum((abs(N).^2), 'all');

S\_f = sum((abs(F).^2), 'all');

snr = S\_f./S\_n;

end

function f\_hat = MyWienerFilter(g, h, K, epsilon)

% f\_hat = MyWiennerFilter(g, f, h, n, epsilon)

% 'g' is degraded image, 'f' is original image, 'h' is kernel of motion

% blur, 'n' is noise, epsilon is for R(u,v)

% All inputs and output is in spatial domain but your process has to be in frequency domain

% implematation here

% Take the 2D Fourier Transforms of the degraded image and kernel

G = fft2(g);

H = fft2(h, size(g, 1), size(g, 2));

% Calculate the Wiener filter in the frequency domain

R = ones(size(H));

R(abs(H) > epsilon) = 1 ./ H(abs(H) > epsilon);

F\_hat = R .\* abs(H).^2 ./ (abs(H).^2 + K) .\* G;

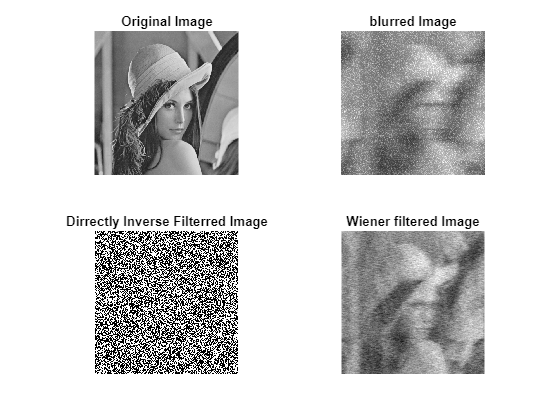
% Take the inverse Fourier Transform to get the estimated image

f\_hat = ifft2(F\_hat);

% Ensure the output is real (due to potential numerical errors)

f\_hat = real(f\_hat);

end



### Compare with different K

K = [1000, 0.0001, 0]; % pick 3 values of K yourself

figure();

subplot(2,2,1); imshow(f); title('Original Image');

for i = 1:3

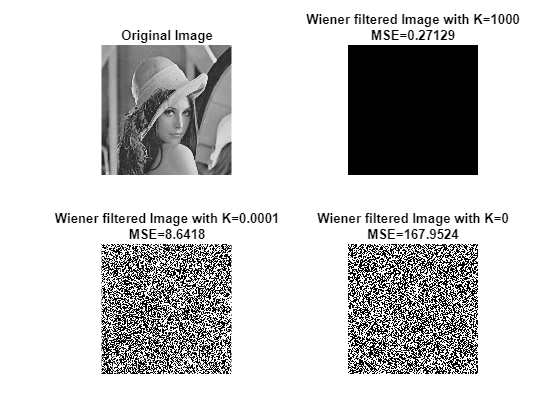
f\_hat = MyWienerFilter(g, h, K(i), 0.001);

mse = sum((f - f\_hat).^2, 'all') / numel(f);

subplot(2,2,i+1); imshow(f\_hat);

title({['Wiener filtered Image with K=' num2str(K(i)) ];['MSE=' num2str(mse)]}) % implematation here for K and MSE

end



**If K goes too small, the effect will be the same as inverse filtering and amplify the noise.**

**If K goes too large, the value of the resulting intensity approach to zero since the denominator goes infinity large relative to the numerator.**