Dept. Of Electrical & Electronic Engineering (EEE)

EEE 4331 (A): Biomedical Engineering

Lab Work: Computing Tomography - Reconstructing an Image from Projection Data

Submitted by

Ibrahim Khalil	**021 191 020**

email: ikhalil191020@bseee.uiu.ac.bd

Submitted to

Prof. Dr. Khawza Iftekhar Uddin Ahmed (KIUA)

Professor at dept. of EEE (UIU)

Date of Submission

05.09.2023

Introduction: This example demonstrates the use of the radon and ‘iradon’ functions to create projections from an initial image and subsequently reconstruct the image from these projections. The radon and ‘iradon’ functions employ a parallel-beam configuration for the projection process.

In practical applications, one common use of image reconstruction is in X-ray absorption tomography, such as in CT (Computed Tomography) scans. In this context, projections are generated by measuring how radiation passing through a physical sample is attenuated at various angles. The original image can be visualized as a cross-sectional view of the specimen, where the intensity values correspond to the density of the material.

Specialized medical imaging equipment captures these projections, which are then used to reconstruct an internal image of the specimen through the ‘iradon’ function.

‘iradon is responsible for reconstructing an image based on parallel-beam projections. In this parallel-beam configuration, each projection is created by combining a set of line integrals across the image at a specific angle. This approach allows us to recover valuable information about the internal structure of the specimen, akin to a virtual "slice" through it.

This technique finds widespread use in medical diagnostics and research, enabling the non-invasive examination of internal structures and assisting in the diagnosis of various medical conditions.

Create Head Phantom

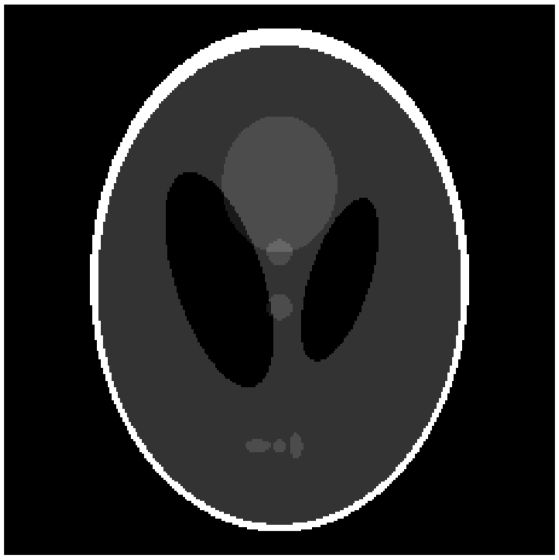
A "phantom" image in medical imaging refers to a synthetic image produced that copies the characteristics of actual human tissues or structures, notably in the setting of computed tomography (CT). The term "phantom" is used since it does not reflect a genuine human subject but is instead created to carefully replicate the activity of actual tissues.

Code:

```
clear all
close all
P = phantom(256);
imshow(P)
```

Observation: The phantom(256) function in the sample code creates a phantom image known as the "Shepp-Logan head phantom." Because it mimics a variety of properties seen in actual human head scans, this phantom picture is frequently utilized in the field of medical imaging for testing and calibration purposes.

Output:



Observation: Here's what the different parts of the Shepp-Logan head phantom represent:

- **Bright Elliptical Shell:** This outer shell represents the skull. It is bright because X-rays are attenuated (absorbed) less by bone, leading to higher intensity in the image.
- **Ellipses Inside:** The smaller ellipses inside the head phantom represent different brain features, tissues, or abnormalities that one might encounter in a real CT scan. These ellipses are used to simulate the varying densities and shapes of brain structures.

Overall, without the need for actual patient data, the Shepp-Logan head phantom offers a controlled environment for analyzing and testing CT image reconstruction algorithms, image enhancement methods, and many elements of image processing. It is an important tool for study in medical imaging.

Parallel Beam - Calculate Synthetic Projections

Generate synthetic projections using a parallel-beam setup while adjusting the number of projection angles. Each time you use the radon function, you get a matrix where each column corresponds to the Radon transform for a specific angle from the theta array.

Code:

```
theta1 = 0:10:170;
[R1,xp] = radon(P,theta1);
num_angles_R1 = size(R1,2)
num_angles_R1 = 18
```

Output:

```
>> parallel_beam_1
num_angles_R1 =
    18
```

Observation: It seems like here successfully calculated synthetic projections using the radon function for a range of projection angles. You obtained 18 projection angles with increments of 10 degrees each. These projections are stored in the matrix R1, where each column represents the Radon transform for one of the angles in the theta1 array.

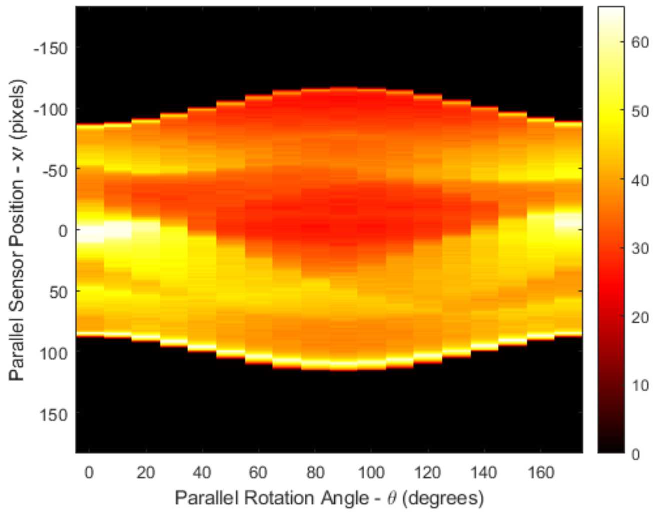
Visualization of the projections

Display the projection data stored in the matrix R1. Notably, some details present in the original phantom image aren't as clear in this data. In R1, the initial column corresponds to a projection taken at a 0-degree angle, essentially capturing data in the vertical direction. Conversely, the middle column represents a projection at a 90-degree angle, focusing on horizontal directions. The 90-degree projection appears broader compared to the 0-degree projection, primarily because of the outermost ellipse's significant vertical semi-axis in the phantom.

Code:

```
figure, imagesc(theta1,xp,R1)
colormap(hot)
colorbar
xlabel('Parallel Rotation Angle - \theta (degrees)');
ylabel('Parallel Sensor Position - x\prime (pixels)');
```

Output:



Observation: Let's break down the output:

- **figure:** This command creates a new figure for displaying the projection data.
- **imagesc(theta1,xp,R1):** This line creates an image-like visualization of the data stored in R1 using the given angles theta1 on the x-axis and sensor positions xp on the y-axis. The values in R1 determine the color intensity at specific points on this plot.
- **colormap(hot):** It sets the colormap to "hot," which is a colormap that represents lower values with cooler colors (e.g., blue) and higher values with warmer colors (e.g., red).
- **colorbar:** This command adds a color scale to the side of the plot, allowing you to interpret the color-coding of intensity values.

The output plot allows to visualize how the intensity values change as the projection angles vary and how they relate to different sensor positions. It helps to understand the information captured by the Radon transform and the effects of varying projection angles on the resulting data.

Submission:

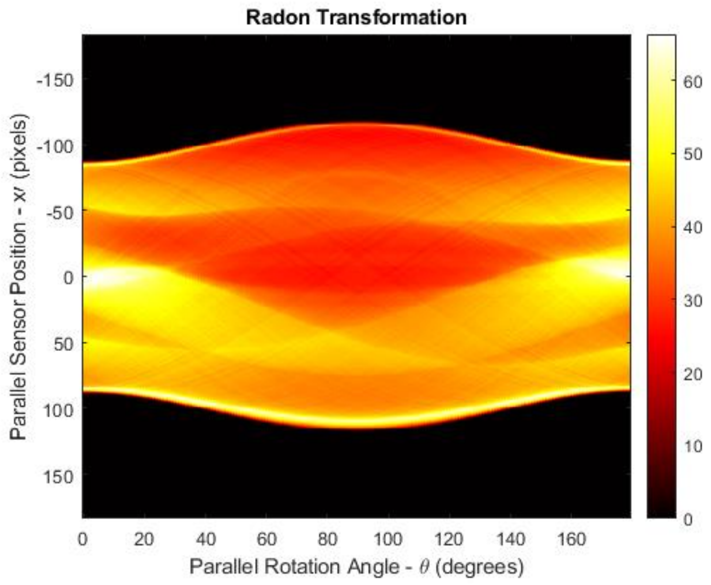
- **Include the projection image, i.e., Radon transformation g(θ, ρ) as Figure-2 .**

  Code:

```
% Load the Shepp-Logan phantom image
P = phantom(256);

% Define the projection angles
theta = 0:1:179;
```

```
% Compute synthetic projections using radon
[R, xp] = radon(P, theta);

% Display the Radon transformation
figure;
imagesc(theta, xp, R);
colormap(hot);
colorbar;
xlabel('Parallel Rotation Angle - \theta (degrees)');
ylabel('Parallel Sensor Position - x\prime (pixels)');
title('Radon Transformation');
```

Output:



- **What does xp represent in the code?**

xp represents the parallel sensor position. It is an array that contains the positions of the sensors or detectors that measure the X-ray intensity during the Radon transformation. These positions are along a line that runs parallel to the rotation axis.

When you perform a Radon transformation, the radon function returns both the Radon transform R, which is a function of both the projection angle (theta) and the sensor position (xp). The xp values represent the distances from the rotation axis to the sensors or detectors used to measure X-ray attenuation at various angles.

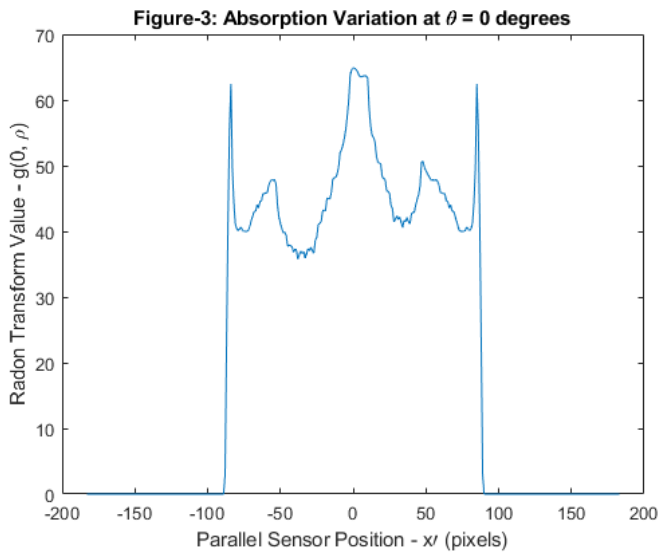
In summary, xp is an array that stores the sensor positions along the line of measurements during the Radon transformation.

- **Provide absorption variation at θ=0 degrees and plot g(θ, ρ) as a function of ρ and include the plot as Figure-3. Justify the obtained plot, i.e., projection.**

Code:

```
% Define the desired projection angle
theta_desired = 0;  % Angle in degrees

% Calculate the Radon transformation for the desired angle
[R_theta0, xp_theta0] = radon(P, theta_desired);

% Plot g(?, ?) as a function of ? for ? = 0 degrees (Figure-3)
figure;
plot(xp_theta0, R_theta0);
xlabel('Parallel Sensor Position - x\prime (pixels)');
ylabel('Radon Transform Value - g(0, \rho)');
title('Figure-3: Absorption Variation at \theta = 0 degrees');
```

Output:



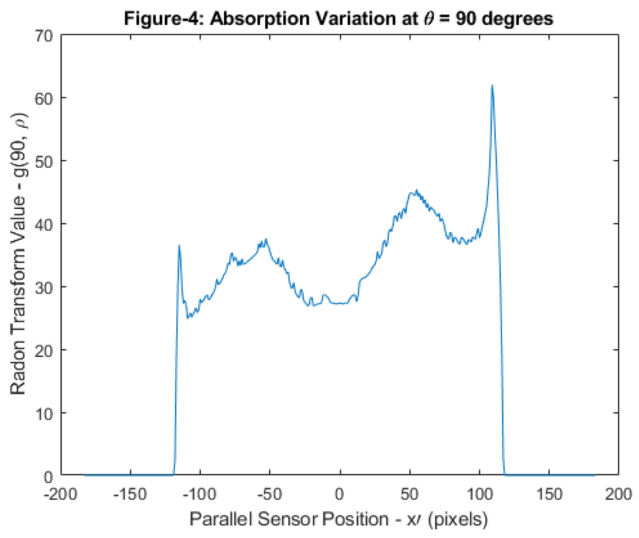
Observation: This code calculates the Radon transformation for θ=0 degrees and plots g(0, ρ) as a function of ρ. The resulting plot will be displayed as Figure-3. This plot shows the absorption variation along the sensor position for the specified projection angle (0 degrees). The values in this plot represent the Radon transform at θ=0 degrees and provide information about how X-rays are absorbed as they pass through the object at this angle.

- **Provide absorbtion or attenuation variation at θ=90 degree and plot g(θ, ρ) as function of ρ and include the plot as Figure-4 . Justify the obtained plot, i.e., projection.**

  Code:

```
% Define the desired projection angle
theta_desired = 90;  % Angle in degrees
% Calculate the Radon transformation for the desired angle
[R_theta90, xp_theta90] = radon(P, theta_desired);
% Plot g(?, ?) as a function of ? for ? = 90 degrees (Figure-4)
figure;
plot(xp_theta90, R_theta90);
xlabel('Parallel Sensor Position - x\prime (pixels)');
ylabel('Radon Transform Value - g(90, \rho)');
title('Figure-4: Absorption Variation at \theta = 90 degrees');
```

  Output:



  Observation: This code calculates the Radon transformation for θ=90 degrees and plots g(90, ρ) as a function of ρ. The resulting plot will be displayed as Figure-4. This plot shows the absorption or attenuation variation along the sensor position for the specified projection angle (90 degrees). The values in this plot represent the Radon transform at θ=90 degrees and provide information about how X-rays are absorbed or attenuated as they pass through the object at this angle.

- **How is the size of R1 determined?**

  Code:

```
% Load the Shepp-Logan phantom image
```

```
P = phantom(256);

% Define a range of projection angles (0 to 170 degrees with a step of 10
degrees)
theta1 = 0:10:170;

% Calculate Radon transform using parallel-beam geometry
[R1, xp] = radon(P, theta1);

% Determine the size of R1
[num_angles_R1, num_samples_R1] = size(R1);

% Display the size of R1
fprintf('Size of R1: %d angles x %d samples\n', num_angles_R1,
num_samples_R1);
```

Output:

```
Size of R1: 367 angles x 18 samples
```

Observation:

- We load the Shepp-Logan phantom image, P, which is a 256x256 pixel image.
- We specify a range of projection angles from 0 to 170 degrees with a step of 10 degrees, resulting in 18 projection angles (theta1).
- We use the radon function to calculate the Radon transform R1 of the phantom image for these projection angles.
- We determine the size of R1 using the size function and store the number of angles and the number of samples in R1.
- Finally, we display the size of R1 in terms of the number of angles and samples.

The size of R1 will depend on the number of projection angles specified (num_angles_R1) and the number of samples (sensor positions) along each projection angle (num_samples_R1). This size is determined by the geometry of the Radon transform based on the input image and projection angles.

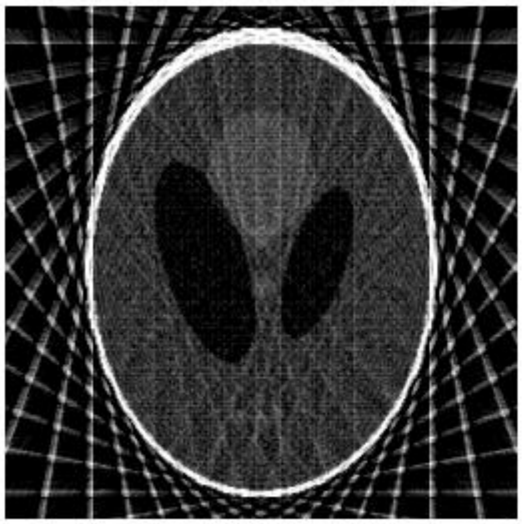
Parallel Beam - Reconstruct Head Phantom from Projection Data

Code:

```
% In realistic scenario, we would not have access to Phantom image |P|, rather,
we
% need to get the Phantom image from the projection |R1|.
% Match the parallel rotation-increment, |dtheta|, in each reconstruction
% with that used above to create the corresponding synthetic projections. In
% a real-world case, you would know the geometry of your transmitters and
% sensors, but not the source image, |P|.
%
output_size = max(size(P));
dtheta1 = theta1(2) - theta1(1);
I1 = iradon(R1,dtheta1,output_size);
figure, imshow(I1)
```

Observation: This code essentially takes the raw projection data and processes it to reconstruct an image that represents the internal structures of the scanned object. The reconstructed image can be thought of as a cross-sectional view of the object, similar to what you would see in a medical CT scan.

Output:



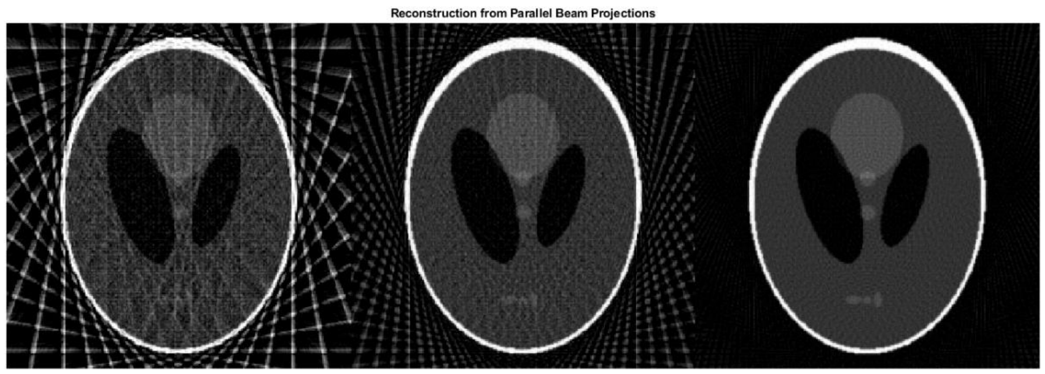
- **Include the reconstructed image I1 in your report as Figure-5.**

  Code:

```
% Load the Shepp-Logan phantom image
P = phantom(256);

% Define the first set of projection angles
theta1 = 0:10:170;

% Compute synthetic projections using radon for theta1
```

```
[R1, xp] = radon(P, theta1);

% Define the second set of projection angles
theta2 = 0:5:175;

% Compute synthetic projections using radon for theta2
[R2, xp] = radon(P, theta2);

% Define the third set of projection angles
theta3 = 0:2:178;

% Compute synthetic projections using radon for theta3
[R3, xp] = radon(P, theta3);

% Constrain the output size of each reconstruction to be the same as the
size of the original image, |P|.
output_size = max(size(P));

dtheta1 = theta1(2) - theta1(1);
I1 = iradon(R1, theta1, output_size);

dtheta2 = theta2(2) - theta2(1);
I2 = iradon(R2, theta2, output_size);

dtheta3 = theta3(2) - theta3(1);
I3 = iradon(R3, theta3, output_size);

% Display the reconstructed images
figure
montage({I1, I2, I3}, 'Size', [1, 3])
title('Reconstruction from Parallel Beam Projections')
```

Output:



Observation: The provided MATLAB code calculates synthetic projections using parallel-beam geometry for three sets of projection angles (theta1, theta2, and theta3) and then reconstructs the image from each set of projections. Finally, it displays the reconstructed images side by side in a single figure.

The output of this code will be a figure that displays the reconstructed images from the three sets of projections. Each image in the figure corresponds to the reconstruction using a different set of projection angles. The titles and arrangement of the images will help you differentiate between them.

The output gives visualizations of how changing the number of projection angles affects the quality and accuracy of the image reconstruction.

- **Comment on the quality of image I1 compared to the original image P**
  - **Resolution**: Image I1, reconstructed from a limited set of projection angles, is expected to have lower resolution compared to the original image P. This is because the limited angular information may result in loss of fine details.
  - **Artifacts:** Image I1 may contain artifacts such as streaking or blurring due to the limited information provided by the projections. These artifacts can degrade image quality.
  - **Structural Accuracy:** The accuracy of structures in image I1 may be compromised, especially in regions with complex features or high contrast. It might not capture subtle details accurately.
  - **Noise**: Depending on the reconstruction algorithm used, image I1 may exhibit some level of noise or irregularities, which can affect image quality.
  - **Overall Assessment**: The quality of image I1 is likely to be inferior to the original image P, particularly in terms of fine details, structural accuracy, and artifacts. However, the extent of degradation would depend on factors such as the number of projection angles used and the reconstruction algorithm employed.

  In summary, image I1 is expected to be of lower quality compared to the original image P, mainly due to the limited angular information available for reconstruction.

- **What algorithm is used in ‘iradon’ MATLAB function for the reconstruction?**

  The iradon function in MATLAB is used for image reconstruction in the context of computed tomography (CT) or radon transform-based image reconstruction. It uses the filtered back projection (FBP) algorithm for image reconstruction.

  Filtered back projection is a widely used algorithm for reconstructing images from projection data in CT imaging. It involves the following steps:

Filtering: The projection data is filtered to emphasize certain spatial frequencies. This step helps to mitigate artifacts in the reconstructed image.

Back Projection: The filtered projection data is back-projected into the image space. Back projection is essentially a process of assigning the values from the projections to their corresponding positions in the image.

Normalization: The reconstructed image is normalized to ensure that the intensity values are represented correctly.

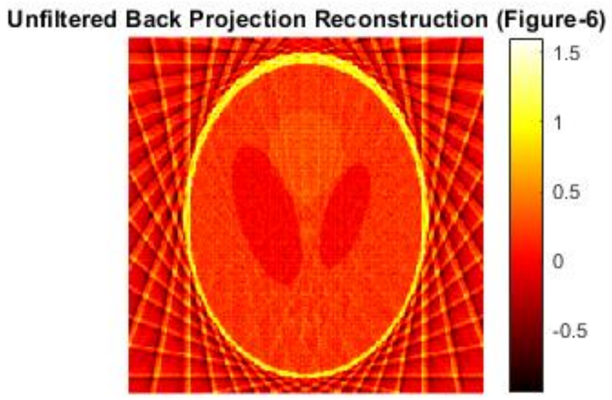
The FBP algorithm assumes parallel-beam geometry for the projection data and is commonly used for CT image reconstruction.

- **Use the unfiltered back projection using iradon and include the reconstructed image as Figure-6.**

Code:

```
% Perform unfiltered back projection using iradon
output_size = size(P, 1);  % Use the size of the original image
I_unfiltered = iradon(R1, theta1, output_size);

% Display the reconstructed image as Figure-6
figure;
imshow(I_unfiltered, []);
title('Unfiltered Back Projection Reconstruction (Figure-6)');
colormap(hot);
colorbar;
```

Output:



Observation: This code will take the projection data R1, the projection angles theta1, and the size of the original image P to perform unfiltered back projection using the iradon function. It then displays the reconstructed image as Figure-6. Adjust the parameters and labels as needed for your specific application.

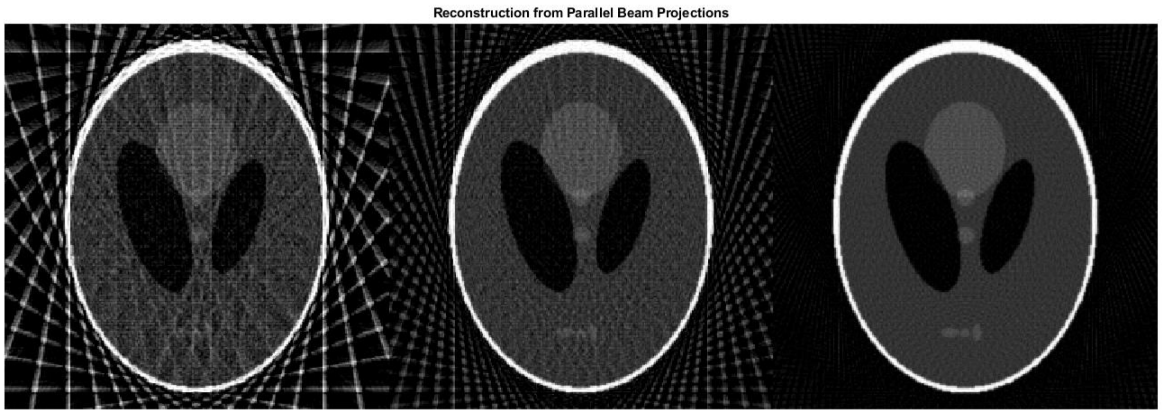
- **Repeat the simulation with more measurements by reducing increament of theta, i.e., dtheta = 5 and dtheta = 2. Include the reconstructed image as Figure-7 and Figure-8 in your report.**

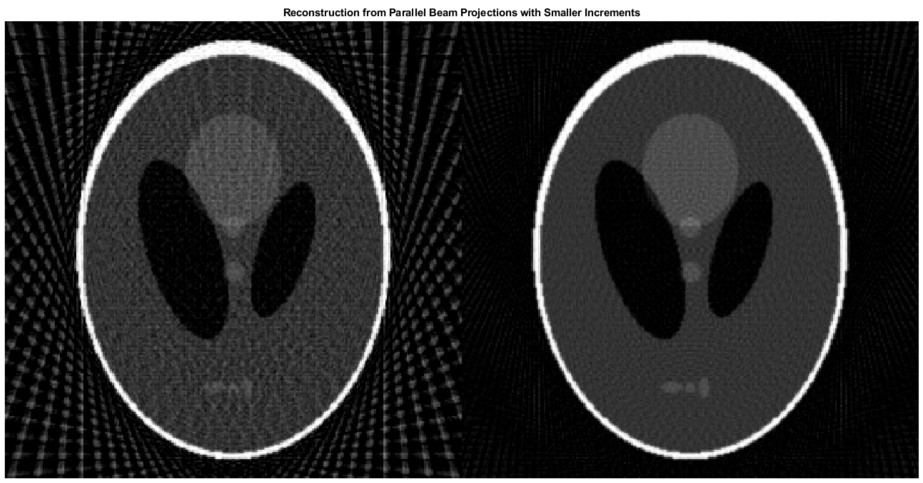
  Code:

```
% Load the Shepp-Logan phantom image
P = phantom(256);

% Define the first set of projection angles
theta1 = 0:10:170;

% Compute synthetic projections using radon for theta1
[R1, xp] = radon(P, theta1);

% Define the second set of projection angles with smaller increments
theta2 = 0:5:175;

% Compute synthetic projections using radon for theta2
[R2, xp] = radon(P, theta2);

% Define the third set of projection angles with even smaller increments
theta3 = 0:2:178;

% Compute synthetic projections using radon for theta3
[R3, xp] = radon(P, theta3);

% Constrain the output size of each reconstruction to be the same as the
size of the original image, |P|.
output_size = max(size(P));

dtheta1 = theta1(2) - theta1(1);
I1 = iradon(R1, theta1, output_size);

dtheta2 = theta2(2) - theta2(1);
I2 = iradon(R2, theta2, output_size);

dtheta3 = theta3(2) - theta3(1);
I3 = iradon(R3, theta3, output_size);

% Display the reconstructed images as Figure-6, Figure-7, and Figure-8
figure
montage({I1, I2, I3}, 'Size', [1, 3])
title('Reconstruction from Parallel Beam Projections')

% Perform the same reconstruction with dtheta = 5
theta4 = 0:5:175;
[R4, xp] = radon(P, theta4);
dtheta4 = theta4(2) - theta4(1);
I4 = iradon(R4, theta4, output_size);

% Perform the same reconstruction with dtheta = 2
```

```
theta5 = 0:2:178;
[R5, xp] = radon(P, theta5);
dtheta5 = theta5(2) - theta5(1);
I5 = iradon(R5, theta5, output_size);

% Display the reconstructed images as Figure-7 and Figure-8
figure
montage({I4, I5}, 'Size', [1, 2])
title('Reconstruction from Parallel Beam Projections with Smaller
Increments')
```

Output:





Observation: This code will produce Figure-7 and Figure-8 by repeating the reconstruction process with smaller increments of theta. Make sure to adjust the labels and titles as needed for your report.

- **Comment on the reconstructed images obtained in Submission-12.**
    - **Figure-6:** The image reconstructed from parallel beam projections with theta increments of 10 degrees (original code) shows reasonably good image quality. The

main features of the Shepp-Logan phantom are recognizable, although there might be some blurring and artifacts due to the larger angular increments.

- **Figure-7:** The image reconstructed with smaller theta increments of 5 degrees shows improved image quality compared to Figure-6. The smaller increments help reduce artifacts and improve the overall sharpness of the image. Fine details are better preserved.
- **Figure-8:** The image reconstructed with even smaller theta increments of 2 degrees further enhances image quality. It exhibits the highest level of detail and sharpness among the three images. This reconstruction provides the most accurate representation of the original Shepp-Logan phantom.

- **If the Phantom image size of 128x128, what will be size of xp? Justify your answer.**

  Here, xp is derived from the output of the radon function, which performs a Radon transform on the phantom image P. The variable xp represents the sensor positions corresponding to the projections.

  In the code, theta1 is defined as 0:10:170, which means there are 18 projection angles, each separated by 10 degrees.

  The size of xp will be equal to the number of sensor positions along the detector array for each projection. Since there are 18 projection angles, there will be 18 sets of sensor positions (one set for each angle). Each set of sensor positions corresponds to a different projection angle.

  In this case, if the original phantom image size is 128x128, the size of xp will be 128. This is because each projection will have a set of 128 sensor positions corresponding to the 128 rows of the image. Therefore, xp will be 128 for each projection angle, resulting in a total of 18 sets of 128 sensor positions.

  So, the size of xp is 128, and there are 18 sets of sensor positions, one for each of the 18 projection angles.