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')
```

Gaussian Filter

Original Phantom
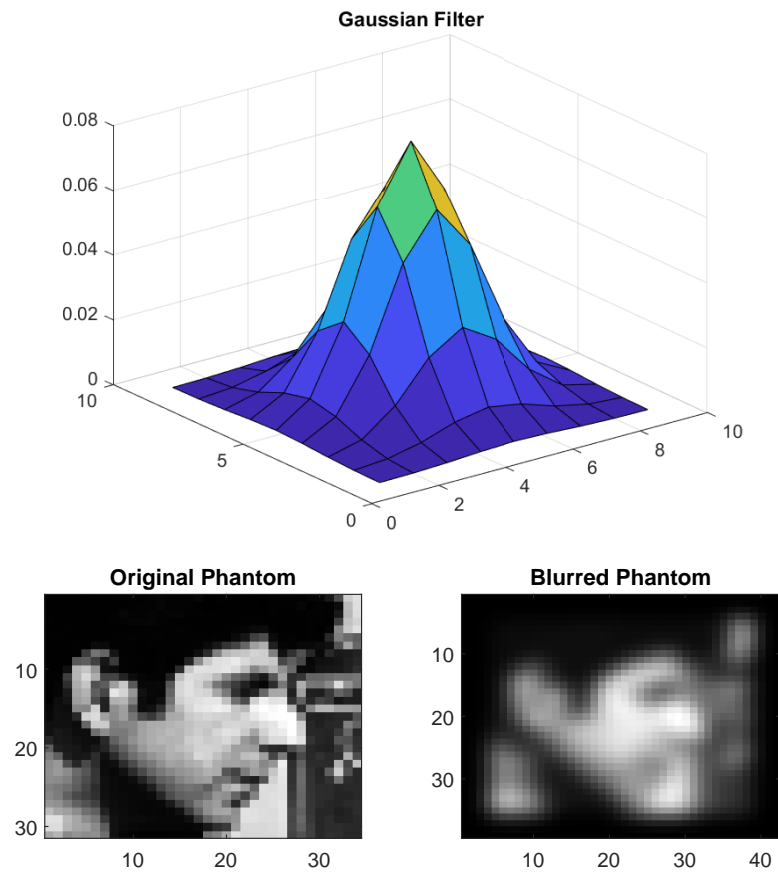
Blurred Phantom

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
```

Original Phantom   **Blurred Phantom**

Blurred Phantom with SNR: 30   **Blurred Phantom with SNR: 40**
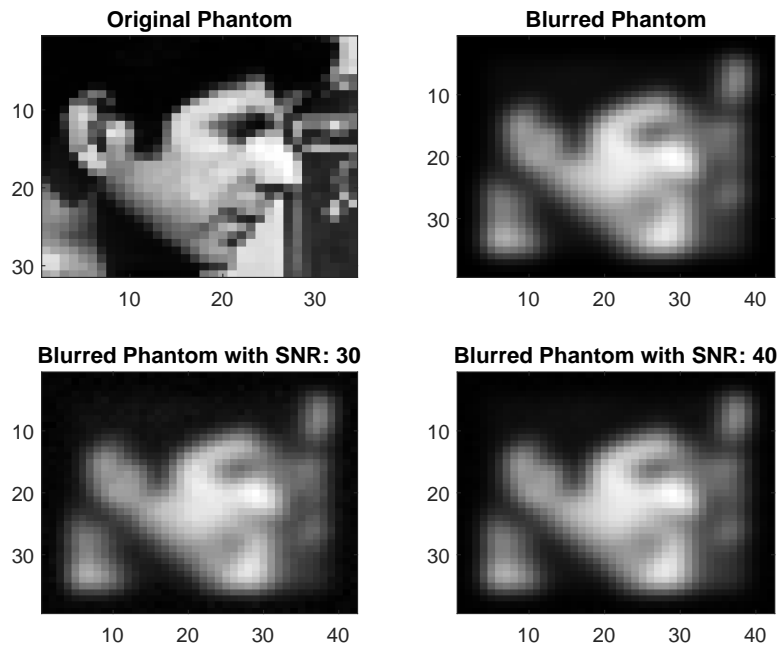
Comments

The effect of the noise is not observable to human eye under the effect of blurring, *i.e.* it is difficult to distinguish the noise from the blurred image.

## Part c: Inverse and LS 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}')

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}')
```

Inverse Filtering of y

Inverse Filtering of y$_{noise}$

Least Squares Filtering of y

Least Squares Filtering of y$_{noise}$
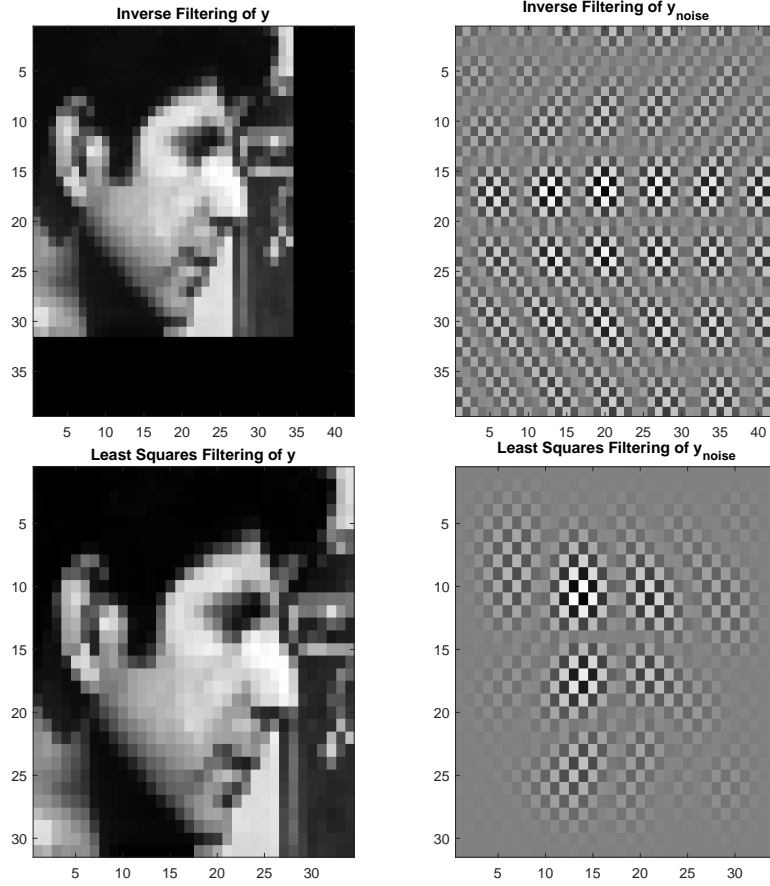
Comments

$$\tilde{y} \quad = y + n \tag{1}$$

$$A \quad = U\Sigma V^H \tag{2}$$

$$A^T \quad = V\Sigma U^H \tag{3}$$

$$\hat{x_{LS}} \quad = (A^T A)^{-1} A^T \tag{4}$$

Inserting Eqn. 2 and 3 in 4, results in following expression:

$$\hat{x_{LS}} = \underbrace{V\Sigma^{-1}U^H}_{A^\dagger}(y + n) \tag{5}$$

4

The least squares solution amplifies the noise with pseudo-inverse of A (in SVD sense) hence results in the checkerboard pattern in the reconstruction as an implication of amplified high frequency noise. Since A is an ill-conditioned matrix and hence has really small singular values, noise is amplified with $\frac{1}{\sigma}$ which is a huge amplification factor.

Part d

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));
f = figure;
f.Position = [100 100 900 300];
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;
```
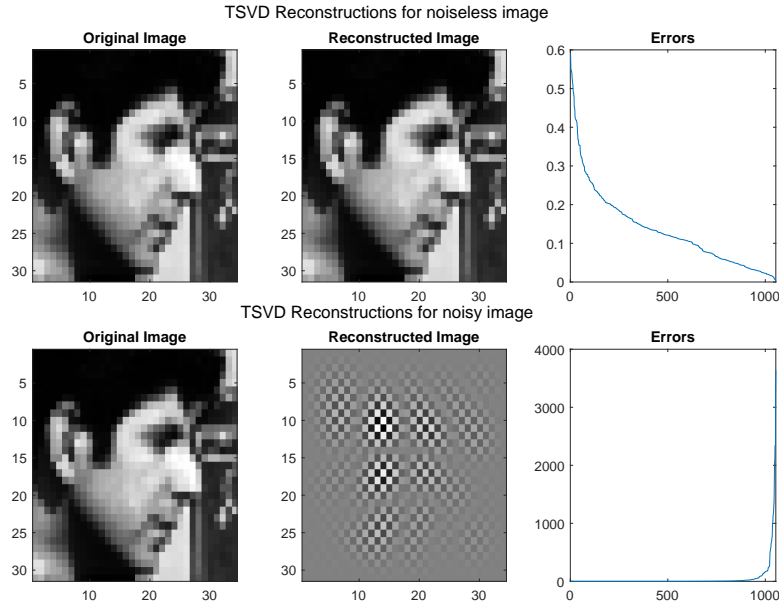
```
    Errors(i) = relative_error(X_vec, X_back_vec);
end
Full_Errors = Errors;
X_back = reshape(X_back_vec,size(X));
f = figure;
f.Position = [100 100 900 300];
colormap gray
subplot(1,3,1)
imagesc(X)
title('Original Image')
subplot(1,3,2)
imagesc(X_back)
title('Reconstructed Image')
subplot(1,3,3)
plot(Errors)
title('Errors')
sgtitle('TSVD Reconstructions for noisy image')
```


Comments

For this question a distance error metric is defined:

$$RelativeError = \frac{||\hat{x} - x||}{||x||} \tag{6}$$

In the figure above, we observe that error is growing after the 900th singular value. Hence, to find a suitable threshold, this interval is scanned.

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')
```
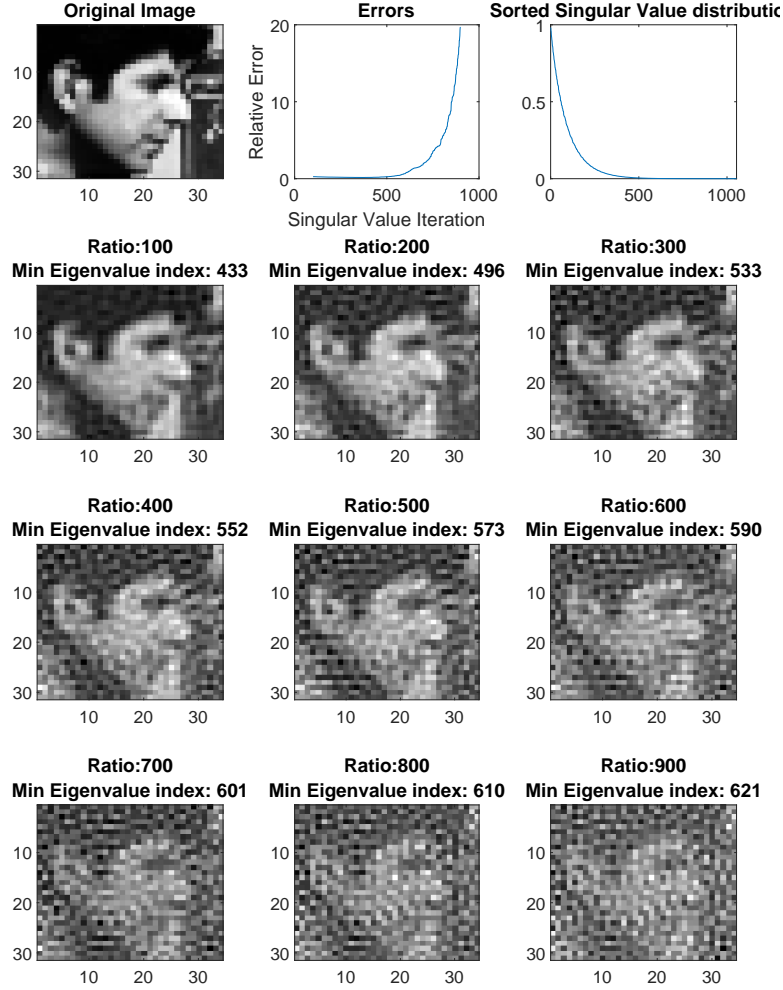
TSVD Reconstructions for noisy image

Original Image

Errors

Sorted Singular Value distribution

Ratio:100
Min Eigenvalue index: 433

Ratio:200
Min Eigenvalue index: 496

Ratio:300
Min Eigenvalue index: 533

Ratio:400
Min Eigenvalue index: 552

Ratio:500
Min Eigenvalue index: 573

Ratio:600
Min Eigenvalue index: 590

Ratio:700
Min Eigenvalue index: 601

Ratio:800
Min Eigenvalue index: 610

Ratio:900
Min Eigenvalue index: 621

Comments

Experiments showed that applying a threshold around the 1% of the maximum
singular value resulted in moderate success when compared to the other recon-
struction thresholds. The corresponding singular value index is found to be 433.
This can also be seen in the error plot given in the top of the above figure, where
error gets drastically higher after the inclusion of 500 singular values.

Conjugate Gradient

0.1 a

0.2 b

Comments

As stated, the main intensive task in the Conjugate gradient method is the repetitive forward operators. Tikhonov regularized solution can be found by assigning $A = (C^T C + \lambda D^T D)$. However, this may not be useful in memory-limited systems. Hence forward projections of A should be replaced with another operation requiring less memory. This can be done as follows:

$$
\begin{aligned}
Az \quad &= (C^T C + \lambda D^T D)z \qquad &(7)\\
&= C^T(Cz) + \lambda D^T(Dz) \qquad &(8)
\end{aligned}
$$

This type of formulation allows us to replace Az with forward operators Cz, $C^T z$, Dz and $D^T z$ for some arbitrary vector z. For some particular C and D, the quantities Cz and $C^T z$ in Eqn. 8 can be computed using fast-forward routines, keeping in mind that the operator C represents convolution operation, hence it can be computed via FFTs. Hence, one only needs to store the discrete derivative operator and using the seperability feature of the 2D discrete derivative operator, reduces memory requirements drastically.