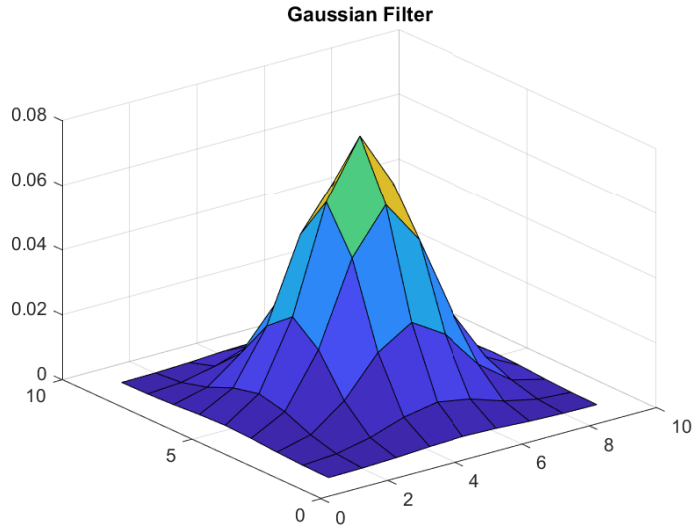
Contents

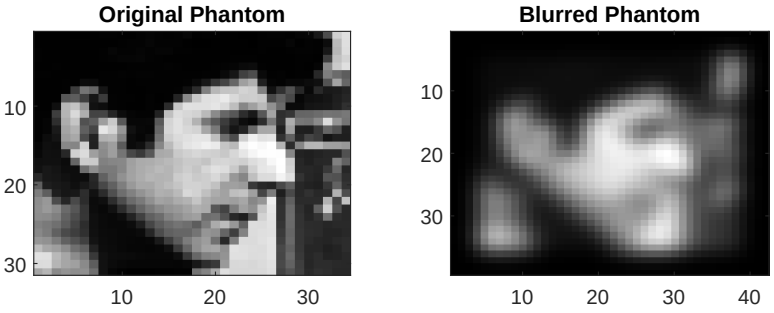
Setup

```
close all;
clc;
```

Part a

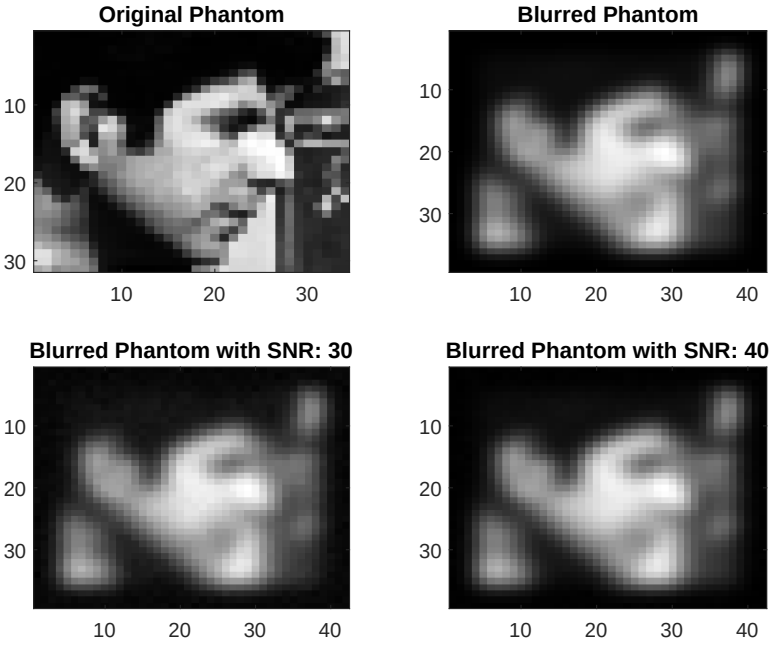
```
I = imread('cameraman.tif');
X = im2double(I);
X = X(50:80,105:138);
F = gauss2d(9,9,0,0,2,2);
figure
surf(F)
title('Gaussian Filter')
C = convmtx2(F,size(X,1),size(X,2));
X_vec = X(:);
y = C*X_vec;
f1 = figure;
ax = gca;
subplot(2,2,1,ax)
imagesc(X), colormap gray
title('Original Phantom')
subplot(2,2,2)
Y_blurred = reshape(y,size(F)+size(X)-1);
imagesc(Y_blurred)
title('Blurred Phantom')
```





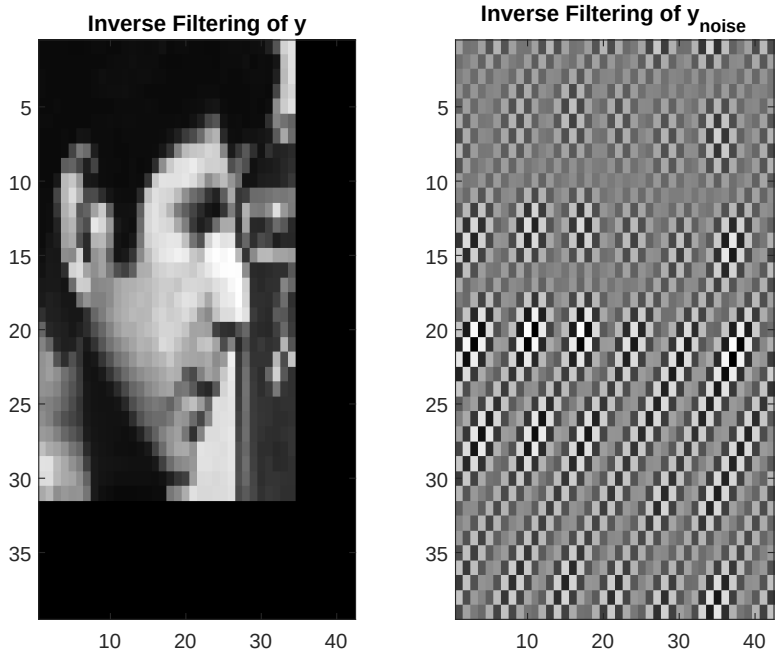
Part b

```
for ratio = [3 4]
    variance = var(y)/10^ratio; % SNR adjustment
    sigma_n = sqrt(variance);
    noise = sigma_n * randn(size(y)); % N~(0,sigma_n^2)
    y_n = y + noise;
    y_img = reshape(y_n,size(F)+size(X)-1);
    subplot(2,2,ratio)
    imagesc(y_img)
    title(['Blurred Phantom with SNR: ',num2str(10*ratio)])
end
```



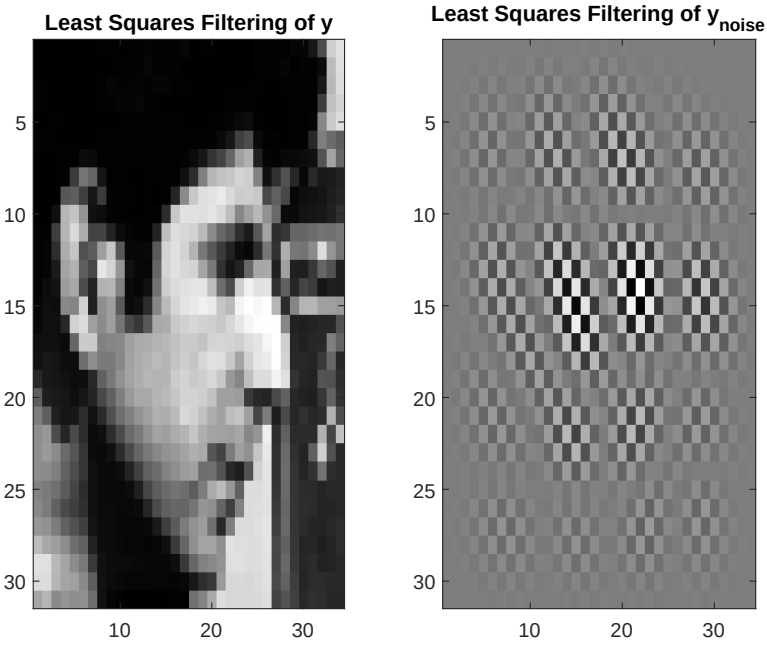
Part c: Inverse Filtering in Frequency domain

```
figure
size_Y = size(Y_blurred);
X_inv = ifft2(fft2(Y_blurred,size_Y(1), ...
    size_Y(2))./fft2(F,size_Y(1),size_Y(2)),size_Y(1),size_Y(2));
X_inv_noisy = ifft2(fft2(y_img)./fft2(F, ...
    size_Y(1),size_Y(2)),size_Y(1),size_Y(2));
subplot(1,2,1)
imagesc(X_inv), colormap gray
title('Inverse Filtering of y')
subplot(1,2,2)
imagesc(X_inv_noisy)
title('Inverse Filtering of y_{noise}')
```



Part d: LS Filtering

```
figure
C = full(C); % The LS solution outputs wrong results if C remains sparse!
X_inv = reshape(inv(C'*C)*C'*y,size(X));
X_inv_noisy = reshape(inv(C'*C)*C'*y_n,size(X));
subplot(1,2,1)
imagesc(X_inv), colormap gray
title('Least Squares Filtering of y')
subplot(1,2,2)
imagesc(X_inv_noisy)
title('Least Squares Filtering of y_{noise}')
```



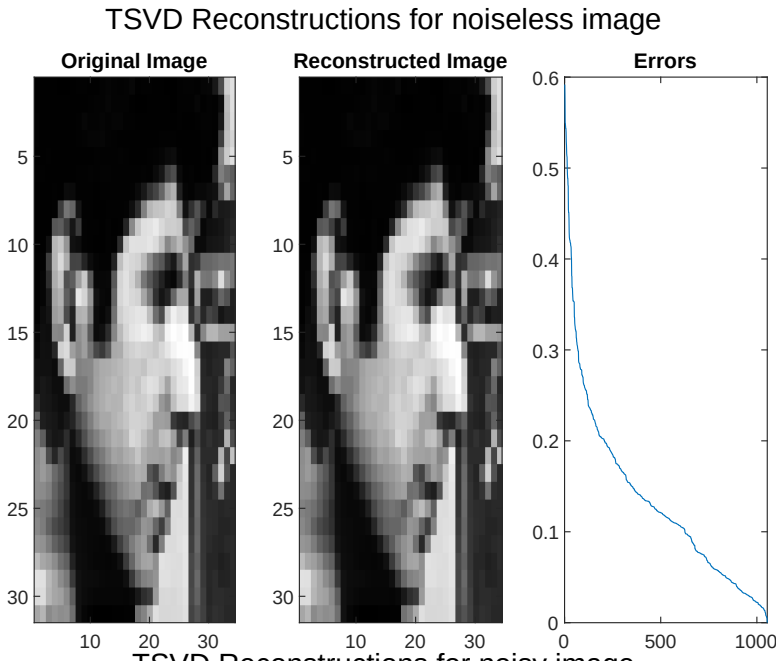
Part e

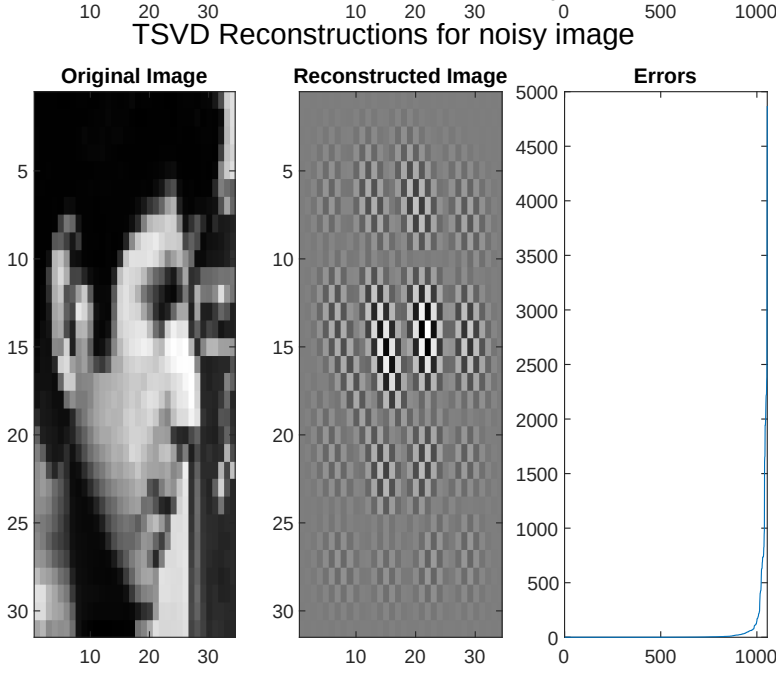
y = U S VH x; x = V 1/S UH y;

```
[U,S,V] = svd(C);
singular_values = diag(S);
X_back_vec = zeros(size(X_vec));
Errors = zeros(size(singular_values));
for i = 1:length(singular_values)
    s = singular_values(i);
    v = V(:,i);
    u = U(:,i);
    X_back_vec = X_back_vec + 1/s * v * u' * y;
    Errors(i) = relative_error(X_vec, X_back_vec);
end
X_back = reshape(X_back_vec,size(X));
figure
colormap gray
subplot(1,3,1)
imagesc(X)
title('Original Image')
subplot(1,3,2)
imagesc(X_back)
title('Reconstructed Image')
```

```
subplot(1,3,3)
plot(Errors)
title('Errors')
sgtitle('TSVD Reconstructions for noiseless image')

[U,S,V] = svd(C);
singular_values = diag(S);
X_back_vec = zeros(size(X_vec));
Errors = zeros(size(singular_values));
for i = 1:length(singular_values)
    s = singular_values(i);
    v = V(:,i);
    u = U(:,i);
    X_back_vec = X_back_vec + 1/s * v * u' * y_n;
    Errors(i) = relative_error(X_vec, X_back_vec);
end
Full_Errors = Errors;
X_back = reshape(X_back_vec,size(X));
figure
colormap gray
subplot(1,3,1)
imagesc(X)
title('Original Image')
subplot(1,3,2)
imagesc(X_back)
title('Reconstructed Image')
subplot(1,3,3)
plot(Errors)
title('Errors')
sgtitle('TSVD Reconstructions for noisy image')
```





Find threshold for noisy reconstruction

A ratio of $\frac{1}{200}$ in singular value amplitude provided a satisfactory amount of reconstruction.

```
f = figure; colormap gray;
f.Position = [100 100 600 800];
plot_idx = 4;
for ratio = 100:100:900
    max_sing_value = max(singular_values);
    threshold = max_sing_value / ratio;
    [minimum, index] = min(abs(singular_values-threshold));

    X_back_vec = zeros(size(X_vec));
    Errors = zeros(index,1);
    for i = 1:index
        s = singular_values(i);
        v = V(:,i);
        u = U(:,i);
        X_back_vec = X_back_vec + 1/s * v * u' * y_n;
        Errors(i) = relative_error(X_vec, X_back_vec);
    end
    X_back = reshape(X_back_vec,size(X));
    subplot(4,3,plot_idx)
    imagesc(X_back)
    title({['Ratio:',num2str(ratio)] ...
        ,['Min Eigenvalue index: ',num2str(index)]})
    plot_idx = plot_idx + 1;
end

colormap gray
subplot(4,3,1)
imagesc(X)
title('Original Image')

subplot(4,3,2)
plot(100:900,Full_Errors(100:900))
title('Errors')
xlabel('Singular Value Iteration')
ylabel('Relative Error')

subplot(4,3,3)
plot(singular_values)
title('Sorted Singular Value distribution')
sgtitle('TSVD Reconstructions for noisy image')
```

TSVD Reconstructions for noisy image