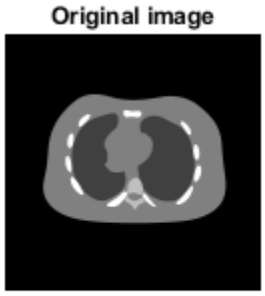
Table of Contents


```
clc;
clear;
tic;
```

Reading image

```
img = imread("../data/ChestPhantom.png");
figure;
imshow(img);
title("Original image");
img = im2double(img);
```



Solving Ax = B

```
% Stategy - To solve Ax = B for A, we used Moore-Penrose pseudoinverse
% method to calculate inverse for a column matrix (in this case, for
 x) and
% therefore, calculated the system matrix A.

[r1, c1] = size(img);
theta = 0:179;
radonTf = radon(img, theta);
[r2, c2] = size(radonTf);
```

```
% vectorized images
X = reshape(img, r1 * c1, 1);
B = reshape(radonTf, r2 * c2, 1);
% using pseudo-inverse of column matrix to solve equation
A = B * pinv(X);
% new radon projection
newRadonTf = A * X;
newRadonTf = reshape(newRadonTf, r2, c2);
```

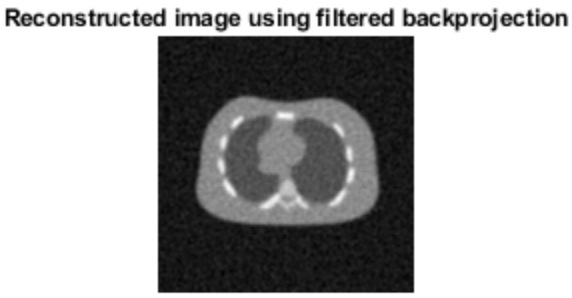
Adding noise to the sinogram

```
% add Gaussian noise
minVal = min(min(newRadonTf));
maxVal = max(max(newRadonTf));
range = maxVal - minVal;
noisyRadonTf = newRadonTf + 0.02 * range * randn(size(newRadonTf));
```

Filtered backprojection

```
% using Cosine filter
filteredTf = myFilter(noisyRadonTf, 1, 3);
bp = iradon(filteredTf, theta, 'linear', 'none', 1, 128);
minVal = min(min(bp));
maxVal = max(max(bp));
bp_min = minVal * ones(size(bp));
bp_max = maxVal * ones(size(bp));
bp = (bp - bp_min) ./ (bp_max - bp_min);
imshow(uint8(255 * bp));
title("Reconstructed image using filtered backprojection");
rrmse = RRMSE(img, bp);
fprintf("RRMSE using Cosine filter = %f\n\n", rrmse);
```

RRMSE using Cosine filter = 0.340415



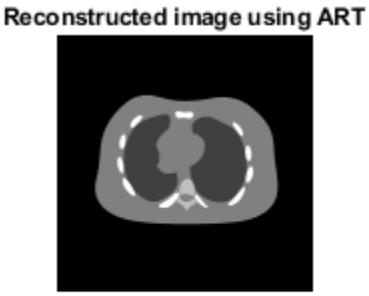
Tikhonov regularized reconstruction

```
range = 10e-20:0.01:1;
```

```
vectNRTf = reshape(noisyRadonTf, r2 * c2, 1);
[reconstructedX, errors] = tikhonov(A, vectNRTf, img, range);
figure;
imshow(uint8(255 * reconstructedX));
title("Reconstructed image using ART");

ind = find(min(errors));
error_min = errors(ind);
fprintf("RRMSE between noiseless and reconstructed img :- \n");
fprintf("At optimal lambda = %f\n", error_min);
range1 = 0.8 * range(ind);
[reconstructedX_0_8, errors_0_8] = tikhonov(A, B, img, range1);
fprintf("RRMSE between noiseless and reconstructed img :- \n");
fprintf("At 0.8 * optimal lambda = %f\n", errors_0_8);
range2 = 1.2 * range(ind);
[reconstructedX_1_2, errors_1_2] = tikhonov(A, B, img, range2);
fprintf("RRMSE between noiseless and reconstructed img :- \n");
fprintf("At 1.2 * optimal lambda = %f\n\n", errors_1_2);
```

```
RRMSE between noiseless and reconstructed img :-
At optimal lambda = 0.000000
RRMSE between noiseless and reconstructed img :-
At 0.8 * optimal lambda = 0.000000
RRMSE between noiseless and reconstructed img :-
At 1.2 * optimal lambda = 0.000000
```



Reconstruction using MRF priors

```
groundTruth = img;
b = radonTf;
angle = theta;
b_noisy = noisyRadonTf;
```

Quadratic

```
alpha = 0.1;
gamma = 0;
```

```
[denoised_b1, loss1] = gradientDescent(b_noisy, alpha, gamma, 1);
Filtered_b1 = myFilter(denoised_b1, 1,3);
ReconstructedSignal1 = iradon(Filtered_b1,
 angle, 'linear','none',1,128);
mx=max(max(ReconstructedSignal1));
mn = min(min(ReconstructedSignal1));
bp_min =ones(size(ReconstructedSignal1)).*mn;
bp_max =ones(size(ReconstructedSignal1)).*mx;
ReconstructedSignal1 = ((ReconstructedSignal1)-bp_min) ./ (bp_max-
bp_min);
newRRMSE1 = RRMSE(groundTruth, ReconstructedSignal1);
fprintf("Optimal alpha = %f\n", alpha);

fprintf("Quadratic errors\n------------------\n");

fprintf("RRMSE between noiseless and reconstructed img :- \n");
fprintf("At alpha = %f\n\n", newRRMSE1);
figure,
imshow(uint8(255 *
 ReconstructedSignal1)),daspect([1,1,1]),colormap('gray')
title('Reconstructed ChestPhantom | Quadratic Function')
```

```
Optimal alpha = 0.100000
Quadratic errors
------------------
RRMSE between noiseless and reconstructed img :-
At alpha = 0.275946
```

Huber

```
alpha = 0.6;
gamma = 0.02;
[denoised_b2, loss2] = gradientDescent(b_noisy, alpha, gamma, 2);
Filtered_b2 = myFilter(denoised_b2, 1,3);
ReconstructedSignal2 =
 iradon(Filtered_b2,angle,'linear','none',1,128);
mx=max(max(ReconstructedSignal2));
```

```
mn = min(min(ReconstructedSignal2));
bp_min =ones(size(ReconstructedSignal2)).*mn;
bp_max =ones(size(ReconstructedSignal2)).*mx;
ReconstructedSignal2 = ((ReconstructedSignal2)-bp_min) ./ (bp_max-
bp_min);
newRRMSE2 = RRMSE(groundTruth, ReconstructedSignal2);
fprintf("Optimal alpha = %f\n", alpha);

fprintf("Huber errors\n-----------------\n");

fprintf("RRMSE between noiseless and reconstructed img :- \n");
fprintf("At alpha = %f\n\n", newRRMSE2);
figure;
imshow(uint8(255 *
 ReconstructedSignal2)),daspect([1,1,1]),colormap('gray');
title('Reconstructed ChestPhantom | Huber Function');
```

```
Optimal alpha = 0.600000
Huber errors
-----------------
RRMSE between noiseless and reconstructed img :-
At alpha = 0.335044
```

DAF

```
alpha = 0.4;
gamma = 0.065;
[denoised_b3, loss3] = gradientDescent(b_noisy, alpha, gamma, 3);
Filtered_b3 = myFilter(denoised_b3, 1,3);
ReconstructedSignal3 =
 iradon(Filtered_b3,angle,'linear','none',1,128);
mx=max(max(ReconstructedSignal3));
mn = min(min(ReconstructedSignal3));
bp_min =ones(size(ReconstructedSignal3)).*mn;
bp_max =ones(size(ReconstructedSignal3)).*mx;
ReconstructedSignal3 = ((ReconstructedSignal3)-bp_min) ./ (bp_max-
bp_min);
```

```
newRRMSE3 = RRMSE(groundTruth, ReconstructedSignal3);
fprintf("Optimal alpha = %f\n", alpha);

fprintf("DAF errors\n------------------\n");

fprintf("RRMSE between noiseless and reconstructed img :- \n");
fprintf("At alpha = %f\n\n", newRRMSE3);
figure;
imshow(uint8(255 *
 ReconstructedSignal3)),daspect([1,1,1]),colormap('gray');
title('Reconstructed ChestPhantom | DAF Function');
```

```
Optimal alpha = 0.400000
DAF errors
------------------
RRMSE between noiseless and reconstructed img :-
At alpha = 0.331004
```


```
toc;
```

```
Elapsed time is 7.869602 seconds.
```

Published with MATLAB® R2019b