bmi-a01

February 7, 2024

BioMedical Imaging **Assignment01**

BM23MTECH11006

PITHANI TEJA VENKATA RAMANA KUMAR

A sinogram is a 2D plot showing all the projections for a CT scan. On the horizontal axis, each projection is displayed. Each vertical line has the data from a different projections. The vertical lines are organized so that the projection angle increases from 0 to 180 degrees. A point in the image will move as a sine wave in a sinogram – hence the name. Generate the sinogram of a Shepp Logan phantom

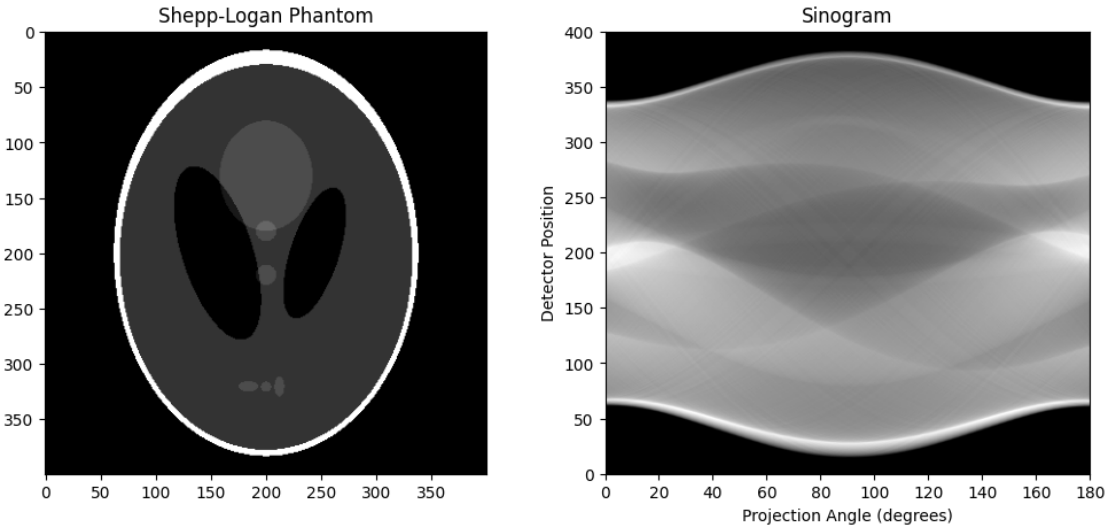
Clues

Decide number of projections and use Radon transfrms (say eg. projections with an angular spacing of 0.5° over 180°.

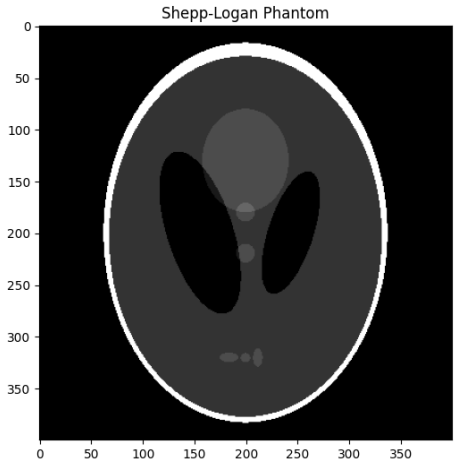
Hint: you can use the MATLAB function radon. Provide an image of the sinogram.

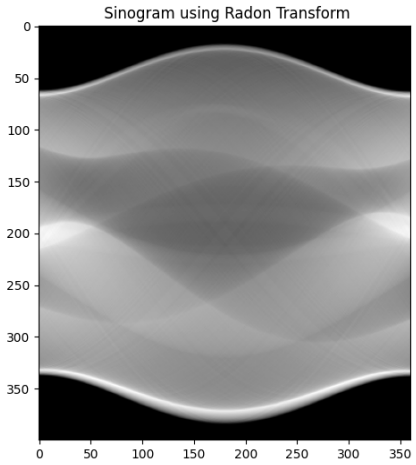
[ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import skimage as sk
from skimage.data import shepp_logan_phantom
from skimage.transform import radon

# Create a Shepp-Logan phantom
phantom_size = 256
shepp_logan_phantom = shepp_logan_phantom()

# Decide the number of projections and angular spacing
num_projections = int(180 / 0.5)  # 0.5-degree angular spacing over 180 degrees
theta = np.linspace(0, 180, num_projections, endpoint=False)

# Use Radon transform to generate sinogram
sinogram = radon(shepp_logan_phantom, theta=theta, circle=True)

# Display the Shepp-Logan phantom and its sinogram
plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
plt.imshow(shepp_logan_phantom, cmap='gray')
```

```
plt.title('Shepp-Logan Phantom')

plt.subplot(1, 2, 2)
plt.imshow(sinogram, extent=(0, 180, 0, sinogram.shape[0]), aspect='auto',␣
 ↪cmap='gray')
plt.title('Sinogram')
plt.xlabel('Projection Angle (degrees)')
plt.ylabel('Detector Position')

plt.show()
```



[ ]:
```
from skimage.transform import  radon
sinogram = radon(shepp_logan_phantom, theta)


# Display the Shepp-Logan phantom, sinogram, and blind backprojection␣
 ↪reconstruction
plt.figure(figsize=(18, 6))

plt.subplot(1, 2, 1)
plt.imshow(shepp_logan_phantom, cmap='gray')
plt.title('Shepp-Logan Phantom')


plt.subplot(1, 2, 2)
plt.imshow(sinogram, cmap='gray')
plt.title(' Sinogram using Radon Transform')

plt.show()
```

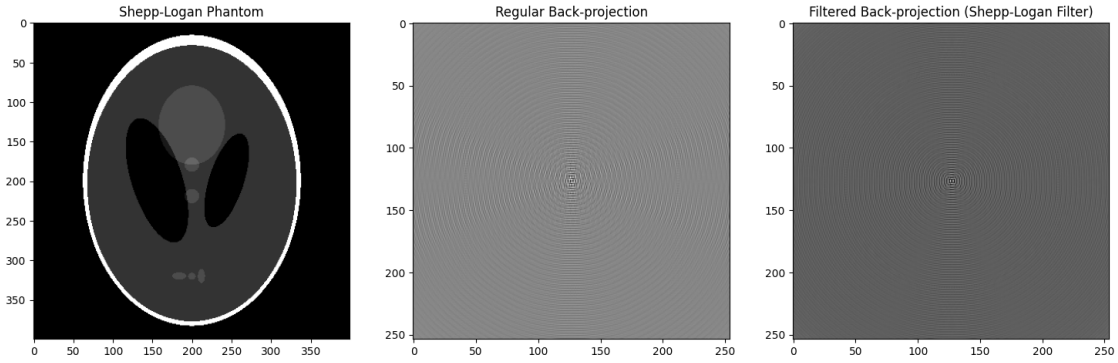




2.What do you mean by filtered back projection? Use a good filter (e.g. Mexican hat wavelet, Shepp-Logan Filter, Rectangular filter etc.) or any other along with backprojection and redo the exercise and show the results of filtered backprojection. Compare the two results and comment how you can make it better.

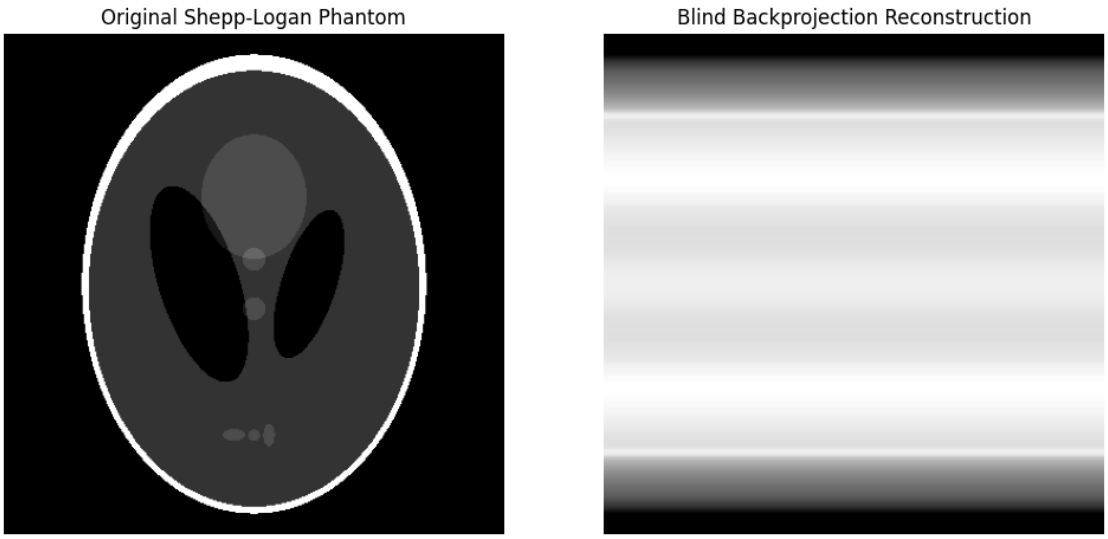
[ ]:
```
import numpy as np
import matplotlib.pyplot as plt
from skimage.transform import iradon
from scipy.signal import convolve2d

# Generate Shepp-Logan phantom image
# Code to generate Shepp-Logan phantom image...

# Generate synthetic projection data and angles
num_projections = 360  # Number of projections
angles = np.linspace(0, 180, num_projections, endpoint=False)
projections = np.zeros((num_projections, shepp_logan_phantom.shape[0]))
for i, angle in enumerate(angles):
    projections[i] = np.sum(np.rot90(shepp_logan_phantom, k=i), axis=1)
theta_360 = np.linspace(0, 180, shepp_logan_phantom.shape[1], endpoint=False)

# Define Shepp-Logan filter
def shepp_logan_filter(shape):
    sinogram_shape = shape[1]
    ramp_filter = np.abs(np.fft.fftshift(np.fft.fftfreq(sinogram_shape)))
    return ramp_filter

# Filtered back-projection function
def filtered_backprojection(projections, angles, filter_func):
    filtered_projections = np.zeros_like(projections)
    for i in range(projections.shape[0]):
```

```
        filtered_projections[i] = np.convolve(projections[i], filter_func,␣
 ↪mode='same')
    reconstruction = iradon(filtered_projections, theta=theta_360, circle=False)
    return reconstruction

# Perform regular back-projection
reconstruction_reg = iradon(projections, theta=theta_360, circle=False)

# Perform filtered back-projection using Shepp-Logan filter
shepp_logan_filt = shepp_logan_filter(projections.shape)
reconstruction_filt = filtered_backprojection(projections, angles,␣
 ↪shepp_logan_filt)

# Display results
plt.figure(figsize=(18, 6))

plt.subplot(1, 3, 1)
plt.imshow(shepp_logan_phantom, cmap='gray')
plt.title('Shepp-Logan Phantom')

plt.subplot(1, 3, 2)
plt.imshow(reconstruction_reg, cmap='gray')
plt.title('Regular Back-projection')

plt.subplot(1, 3, 3)
plt.imshow(reconstruction_filt, cmap='gray')
plt.title('Filtered Back-projection (Shepp-Logan Filter)')

plt.show()
```



Q3) Perform projection tomography on shep logan using blind backprojection and show the reconstruction.

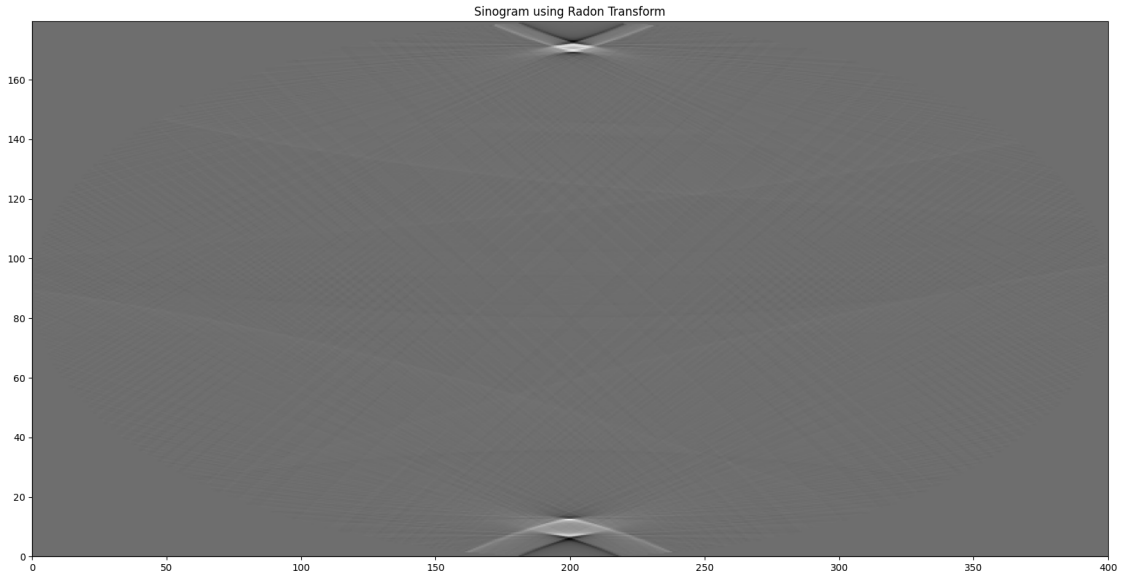
[ ]:
```python
import numpy as np
import matplotlib.pyplot as plt
from skimage.data import shepp_logan_phantom
from scipy.ndimage import rotate

# Generate Shepp-Logan phantom
shepp_logan = shepp_logan_phantom()

# Parameters for back projection
step_size = 2.5
num_projections = int(360 / step_size)
angles = np.linspace(0, 360, num_projections, endpoint=False)  # Angles in␣
↪degrees
detector_size = shepp_logan.shape[1]  # Assuming a square detector

# Simulate projections from the original image
projections = np.zeros((num_projections, detector_size))

for i, angle in enumerate(angles):
    rotated_phantom = rotate(shepp_logan, angle, reshape=False)
    projections[i, :] = np.sum(rotated_phantom, axis=0)

# Blind backprojection reconstruction
blind_backprojection_reconstruction = np.zeros_like(shepp_logan)

for i, angle in enumerate(angles):
    rotated_projection = np.tile(projections[i, :], (detector_size, 1)).T
    blind_backprojection_reconstruction += rotated_projection

blind_backprojection_reconstruction /= num_projections

# Display original and blind backprojection reconstruction
plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)
plt.imshow(shepp_logan, cmap='gray')
plt.title('Original Shepp-Logan Phantom')
plt.axis('off')

plt.subplot(1, 2, 2)
plt.imshow(blind_backprojection_reconstruction, cmap='gray')
plt.title('Blind Backprojection Reconstruction')
plt.axis('off')

plt.show()
```



4.Implement projection and back-projection using Radon transform on Shepp Logan phantom. Generate a sinogram using 360-degree projection. Clues Decide number of projections and use Radon transforms (say eg. projections with an angular spacing of 0.5° over 180°. Hint: you can use the MATLAB function radon. Provide an image of the sinogram.

ChatGPT

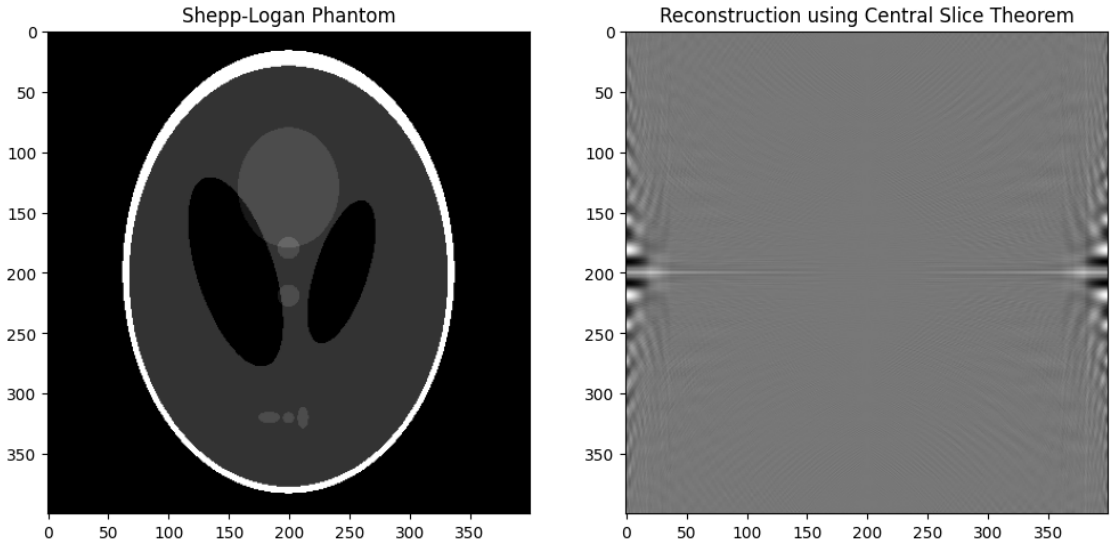
[ ]:
```
import numpy as np
import matplotlib.pyplot as plt
from skimage.transform import iradon
from skimage.data import shepp_logan_phantom

# Generate Shepp-Logan phantom image
shepp_logan_phantom = shepp_logan_phantom()

# Use Inverse Radon transform to perform back-projection
theta_360 = np.linspace(0, 180, shepp_logan_phantom.shape[1], endpoint=False)

sinogram = iradon(shepp_logan_phantom, theta=theta_360)

plt.figure(1, figsize=(20, 10))
plt.imshow(sinogram, cmap='gray', extent=(0, sinogram.shape[0], 0,␣
 ↪theta_360[-1]), aspect='auto')
plt.title('Sinogram using Radon Transform')
plt.show()
```



5.Show a reconstruction of shep logan phantom applying Central slice theorem on shepp logan phantom.

[ ]:
```
from scipy.fft import fft, ifft, fftshift, ifftshift


# Apply Central Slice Theorem for Reconstruction
def central_slice_theorem(sinogram, phantom_size):
    fft_sinogram = fftshift(fft(sinogram, axis=0))
    k = np.linspace(-np.pi, np.pi, phantom_size, endpoint=False)
    kx, ky = np.meshgrid(k, k)
    filtered_fft_sinogram = fftshift(fft(sinogram, axis=0))
    filtered_sinogram = ifft(filtered_fft_sinogram, axis=0)
    reconstruction = np.real(ifft(filtered_sinogram, axis=0))
    return reconstruction

# Reconstruct using Central Slice Theorem
reconstruction_cst = central_slice_theorem(sinogram, phantom_size)

# Display the Shepp-Logan phantom and the reconstructed image using Central␣
 ↪Slice Theorem
plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)
plt.imshow(shepp_logan_phantom, cmap='gray')
plt.title('Shepp-Logan Phantom')

plt.subplot(1, 2, 2)
```

```
plt.imshow(reconstruction_cst, cmap='gray')
plt.title('Reconstruction using Central Slice Theorem')

plt.show()
```

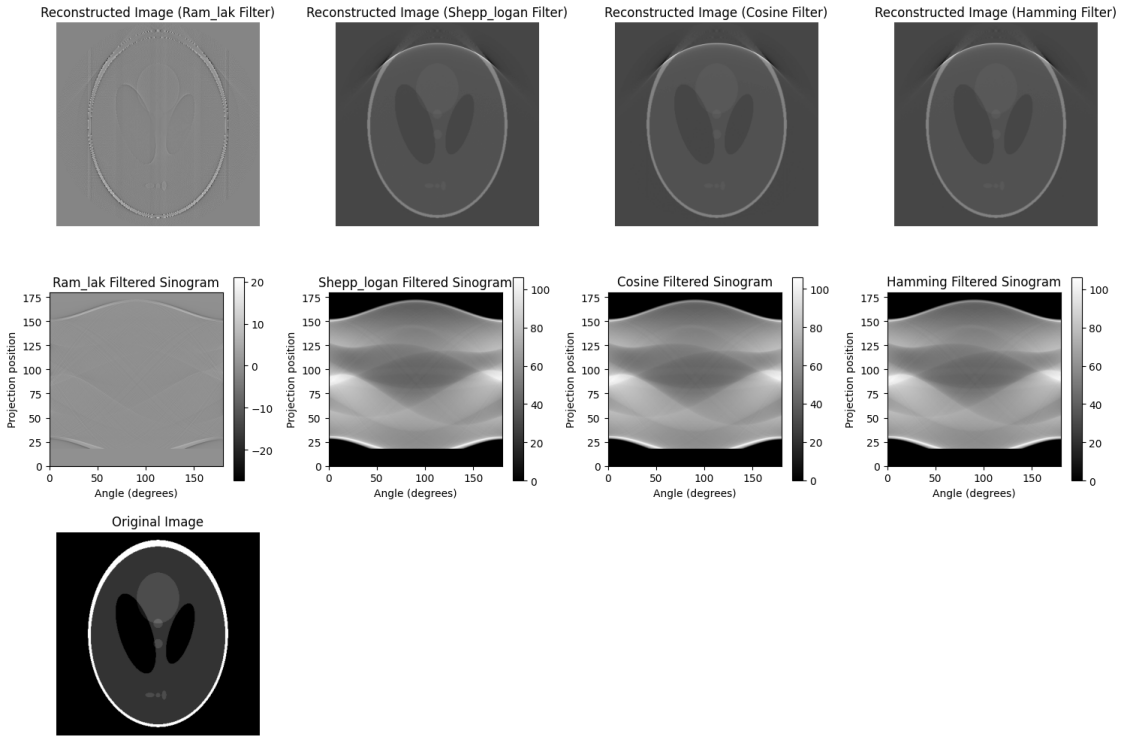


6. Implement filtered backprojection with different filters and find mean squared error between original image and reconstructed image for all filters. 1. Ram-Lak 2. Shepp-Logan 3. cosine, 4. Hamming.

[ ]:
```
def apply_filter(projections, filter_type):
    """
    Apply the specified filter to the sinogram.
    """
    if filter_type == 'ram_lak':
        # Ram-Lak filter
        filter_func = lambda w: np.abs(w)
    elif filter_type == 'shepp_logan':
        # Shepp-Logan filter
        filter_func = lambda w: np.sinc(w / (2 * np.pi))
    elif filter_type == 'cosine':
        # Cosine filter
        filter_func = lambda w: np.cos(w / 2)
    elif filter_type == 'hamming':
        # Hamming filter
        filter_func = lambda w: 0.54 + 0.46 * np.cos(w / 2)
    else:
        raise ValueError("Invalid filter type")

    angles = np.arange(0, 180, angular_spacing)
```

```
    filtered_projections = np.zeros_like(projections)

    for i, angle in enumerate(angles):
        # Apply filter in frequency domain
        frequency_values = np.fft.fftfreq(len(projections[i]))
        filter_values = filter_func(2 * np.pi * frequency_values)
        filtered_projections[i] = np.fft.ifft(np.fft.fft(projections[i]) *␣
 ↪filter_values).real

    return filtered_projections

def backprojection(sinogram, theta):
    """
    Backprojection of the filtered sinogram.
    """
    reconstructed_image = iradon(sinogram, theta=theta, circle=True)
    return reconstructed_image

def calculate_mse(original_image, reconstructed_image):
    """
    Calculate the mean squared error between original and reconstructed images.
    """
    mse = np.mean((original_image - reconstructed_image)**2)
    return mse

# Set the number of projections and angular spacing
angular_spacing = 0.5
theta = np.arange(0, 180, angular_spacing)


# Generate sinogram using Radon transform
sinogram = radon(shepp_logan_phantom, theta=theta, circle=True)

# Apply different filters and perform filtered backprojection
filter_types = ['ram_lak', 'shepp_logan', 'cosine', 'hamming']
mse_values = []

plt.figure(figsize=(15, 10))

for i, filter_type in enumerate(filter_types, start=1):
    # Apply filter to sinogram
    filtered_sinogram = apply_filter(sinogram, filter_type)

    # Perform backprojection
    reconstructed_image = backprojection(filtered_sinogram, theta)

    # Calculate MSE
```

```
    mse = calculate_mse(shepp_logan_phantom, reconstructed_image)
    mse_values.append(mse)

    # Display results
    plt.subplot(3, 4, i)
    plt.title(f"Reconstructed Image ({filter_type.capitalize()} Filter)")
    plt.imshow(reconstructed_image, cmap='gray')
    plt.axis('off')

    plt.subplot(3,4, i + 4)
    plt.title(f"{filter_type.capitalize()} Filtered Sinogram")
    plt.imshow(filtered_sinogram, cmap='gray', extent=[0, 180, 0, len(theta) *␣
↪angular_spacing])
    plt.xlabel("Angle (degrees)")
    plt.ylabel("Projection position")
    plt.colorbar()


# Display original image
plt.subplot(3, 4, 9)
plt.title("Original Image")
plt.imshow(shepp_logan_phantom, cmap='gray')
plt.axis('off')

plt.tight_layout()
plt.show()

# Display MSE values
for i, filter_type in enumerate(filter_types):
    print(f"MSE ({filter_type.capitalize()} Filter): {mse_values[i]}")
```




```
MSE (Ram_lak Filter): 0.05670872704738579
MSE (Shepp_logan Filter): 0.02045409417517703
MSE (Cosine Filter): 0.02057306616792185
MSE (Hamming Filter): 0.020471083360460204
```

0.1 X-RAY IMAGING

8) Using your knowledge of Fourier convolution, calculate the Fourier transforms of the following functions and draw both the real and imaginary spectra. k0 is a real number.

a. cos2(k0 z)

b. sin3(k0 z)

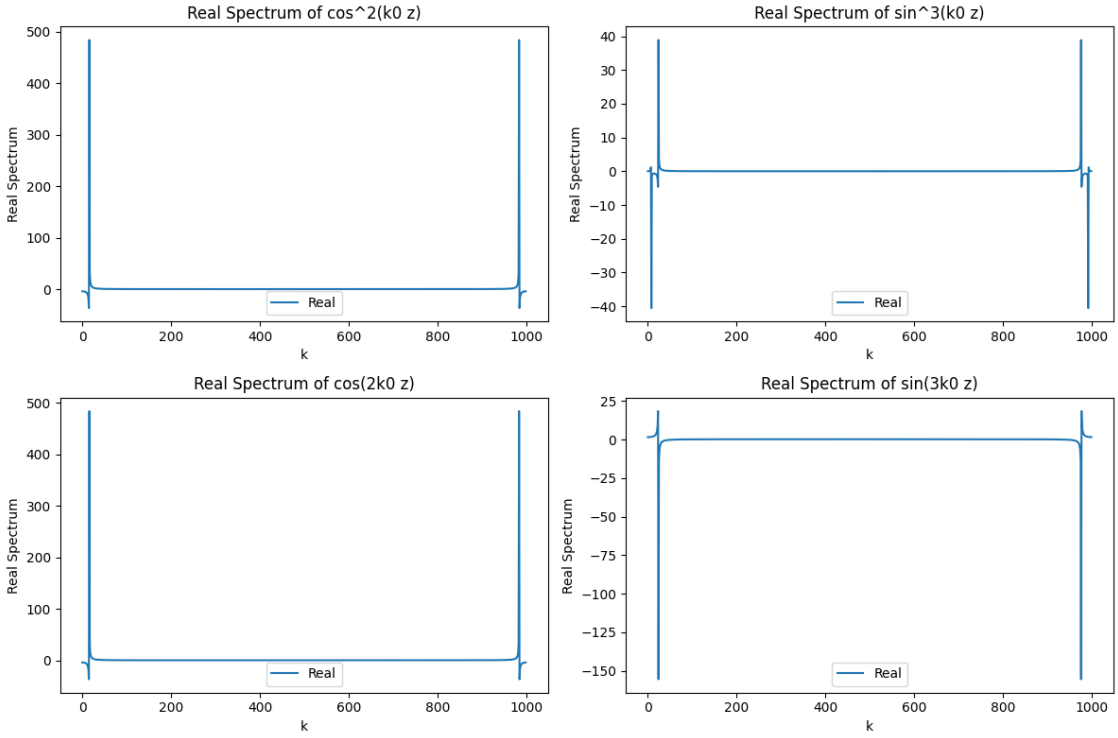
c. cos(2k0 z)

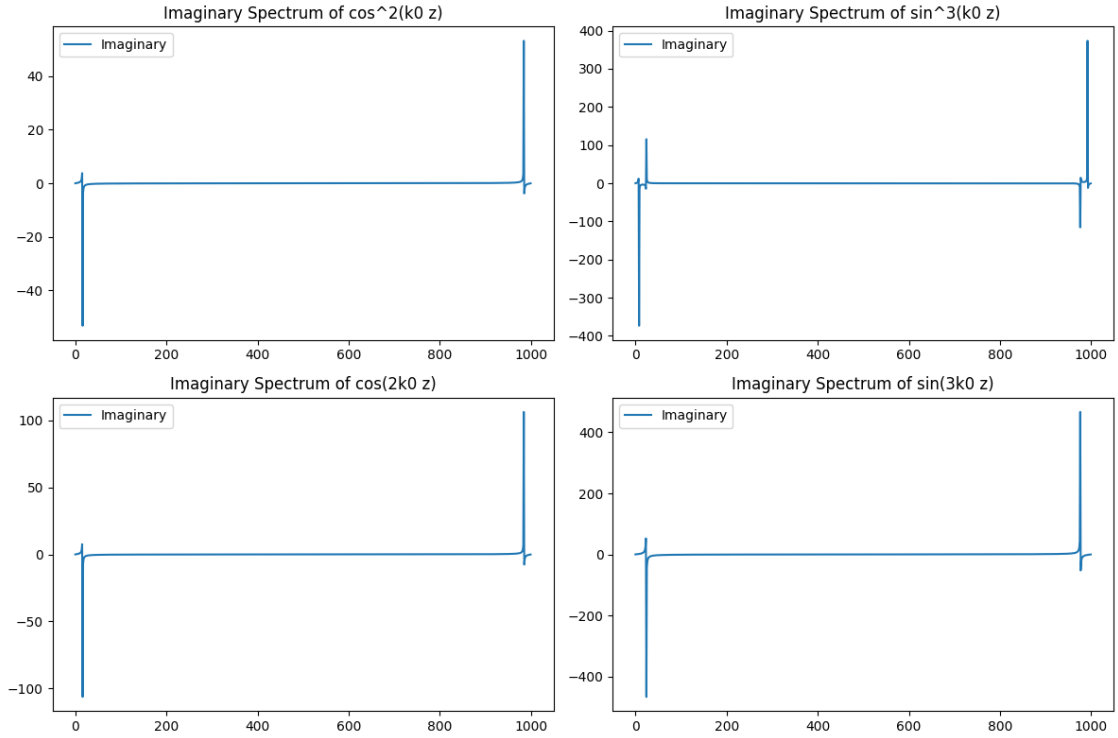
d. sin(3k0 z)

[ ]:
```python
import numpy as np
import matplotlib.pyplot as plt
import scipy as sc

# Define the values
k = 5  # Assuming k0 is 1 for simplicity
z = np.linspace(0, 10, 1000)
```

```
# Functions
cos2 = np.cos(k*z)**2
sin3 = np.sin(k*z)**3
cos_2k0 = np.cos(2*k*z)
sin_3k0 = np.sin(3*k*z)

# Fourier transforms
cos2_fft =  fn_fft = sc.fftpack.fft(cos2)

sin3_fft = sc.fftpack.fft(sin3)
cos_2k0_fft = sc.fftpack.fft(cos_2k0)
sin_3k0_fft = sc.fftpack.fft(sin_3k0)
# Plotting
plt.figure(figsize=(12, 8))

# Real Spectra
plt.subplot(2, 2, 1)
plt.plot(np.real(cos_2k0_fft), label='Real')
plt.xlabel('k')
plt.ylabel('Real Spectrum')
plt.title('Real Spectrum of cos^2(k0 z)')
plt.legend()

plt.subplot(2, 2, 2)
plt.plot(np.real(sin3_fft), label='Real')
plt.xlabel('k')
plt.ylabel('Real Spectrum')
plt.title('Real Spectrum of sin^3(k0 z)')
plt.legend()

plt.subplot(2, 2, 3)
plt.plot(np.real(cos_2k0_fft), label='Real')
plt.xlabel('k')
plt.ylabel('Real Spectrum')
plt.title('Real Spectrum of cos(2k0 z)')
plt.legend()

plt.subplot(2, 2, 4)
plt.plot(np.real(sin_3k0_fft), label='Real')
plt.xlabel('k')
plt.ylabel('Real Spectrum')
plt.title('Real Spectrum of sin(3k0 z)')
plt.legend()

plt.tight_layout()
plt.show()
```

```
# Imaginary Spectra
plt.figure(figsize=(12, 8))

plt.subplot(2, 2, 1)
plt.plot(np.imag(cos2_fft), label='Imaginary')
plt.title('Imaginary Spectrum of cos^2(k0 z)')
plt.legend()

plt.subplot(2, 2, 2)
plt.plot(np.imag(sin3_fft), label='Imaginary')
plt.title('Imaginary Spectrum of sin^3(k0 z)')
plt.legend()

plt.subplot(2, 2, 3)
plt.plot(np.imag(cos_2k0_fft), label='Imaginary')

plt.title('Imaginary Spectrum of cos(2k0 z)')
plt.legend()

plt.subplot(2, 2, 4)
plt.plot(np.imag(sin_3k0_fft), label='Imaginary')

plt.title('Imaginary Spectrum of sin(3k0 z)')
plt.legend()

plt.tight_layout()
plt.show()
```


Real Spectrum of cos^2(k0 z)
Real Spectrum of sin^3(k0 z)
Real Spectrum of cos(2k0 z)
Real Spectrum of sin(3k0 z)
Real Spectrum
k
Real


Imaginary Spectrum of cos^2(k0 z)
Imaginary Spectrum of sin^3(k0 z)
Imaginary Spectrum of cos(2k0 z)
Imaginary Spectrum of sin(3k0 z)
Imaginary

[ ]:

7) Projection radiography, commonly used in computed tomography (CT) imaging, has following limitation:

1. Superimposition of Structures: In traditional projection radiography, structures in the body are superimposed onto a single 2D image, making it challenging to distinguish overlapping tissues. This limitation can reduce the clarity and accuracy of the image, particularly in complex anatomical regions.
2. Limited Soft Tissue Contrast: Projection radiography provides limited contrast between soft tissues, as it primarily relies on differential X-ray attenuation by tissues with varying densities. Consequently, subtle differences in soft tissue contrast may not be adequately visualized.
3. Image Artifacts: Projection radiography images are susceptible to various artifacts, such as beam hardening artifacts, scatter radiation and motion artifacts. These artifacts can degrade image quality, distort anatomical structures and obscure diagnostic information leading to potential misinterpretations.

4. Radiation Exposure: CT imaging involves exposure to ionizing radiation which carries potential health risks, particularly with repeated & high dose exposures. While modern CT scanners employ dose reduction techniques to minimize radiation exposure, radiation dose remains a concern, especially in sensitive populations such as children & pregnant individuals.

5. Limited Functional Information: Projection radiography primarily provides anatomical information and lacks functional data about physiological processes occuring within the body.

6. Cost and Accessibility: CT imaging equipment and procedures can be costly, limiting accessibility for some patients particularly in resource-limited settings. Additionally, the availability of CT scanners may be limited in certain regions, leading to potential delays in diagnosis & treatment.

1) The Signal-to-Noise Ratio (SNR) is a measure of the quality of an image in ~~an~~ X-ray imaging. It is the ratio of the signal, representing useful information, to the noise which is unwanted background or random variations.

Following are the factors affecting SNR:

1. Exposure Time: Longer exposure times can increase the signal by capturing more photons, improving SNR.
2. Radiation Dose: Higher radiation doses generally lead to increased signal, but there is a trade-off with patient safety and regulatory limits.
3. Detector Efficiency: Efficient detectors enhance the detection of x-ray signals, improving SNR.
4. Image processing: Advanced image processing techniques can be employed to enhance the signal and reduce noise.

We can determine the ~~dose~~ increase if the Signal-to-Noise Ratio (SNR) doubles, using the relationship between SNR and

dose, which is typically quadratic.

If SNR doubles, it means the ratio of signal to noise has increased by a factor of 2. Given the quadratic relationship, the dose increase needed to achieve this doubling of SNR would be proportional to the square of the change in SNR.

Let's denote the initial SNR as SNR_1, and the final SNR as SNR_2. The dose increase can be calculated using the formula

$$DI = \left(\frac{SNR_2}{SNR_1}\right)^2$$

If $SNR_2 = 2 \times SNR_1 \Rightarrow \frac{SNR_2}{SNR_1} = 2$

$$\therefore DI = (2)^2 = 4$$

So the dose increase would be 4 times the original dose to double the SNR. This illustrates the quadratic relationship between SNR and dose in x-ray imaging.

2) Beam hardening is a phenomenon that occurs in X-ray imaging when the X-ray beam passes through an object or tissue of varying thicknesses and compositions. It leads to an alteration in the X-ray spectrum, resulting in the hardening of the beam. Here's how it happens:

1. X-ray Absorption: As the X-ray beam travels through the object or tissue, lower energy X-rays are preferentially absorbed, while higher energy X-rays penetrate further.

2. Increase in Mean Energy: With lower energy X-rays being absorbed more readily, the mean energy of the X-ray beam increases as it passes through the object. This shift towards higher energies is known as beam hardening.

3. Effect on Image Quality: Beam hardening can lead to artifacts in the X-ray image, such as streaks or cupping artifacts, where areas of the image appear

brighter or darker than they should due to the uneven attenuation of X-rays.

Beam hardening artifacts can be mitigated using techniques such as beam filtration, where filters are used to selectively remove lower energy X-rays, or by employing advanced image correction algorithms during image reconstruction. These methods help to reduce the impact of beam hardening on the quality of X-ray images.

3) Aliasing: Aliasing is a phenomenon that occurs when a signal is sampled at too low a rate, causing high frequency components of the signal to be incorrectly represented as lower frequencies. When the sampling rate is insufficient to accurately capture the frequency content of the signal, aliasing artifacts appear in the sampled signal. These artifacts can distort or obscure the

original signal, leading to inaccuracies or misinterpretations in data analysis or image reconstruction.

Bandwidth Limiting:

Bandwidth limiting refers to the process of restricting the frequency range of a signal or system to a specific range of frequencies. This can be achieved through various methods such as analog or digital filters. By limiting bandwidth of a signal, you effectively control the range of frequencies that are transmitted or processed, which can be important for avoiding aliasing or for focusing on specific frequency components of interest while attenuating others.

Nyquist Condition:

The Nyquist condition, also known as the Nyquist criterion or Nyquist Sampling theorem, is a fundamental principle in signal processing that governs the minimum sampling rate required to accurately represent a continuous signal without aliasing. According to Nyquist theorem, to avoid aliasing, the

Sampling rate must be at least twice the highest frequency component present in the signal. In other words, the Nyquist frequency must be greater than the highest frequency in the signal. By adhering to the Nyquist condition during sampling, you ensure that the original signal can be accurately reconstructed from its discrete samples without introducing aliasing artifacts.

6) If a friend sought my opinion on whether to undergo X-ray imaging for heart related diseases, I would approach the conversation by considering pros and cons of X-ray imaging in this context, while also highlighting alternative diagnostic options. Here's how I would advise them:

Advice:

1. Consultation with a Healthcare Professional:
   I would strongly recommend consulting with a cardiologist or a healthcare professional specialization in cardiac imaging. They can access the friend's medical history,

symptoms and specific diagnostic needs, providing personalized guidana based on their expertise.

2. Exploration of Alternative Imaging Modalities:
It's important to explore alternative imaging modalities that may offer advantages in terms of diagnostic accuracy and safety. Options such as echocardiography, MRI or CT scans might be more suitable for evaluating heart-diseases, providing detailed information about cardiac anatomy and function without the potential risks associated with ionizing radiation.

3. Pros and Cons of X-ray Imaging:

Pros:

- Accessibility: X-ray imaging is widely available in healthcare facilities, offering quick and convenient access to diagnostic information.
- Cost-effectiveness: Compared to some advanced imaging modalities, X-ray imaging may be more cost-effective.
- Initial assessment: X-rays can provide an initial assessment of cardiac size, shape and the presence of abnormalities such as fluid accummulation or calcifications.

Cons:

- Radiation Exposure: X-ray imaging involves exposure to ionizing radiation, which carries potential health risks, particularly with repeated or high dose exposures.
- Limited soft tissue detail: X-rays primarily visualize bone and dense structures, providing limited detail of soft tissues such as heart muscles or blood vessels.
- Diagnostic Limitations: X-ray imaging may not always provide sufficient information for comprehensive evaluation of certain heart conditions, potentially necessiating additional imaging tests for accurate diagnostics.

Ultimately, the decision to undergo X-ray imaging for heart related diseases should be guided by a thorough discussion with a healthcare professional, considering the friend's medical needs, concerns and the available diagnostic resources.

4) HVL (Half-Value Layer) means a particular layer thickness that reduces X-ray beam intensity by half.

From Beer-Lamert's law $I = I_0 e^{-ut}$

$I_0/2 = I_0 e^{-ut} \Rightarrow 1 = 2e^{-ut}$

Applying log on both sides

$0 = \ln(2) - ut \Rightarrow t = \ln(2)/u$

For muscle, $u = \ln(2)/3.5 \sim 0.2\ cm^{-1}$

Now, 16 cm tissue $N = N_0 e^{-(0.2)16} \sim N \times 0.04$

For bone, $u = \ln(2)/1.8 \sim 0.4\ cm^{-1}$

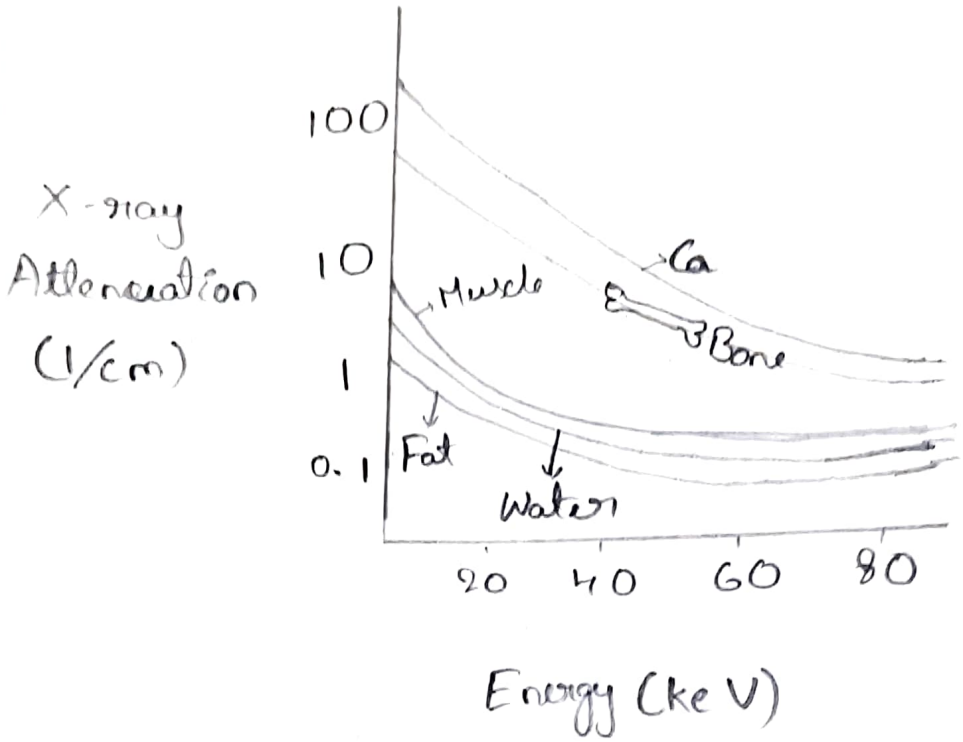
Now, 4 cm bone $N = N_0 e^{-(0.4)4} \sim 0.2$

$\therefore$ X-rays transmission percentage

$= 100 \times 0.2 \times 0.04$

$= 0.8\ \%$

5)



In the Attenuation vs Energy graph, it is observed that around 50 keV, most body tissues, including bones and muscles, absorb low-energy photons due to their absorption coefficient at lower keV levels. However at 140 keV, there is a distinct difference in X-ray attenuation (absorption coefficient) between bones and soft tissues (muscles), leading to clear differentiation on radiographic film.

7) A Multitap AC transformer is available for accessing multiple output lines (taps), enabling the selection of varying voltage levels as needed for X-ray imaging. This feature allows users to choose higher or lower voltage taps based on specific imaging requirements. In X-ray imaging systems, timing circuits incorporate counters, such as flip-flops, which are utilized to measure the exposure duration accurately.

7) A Multitap AC transformer is available for accessing multiple output lines (taps), enabling the selection of varying voltage levels as needed for X-ray imaging. This feature allows users to choose higher or lower voltage taps based on specific imaging requirements. In X-ray imaging systems, timing circuits incorporate counters, such as flip-flops, which are utilized to measure the exposure duration accurately.