Part A

Question 1

Calculate the Radon transform of an image and test the back-projection method.

(a) Load an image of the Shepp-Logan phantom of size 128 × 128. We will refer to this as f_{true}. Show a picture of f_{true}.

ANSWER:

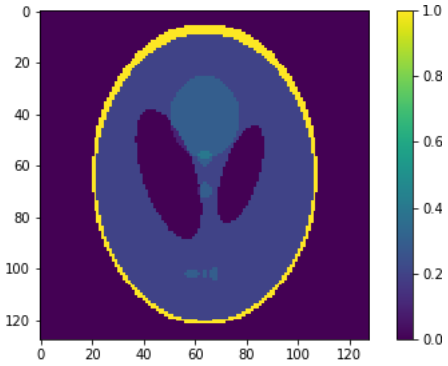

Figure 1: Schepp-Logan Phantom

(b) Generate the Radon transform $g = \mathcal{R}f$ of this phantom in 1-degree intervals from 0 179. Display g as a 2D-image; this is referred to as the sinogram of f_{true} . What is the size of this sinogram and how is this determined?

ANSWER:

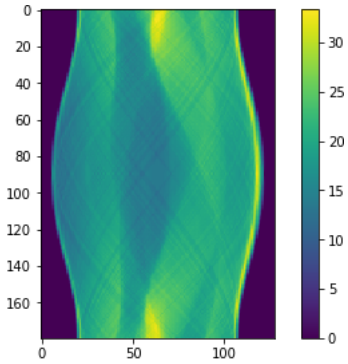

Figure 2: Sinogram

Above I display the Radon transform of the Schepp-Logan phantom depicted in Figure 1. Associated with a discretized form of the Radon transform is a 'projection

geometry'. A projection geometry is defined by the number of measurements taken by a detector. Measurements can be taken at multiple angles and using a number of detectors. Hence, we are interested in two quantities - the number of angles sampled n_{ang} and the number of detector pixels n_{det}. The resultant sinogram of the transform is then of size $n_{\text{ang}} \times n_{\text{det}}$. Above we see that our sinogram is of size 180 x 128 corresponding to 1 degree intervals of a 180 angle range and 128 detector counts.

(c) Compute the unfiltered back-projection and apply it to the sinogram data you generated. What is the size of the back-projected image?

Below I display the phantom reconstructed using back projection, as well as the difference between the reconstruction and the original. We see that there is significant changes in scale, as well as distortion in nearly all parts of the image. The size of the back projected reconstruction is the same as the image corresponding to the sinogram data.

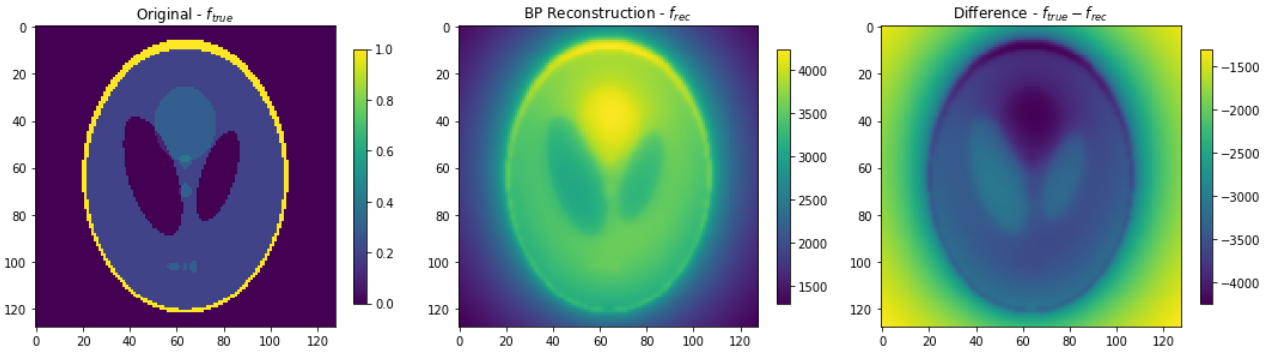

Figure 3: Back Projection

(d) Compute the filtered back-projection and apply it to the sinogram data g that you generated. Verify that this gives a good estimate of the inverse of the Radon transform.

ANSWER:

Below I display the phantom reconstructed using filtered back projection. We can see from the difference between the reconstruction and original that the reconstruction is a good estimate.

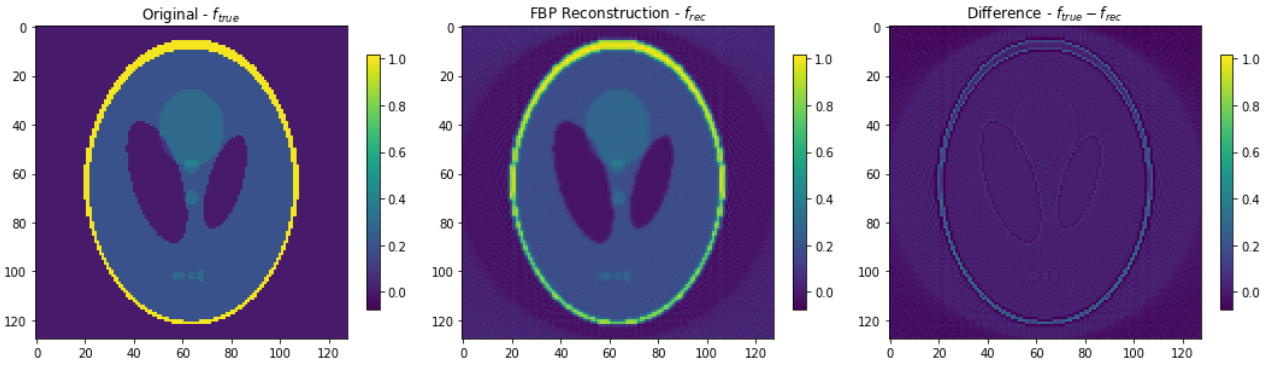

Figure 4: Filtered Back Projection

(e) Add noise to the data g and test how the error in your reconstruction grows with the scale of the measurement noise.

ANSWER:

The ASTRA package allows one to produce a noisy sinogram measurement from a given image, using the function

```
functions.add_noise_to_sino(sinogram,noise_level)
```

The noise level parameter is not especially well documented. Below I calculate the 'signal-to-noise ratio' for increasing values of the noise level parameter. The signal-to-noise ratio (SNR) of a noisy signal g relative to a noiseless signal Af is defined as:

$$\frac{\|Af\|^2}{\mathbb{E}[\|g_{\text{noisy}} - Af\|^2]} \tag{1}$$

Here the expectation is with respect to the noise distribution. This is estimated empirically through repeated sampling, leading to a sample mean. Below I plot the SNR of the noisy sinogram compared to the original, for varying values of the ASTRA noise parameter.

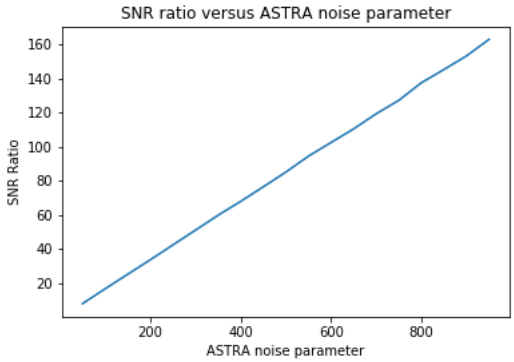

Figure 5: ASTRA Noise Parameter

We see that it is approximately a linear relationship, and higher values of the noise parameter produces less noisy data. With this in mind, we investigate the relationship between filtered back projection reconstruction quality (measured by both L2 error and SNR), and the amount of noise added to our data.

Below is an example reconstruction.

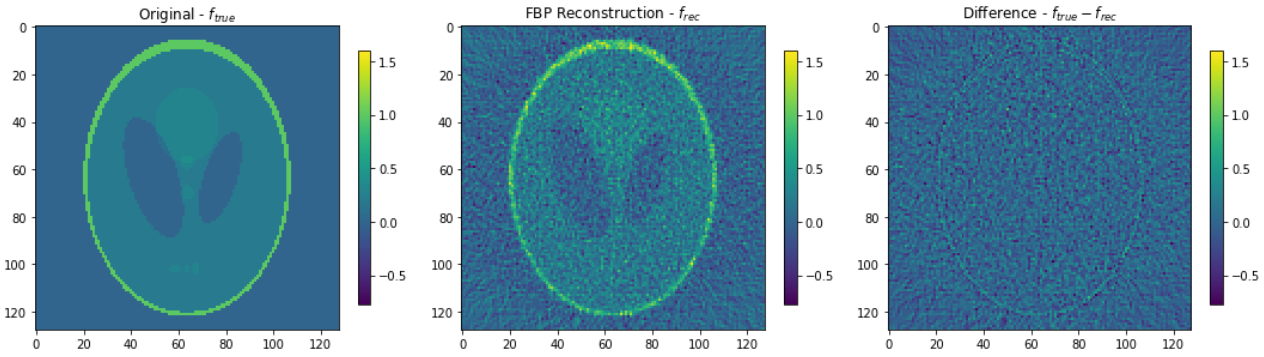

Figure 6: Sinogram

The next chart plots the L2 error $\|f_{\text{true}} - f_{\text{rec}}\|_2$ against the noise level. To make the chart more intuitive, I have reversed the values on the x-axis so that higher values of the noise parameter correspond to higher noise levels in the data.

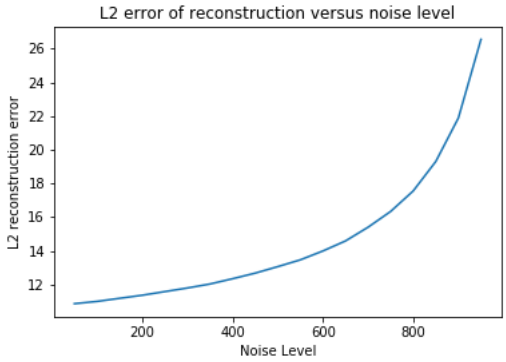

Figure 7: L2 Reconstruction Error

Next we plot the SNR of the reconstruction against the noise level.

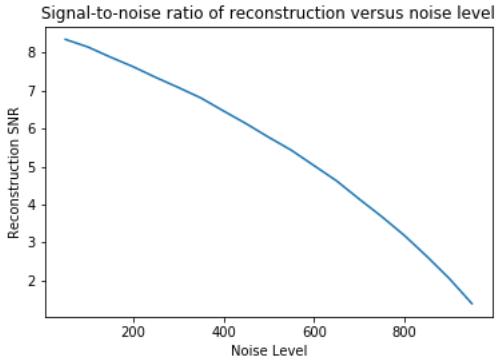

Figure 8: SNR of reconstructed image

Next we plot the SNR of the reconstruction against the SNR of the sinogram data.

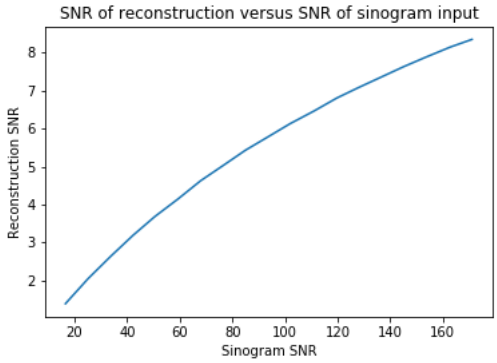

Figure 9: SNR of reconstructed image versus SNR of sinogram

As expected, higher noise levels result in a larger reconstruction error.

Question 2

Calculate an explicit matrix form of the Radon transform and investigate its SVD.

(a) Compute a matrix representation for the Radon transform of an 64 x 64 image and 45 projections. Take the SVD of the Radon matrix and investigate how the spectrum of singular values varies when limiting the number of projections (keeping range of angles constant) and then when limiting the range of angles taken.

Below I plot the singular value distribution of the Radon transform matrix, for varying number of angle projections. I investigated the spectrum for projections $p \in \{5 + 5k : k = 1, \cdots, 16\}$.

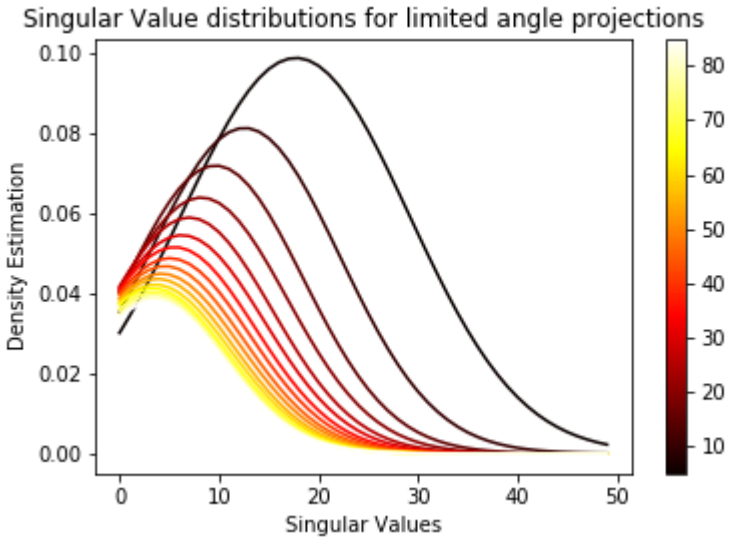

Figure 10: Singular Value distributions for limited angle projections

We see that as the number of projections increase i.e. the angle spacing $d\theta$ reduces, the singular values become increasingly concentrated towards zero.

Below I plot the singular value distribution of the Radon transform matrix, for a varying angle range. I investigated the spectrum for max angle $\theta \in \{\frac{\pi}{k} : k = 1, \cdots, 5\}$.

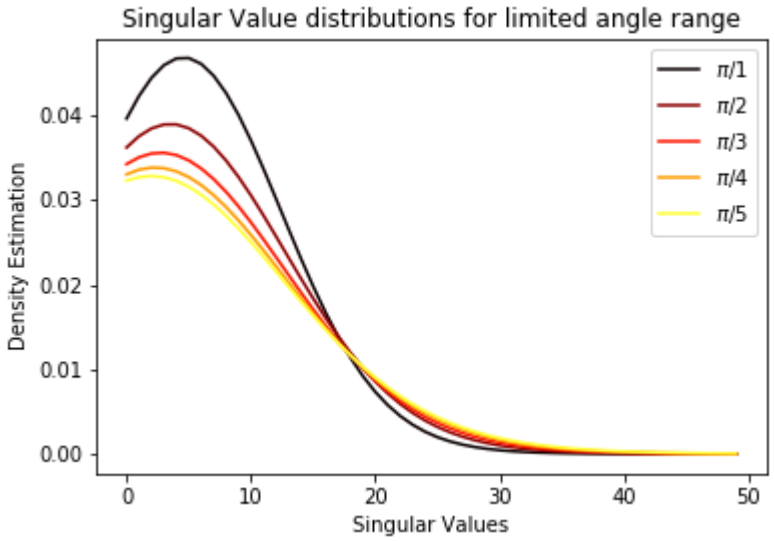

Figure 11: Singular Value distributions for limited angle projections

We see that as the angle range increases (given a constant number of projections, the angle spacing $d\theta$ hence increases), the singular values become less concentrated towards zero.

From the perspective of the angle spacing $d\theta$, these two results are consistent with each other.

Question 3

Implement a matrix-free regularised least-squares solver for the Radon Transform.

One can find a regularised solution to the inverse Radon Transform by solving

$$(A^T A + \alpha L) f_* = A^T g \tag{1}$$

where L is the regularisation matrix. Implement a matrix free solver (considering both zero-order and first-order Tikhonov regularisation) and compare solutions obtained to the filtered back projection method for the following cases:

1. Full range of angles, but limited number of projections
2. Full number of projections, but limited range of angles

   ANSWER:

   First I solve the problem using the normal equation method. I define a scipy LinearOperator class with the appropriate matvec method defined, and pass this linear operator to SciPy's GMRES solver along with the target $A^T g$.

   I first show an example reconstruction, for the projection geometry ($n_{\text{angle}} = 180, \theta_{\max} = \pi$), and using zero-order Tikhonov regularization (corresponding to $L = I$). I use the L-curve method to select a regularization value α. The resulting L-curve can be seen below.

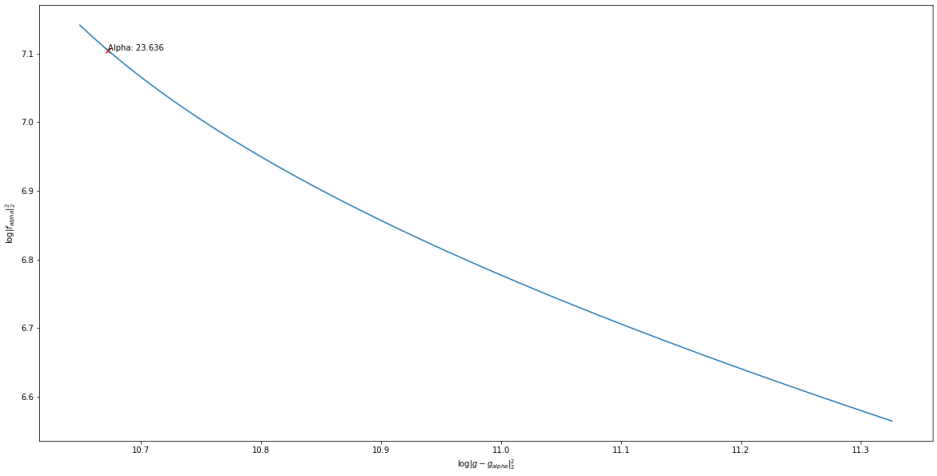

Figure 12: L-curve used to select α

The alpha selected using the L-curve method was $\alpha = 23.63$. The reconstruction obtained by solving the normal equations is shown below. Finally I compare this to the filtered back projection procedure provided by the ASTRA toolbox.

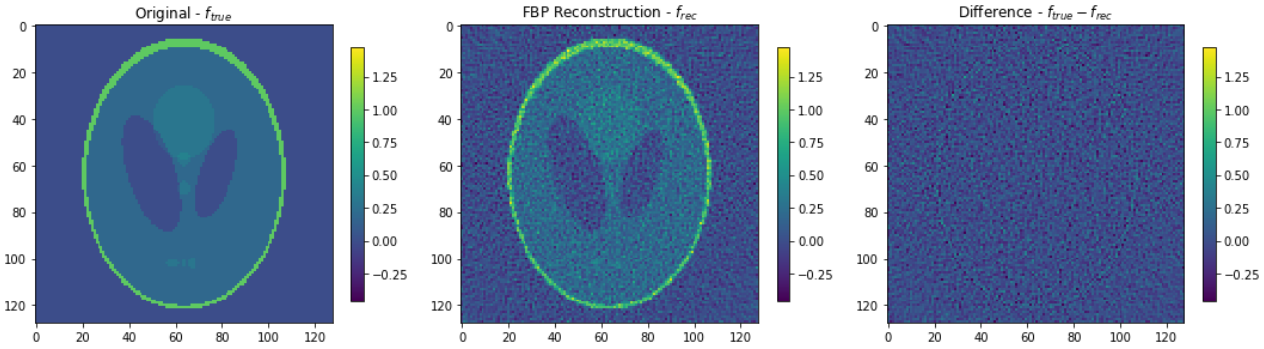

Figure 13: Tik0 - Regularised Least Square Solution

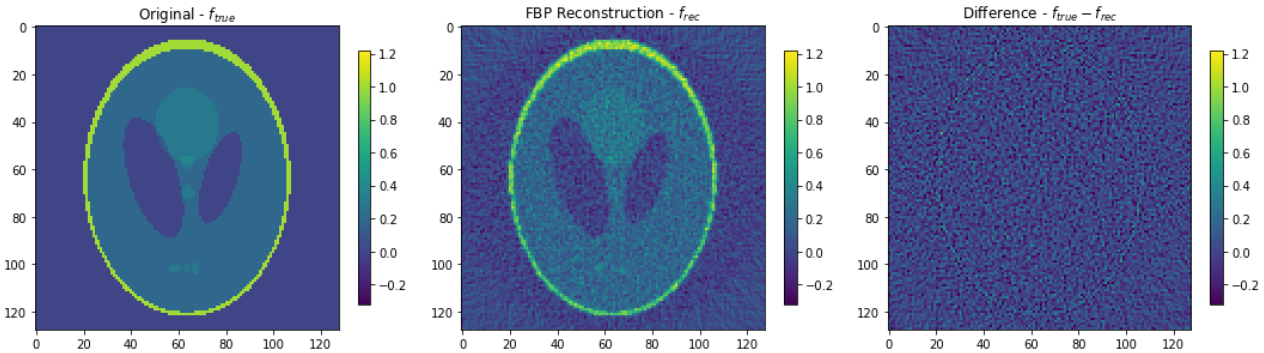

Figure 14: Filtered Back Projection Solution

The L2-error using the ASTRA Filtered Back Projection on this example was $L2_{\text{ASTRA}} = 180.7$, in comparison to $L2_{\text{Tik0}} = 347.6$ achieved by zero-order Tikhonov regularization. Hence the filtered back projection appears to outperform our method in this example.

I now repeat the same experiment but now using first-order Tikhonov regularization (corresponding to $L = D^T D$ where $D = (\nabla_x, \nabla_y)$). The L-curve selected alpha value was $\alpha = 27.96$. The reconstruction results are shown below.

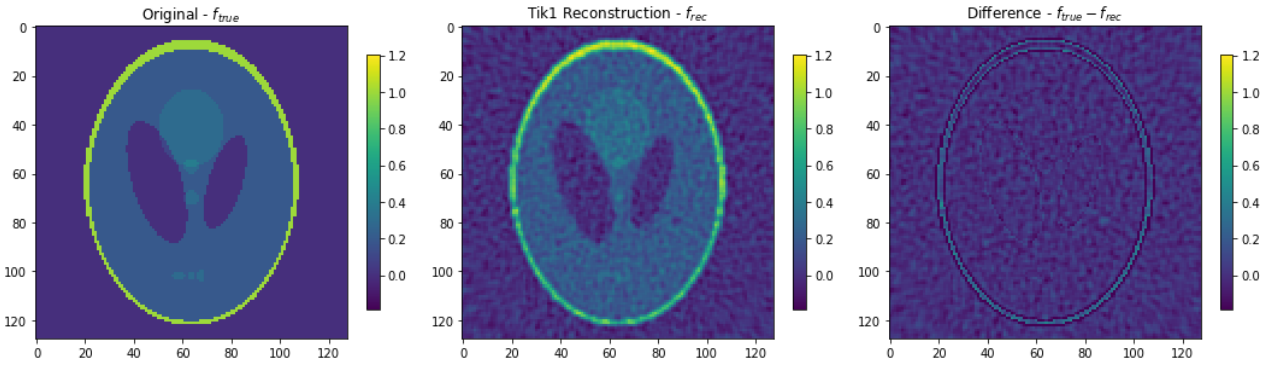

Figure 15: Tik1 - Regularised Least Square Solution

In this instance, the L2 reconstruction error achieved was $L2_{\text{Tik1}} = 123.6$, hence outperforming filtered back projection.

Now we investigate the solutions obtained from the Tik0 and Tik1 method for alternative projection geometries outlined in (1) and (2).

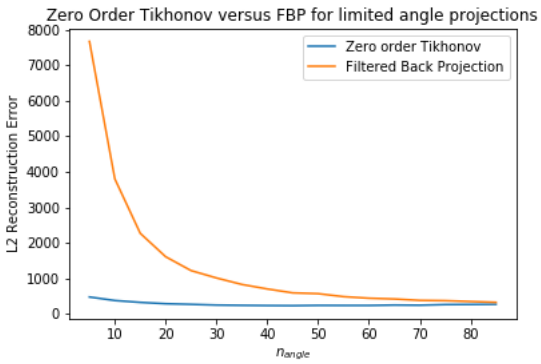

Figure 16: Tik0/FBP - Limited Angle Proj - L2 Reconstruction Comparison

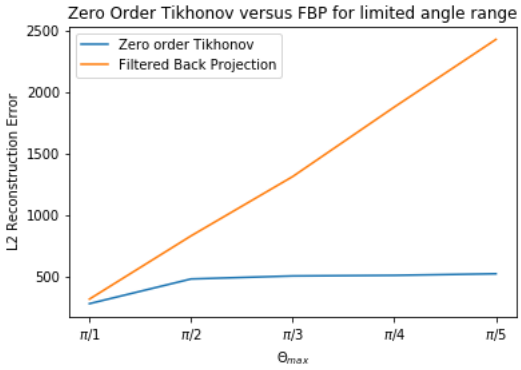

Figure 17: Tik0/FBP - Limited Angle Range- L2 Reconstruction Comparison

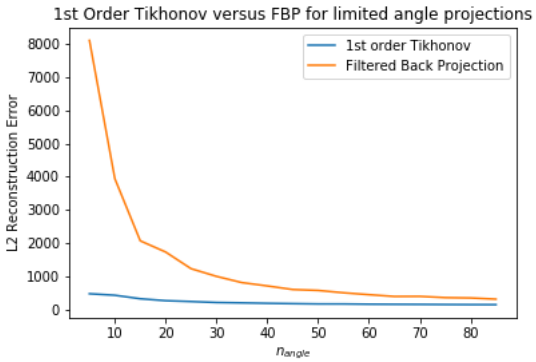

Figure 18: Tik0/FBP - Limited Angle Proj - L2 Reconstruction Comparison

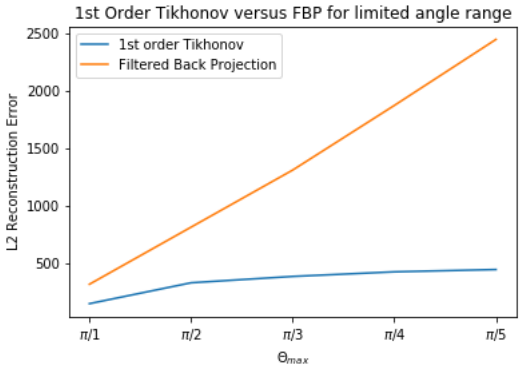

Figure 19: Tik0/FBP - Limited Angle Range - L2 Reconstruction Comparison

We see that in the cases of 'reduced' measurements (i.e limited angle range, limited

projections) Tikhonov regularization (zero and first order) outperforms filtered back projection.

Question 4

Write a Haar wavelet denoiser

In this task we perform denoising using a wavelet transform and shrinkage of the wavelet coefficients.

1. Take any (monochrome) image of your choice. Calculate the Haar wavelet transform of this image. Plot some of the coefficients and explain what you see. Reconstruct the image from the coefficients by calling the inverse wavelet transform. Check if your reconstructed image coincides with the original.

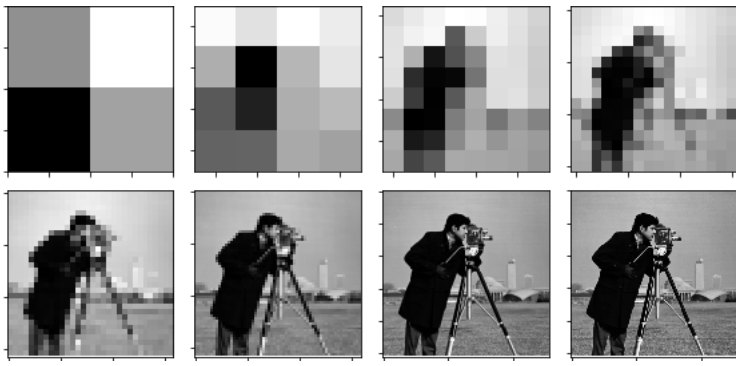

Figure 20: Image reconstruction, resulting from inclusion of increasing wavelet scales

The images above show the effect of reconstructing an image from a restricted set of wavelet coefficients resulting from the wavelet transform applied to the image. Moving left-to-right, we allow coefficients representing finer spatial detail into the reconstruction, hence reaching the original image as all the coefficients are used. The images also demonstrate the 'localisation' of the coarse graining, i.e the top left image is in some way still representative of the original structure, even though many details have been lost.

2. Write a function that implements thresholding for a given range (the different scales of your wavelet coefficients) and threshold parameter, and form a modified image by performing the inverse wavelet transform on the thresholded coefficients.

   ANSWER: See Appendix for code.

3. Create a noisy version of your original image and perform denoising by thresholding of the wavelet coefficients. Investigate the effect of changing the range and the threshold parameter.

   ANSWER:

   Below is the denoised image, resulting from all parameter combinations of threshold value (determined by quantile values) and coefficient ranges. The heat map shows

the signal-to-noise ratio (SNR), defined as $SNR = \frac{\|f_{true}\|^2}{\|f_{recon} - f_{true}\|^2}$, achieved for each reconstruction across this parameter grid. We note that the constant value 'L' shape is the SNR of the noisy image, since these parameter choices amount to no filtering.

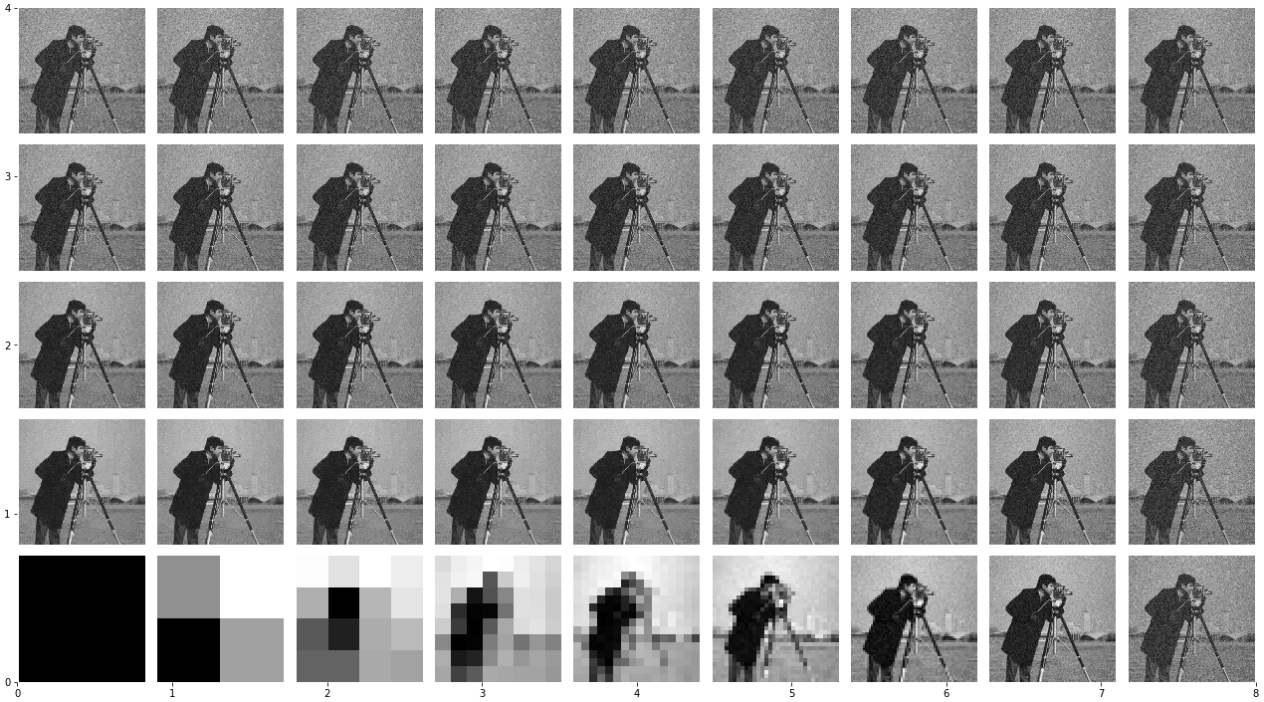

Figure 21: Image reconstruction over parameter grid

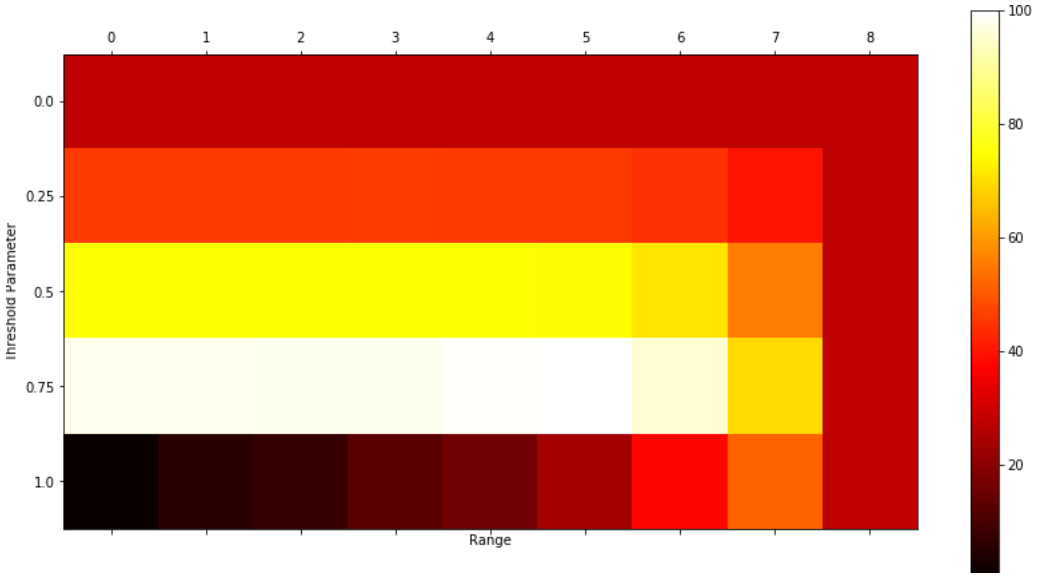

Figure 22: SNR heatmap over parameter grid

We see that the parameter combination leading to the highest SNR is thresholding of the last 4 coefficient scales, by a value equal to the 75-percentile of the absolute value of the set of wavelet coefficients.

I show the reconstruction for this choice below.

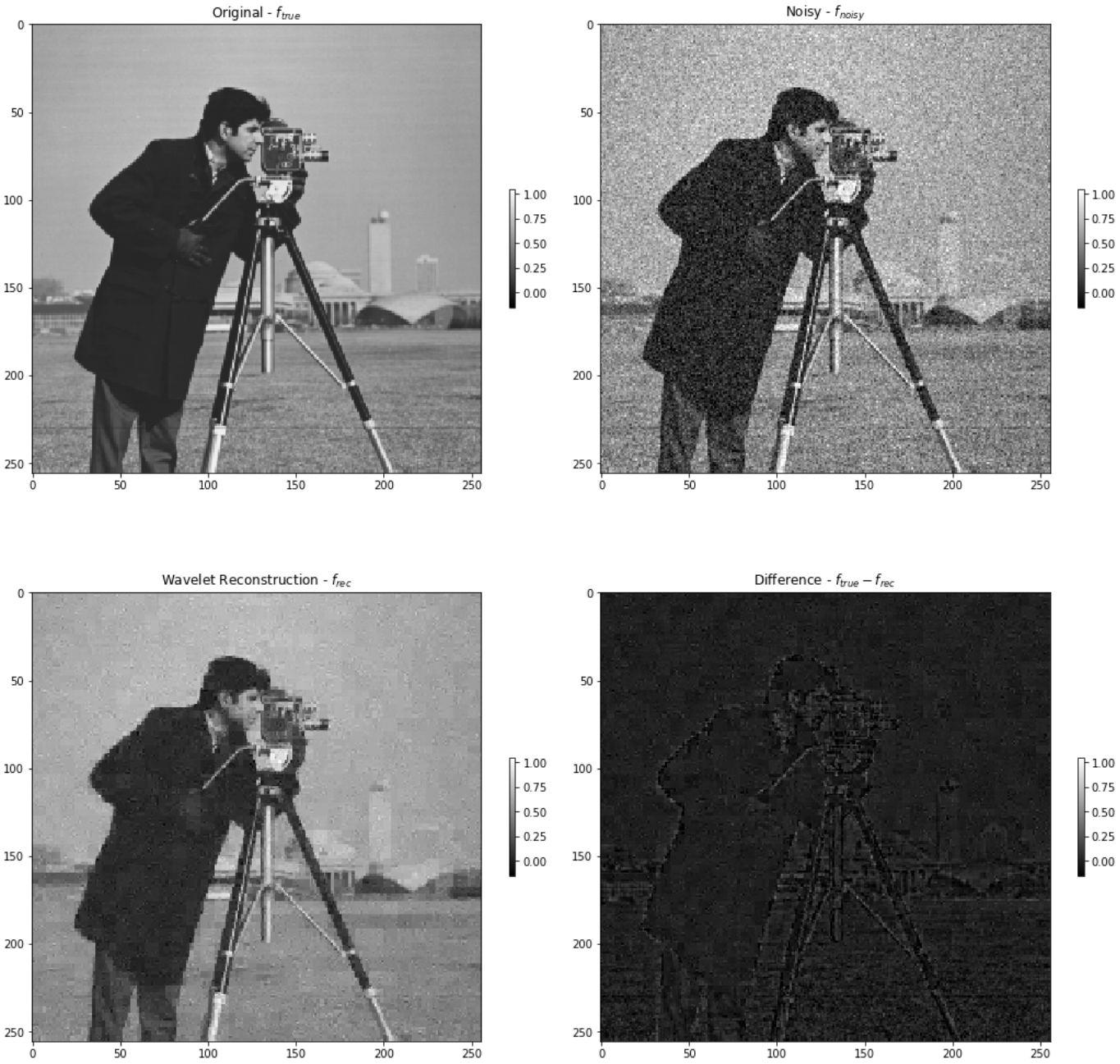

Figure 23: Optimum SNR parameters - reconstruction

Question 5

Iterative soft-thresholding for X-ray tomography

Write your own sparsity promoting reconstruction algorithm for X-ray tomography, by solving

$$\min \frac{1}{2}\|Af - g\|_2^2 + \alpha\|Wf\|_1 \tag{1}$$

where W denotes the wavelet transform. To achieve this use iterative soft-thresholding with Haar wavelets. Iterative soft thresholding completes the following update:

$$f_{k+1} = S_{\alpha,W}(f_k - \lambda A^T(Af - g)) \tag{2}$$

Here $S_{\alpha,W}$ is the soft thresholding operator

$$S_{\alpha,W}(f) = W^{-1}S_\mu Wf \tag{3}$$

Here S_μ is equivalent to the threshold function defined in the Haar wavelet denoiser from Question 4, with threshold parameter $\mu = \alpha\lambda$.

1. Evaluate the algorithm for varying noise levels and projection geometries (limited angles/limited projections).

   ANSWER:

   In order to evaluate the algorithm I tested both limited angle range and limited angle projection geometries. As an initial iterate, I took the naive backprojection $f_0 = A^T g$. As a stopping criterion, I imposed a stopping condition on the size of the L2 norm difference between iterate f_k and iterate f_{k-1}. For the purpose of investigation this was set at 1e-6.

   To test the effect of limiting the angle range, I tested cases corresponding to the maximum angle $\theta_{\max} \in \{\frac{\pi}{k} : k = 1, \cdots, 5\}$. The results are displayed below (for reference, moving down the set of images corresponds to reducing the maximum angle).

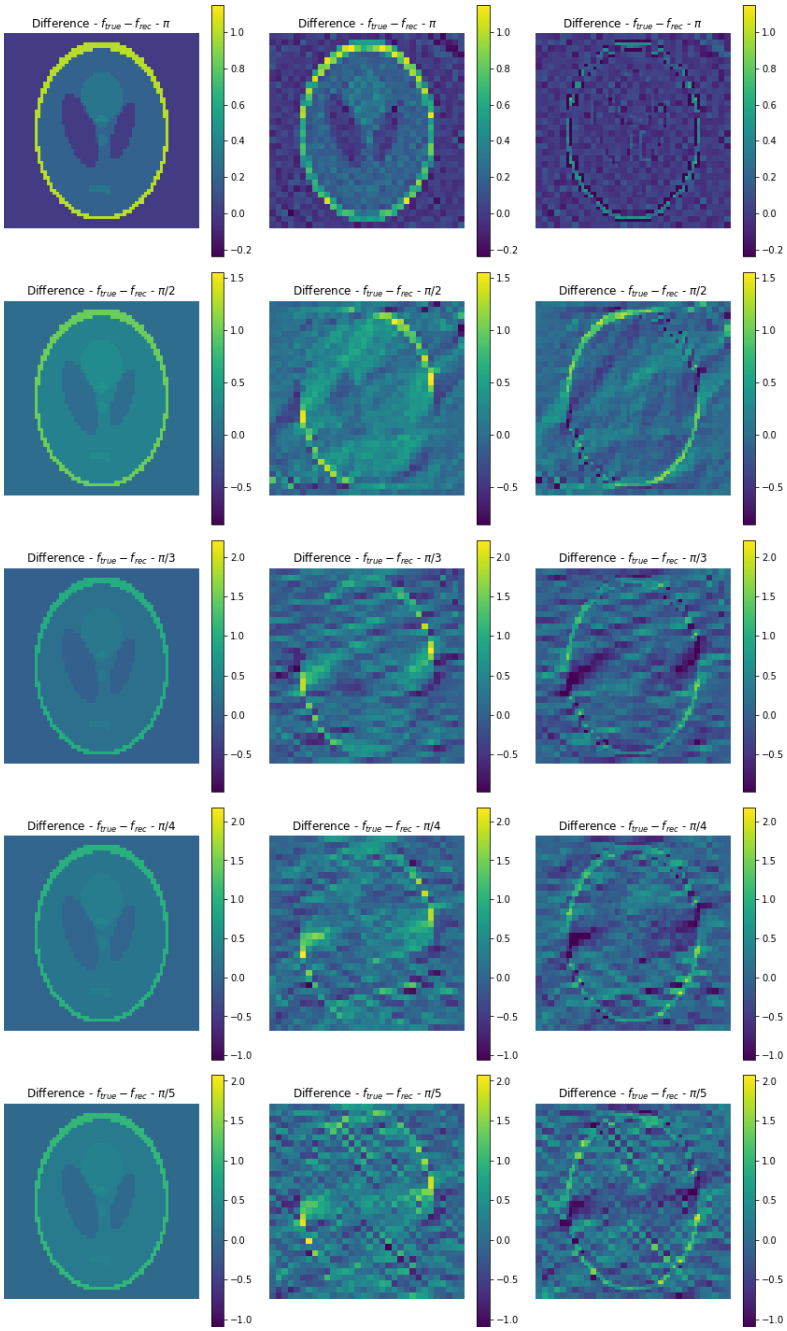

Figure 24: Reconstructions for limited angle ranges

We can see that as the angle range decreases, the reconstructions become progressively distorted. This is similar to what was observed in Question 2) for Tikhonov regularised reconstruction, in which the L2 reconstruction error increased as the maximum angle decreased. Below I plot the reconstruction error. We will see later that the poor

reconstruction is a result of increasingly slow convergence of the iterative soft thresholding method for increasing limited angle ranges, and hence a failure of the iterative procedure to converge to an acceptable tolerance.

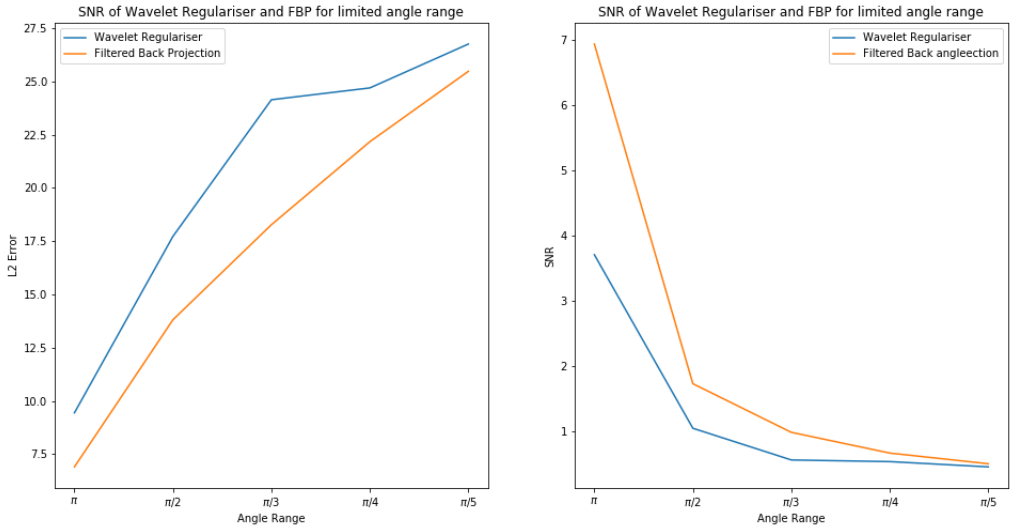

Figure 25: Limited Angle Reconstruction Error - $\|f_{\text{true}} - f_{\text{rec}}\|_2$ and SNR

To test the effect of limited number of projections, I tested cases corresponding to the number of projections $n_{\text{angle}} \in \{10 + 10k : k = 1, \cdots, 8\}$. The results are shown below (for reference, moving down the image corresponds to increasing the number of projections).

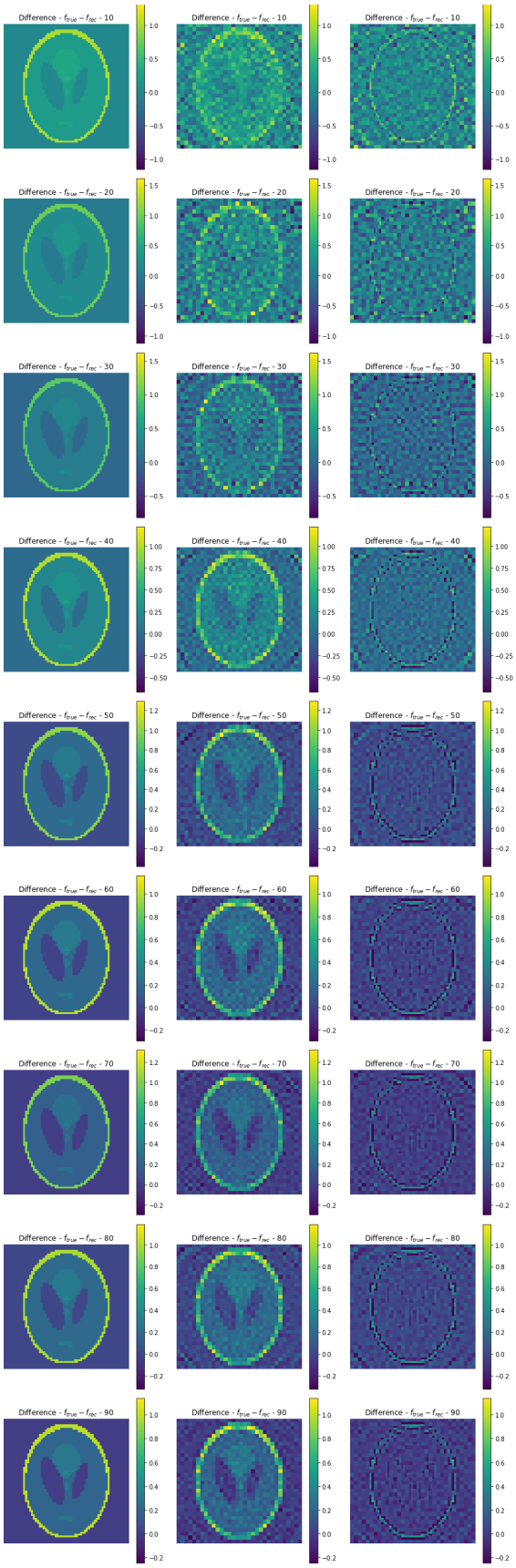

Figure 26: Reconstructions for varying number of angle projections

We can see that increasing the number of projections results in a better reconstruction. This is consistent with the results observed in Question 2. Below I plot the reconstruction error.

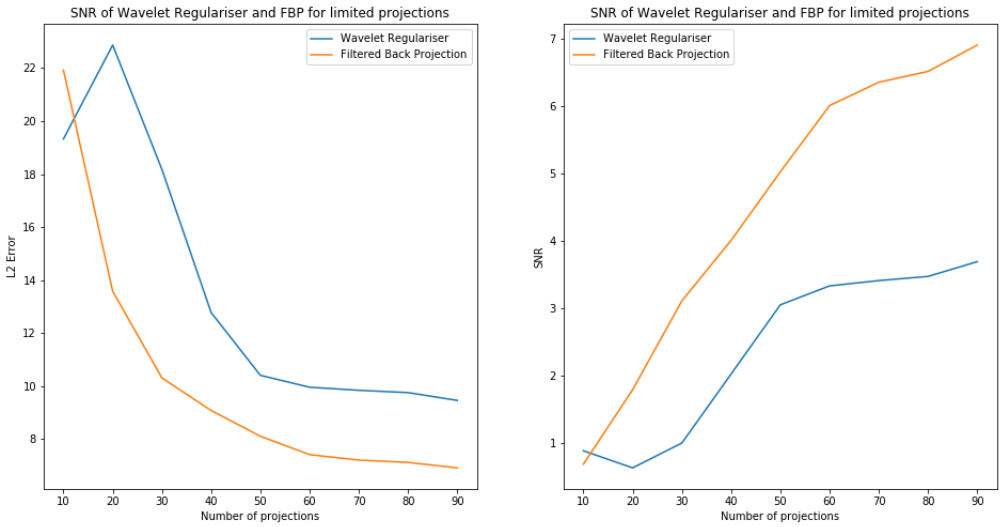

Figure 27: Limited Proj - Reconstruction Error $\|f_{\text{true}} - f_{\text{rec}}\|_2$ and SNR

Now, we examine the convergence plots of the iterative soft thresholding procedure for both tests, which are presented below. We can see that for increasing limited angle projections and ranges, the convergence of the procedure is incredibly slow. The stopping tolerance was set to 1e-6 - after 10000 iterations some cases had barely reached 1e-5. This explains the poor reconstructions visualised above. It could be the result of an inappropriate regularisation value - selecting an optimum value based on an L-curve method was too computationally demanding, and hence selection was based on inspection. In practice I found that the method was sensitive to both the choice of alpha and the step size λ.

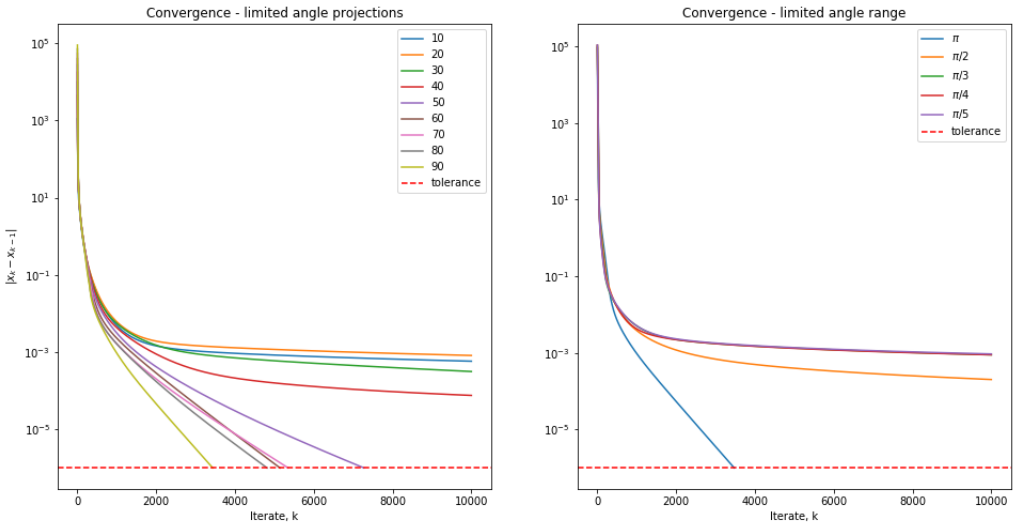

Figure 28: Iterative soft thresholding - convergence

Part B - Advanced Topics

Advanced Topic 2: Low photon count and Poisson noise model

Although the Radon transform and its inverse are correct for integral transform data, actual measured data are of course given for a discrete number of detectors and have random variations due to measurement noise. In part A, you used least squares minimisation, corresponding to normally distributed data. However, when the number of photons is low, a better model for emissions measurements is Poisson statistics. This leads to the negative log Poisson likelihood (NLPL), which is the same as the Kullback-Leibler generalised distance, for the data term.

1. Generate data by using a Poisson pseudo-random number generator based on the Radon Transform i.e $g_{\text{obs}} \sim \text{Poisson}(Af)$. Investigate different levels of noise and their relation to the magnitude of the image.

   ANSWER:

   Suppose we sample N poisson measurements $g \in \mathbb{R}^N$, where $g_i \sim \text{Poisson}(Af_i)$. In order to quantify the size of the added noise in g compared to the original image signal Af, I will use the signal-to-noise metric, defined as follows:

$$SNR = \frac{\|Af\|^2}{\mathbb{E}(\|g - Af\|^2)} \tag{4}$$

In the case of $g \in \mathbb{R}^N$, where $g_i \sim \text{Poisson}(Af_i)$ as in the problem setting, this can be shown to be equal to the following expression:

$$\text{SNR} = \frac{\|Af\|^2}{\sum_{i=1}^N Af_i} \tag{5}$$

How can we change the noise level within the sampled image? Let $g_{\alpha,i} \sim \text{Poisson}(\alpha Af_i)$, where $\alpha \in \mathbb{R}^+$. Then similarly we have:

$$\text{SNR}_\alpha = \frac{\|Af\|^2}{\mathbb{E}(\|g_\alpha - Af\|^2)} \tag{6}$$

Hence by varying α we can change the signal to noise ratio in the resulting image. It is possible to show that SNR_α is equal to the following:

$$\text{SNR}_\alpha = \frac{\|Af\|^2}{\alpha \sum_{i=1}^N Af_i + (\alpha-1)^2 \sum_{i=1}^N Af_i^2} \tag{7}$$

$$= \frac{\sum_{i=1}^N Af_i^2}{\alpha \sum_{i=1}^N Af_i + (\alpha-1)^2 \sum_{i=1}^N Af_i^2} \tag{8}$$

$$= \frac{\text{SNR}_1}{\alpha + (\alpha-1)^2 \text{SNR}_1} \tag{9}$$

Hence we see that increasing alpha does not necessitate a decrease in SNR (not monotonically decreasing), which is what might have been intuitive initially. In fact, one can show that the maximum SNR ratio is achieved when $\alpha_{\max} = 1 - \frac{1}{2\text{SNR}_1}$. Increasing α from this value does result in a decrease in SNR, i.e SNR_α is monotonically decreasing on the interval $[\text{SNR}_{\alpha_{\max}}, \infty)$. Given that $g = \text{Poisson}(\alpha_{\max} Af)$ produces noisy observations with the highest SNR, this will be taken as a start point for the assessment of all reconstruction methods. An example sinogram is shown below.

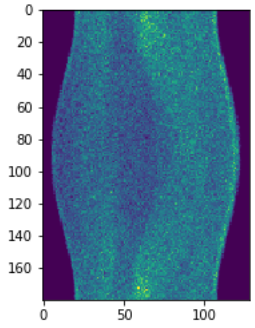

Figure 29: Example sinogram

2. Solve using Weighted least squares reconstruction using Gaussian approximation to Poisson noise. Include an appropriate regularisation as in part A. Use the "plug-in" method where you use the data itself as variance (i.e. $\sigma = \sqrt{g_{obs}}$). Take care to avoid dividing by zero if there are zero photons detected. Handle any zeroes in the measured data.

   ANSWER: One can approximate the poisson data process $g = \text{Poisson}(Af)$ using a multivariate Gaussian, by supposing that $g = N(Af, C)$ where $C = \text{diag}(Af)$.

   Using this Gaussian approximation along with the addition of zero order Tikhonov regularisation, we obtain the following problem to minimise:

$$\min_f \frac{1}{2}(g - Af)^T C^{-1}(g - Af) + \frac{\alpha}{2}\|f\|_2^2 \tag{10}$$

   Given that in practice we dont know $C = \text{diag}(Af)$, one can use the 'plug in' method, which takes $C = \text{diag}(g)$, and hence $C^{-1} = \text{diag}(\frac{1}{g})$.

   This corresponds to a maximum likelihood estimate of the Poisson intensity parameter's Af_i, based on the single data point g_i. In order to 'avoid' dividing by zero when producing our estimate using the data g_i, I will adopt a Bayesian approach and introduce a prior over $\lambda_i = Af_i$. Given that our data g_i is Poisson, I will use the conjugate prior of the Poisson distribution, the gamma distribution. We can write the gamma pdf as

$$p(\lambda_i|\beta) = \frac{\beta_2^{\beta_1}}{\Gamma(\beta_1)}\lambda_i^{\beta_1 - 1}e^{-\beta_2\lambda_i} \tag{11}$$

   Given this parameterisation, it can be shown that $\mathbb{E}(\lambda_i|\beta) = \frac{\beta_1}{\beta_2}$ and $\text{Var}(\lambda_i|\beta) = \frac{\beta_1}{\beta_2^2}$. How should we determine the parameters β? As a reasonable estimate, I match these moments with those of the sample data $g \in \mathbb{R}^N$. In other words, I set

$$\frac{\beta_1}{\beta_2} = \frac{1}{N}\sum_{i=1}^{N} g_i \tag{12}$$

$$\frac{\beta_1}{\beta_2^2} = \frac{1}{N-1}\sum_{i=1}^{N}(g_i - g)^2 \tag{13}$$

$$\tag{14}$$

   This gives a system of two equations with two unknowns, which can hence be solved for. Given this prior, we obtain the following posterior mean:

$$\mathbb{E}(Af_i|\beta, g_i) = \kappa\frac{\beta_1}{\beta 2} + (1 - \kappa)g_i \tag{15}$$

where $\kappa = \frac{\beta_2}{1+\beta_2}$. This expression then gives us a new estimate for our covariance matrix - $C = \text{diag}(\kappa\frac{\beta_1}{\beta 2} + (1-\kappa)g)$. The strength in this method comes from including all the information in the sampled data to estimate a single pixel intensity, providing a consistent way of avoiding inferring zeros.

Using this new estimate, and solving the relevant normal equation using a GMRES solver for various values of regularisation parameter α, I obtain the following L-curve:

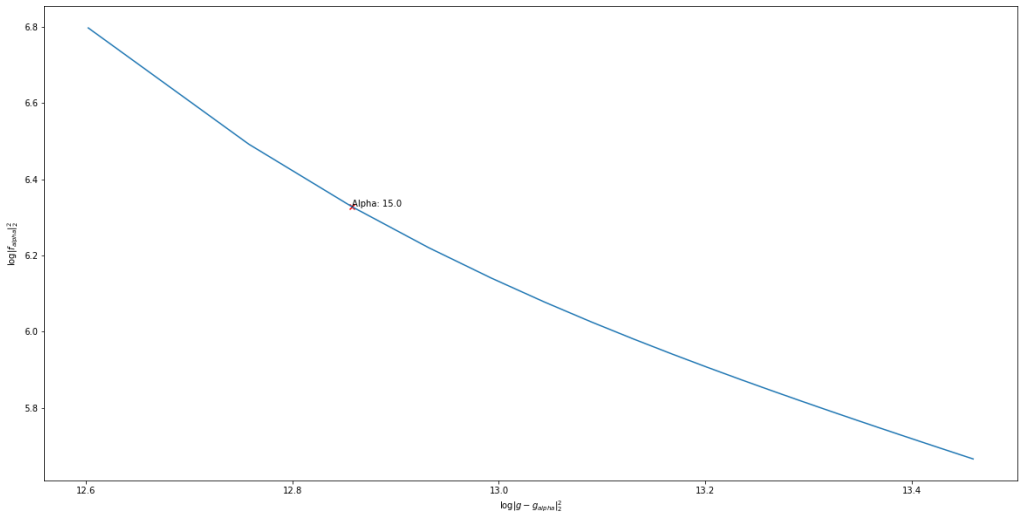

Figure 30: L-curve to determine regularisation value

From the L-curve I determine the optimum regularisation level, $\alpha = 15$. Using this, I obtained the following reconstruction:

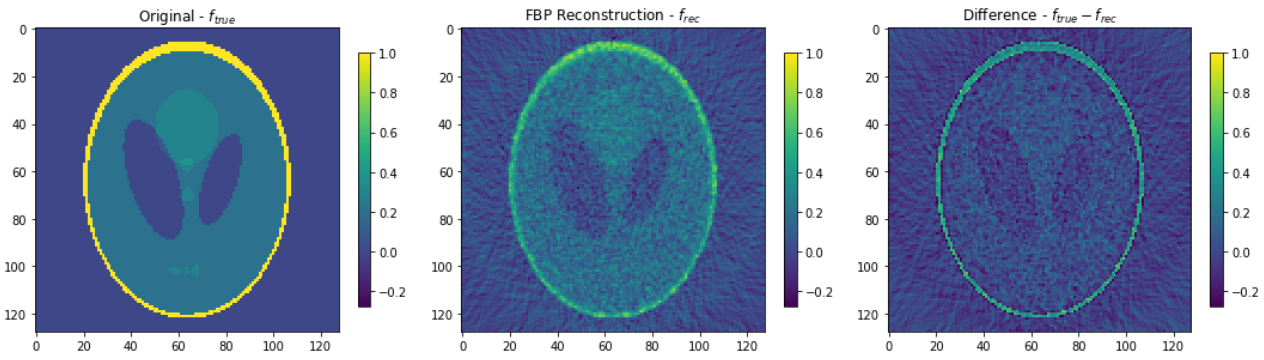

Figure 31: Gaussian Approximation - Reconstruction

Below I plot the reconstruction obtained using filtered back projection. The L2 error of the gaussian approximation method is lower at 0.016, compared to 0.037 achieved using filtered back projection.

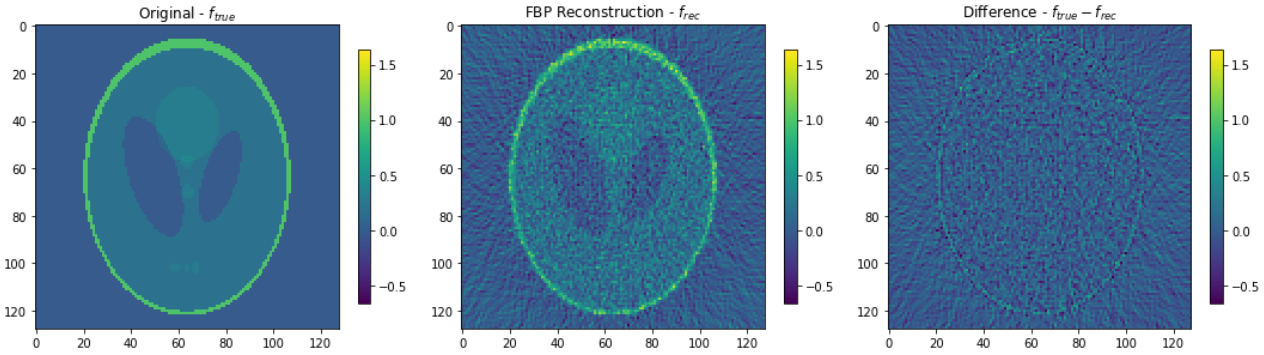

Figure 32: Filtered Back projection - Reconstruction

3. Implement one-step late MLEM. Use both small and large values of the regularisation parameter α. Notice if the algorithm fails when α is too high, e.g. by producing negative values in the reconstruction or by diverging.

   ANSWER:

   The code implementing the one-step-late MLEM algorithm can be found in the appendix.

   Below I show an example solution obtained by setting $\alpha = 15$.

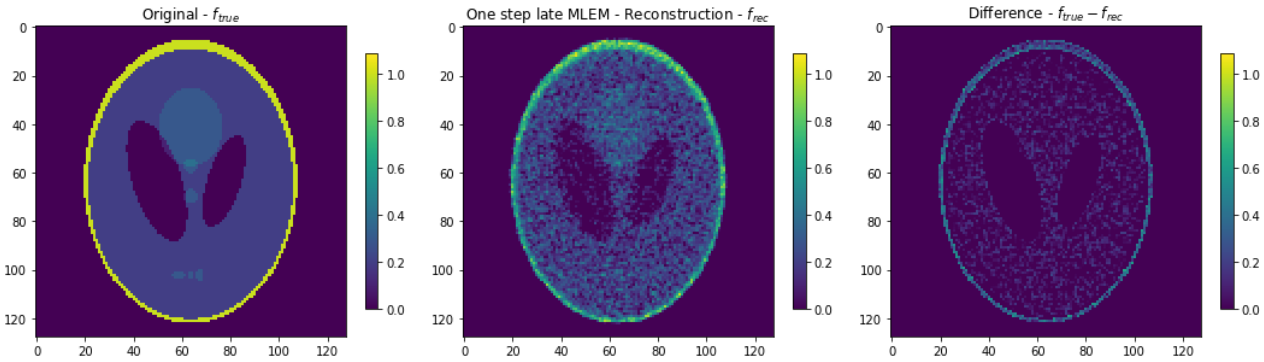

Figure 33: OSL MLEM - Reconstruction

Below I plot the convergence plots for increasing values of the regularisation paramater α. In the range examined, the algorithm did not diverge nor produce negative

reconstructions. I imposed a max iteration condition of 1500, hence some of the reconstructions have not reached the required stopping tolerance (albeit by a small amount in most cases).

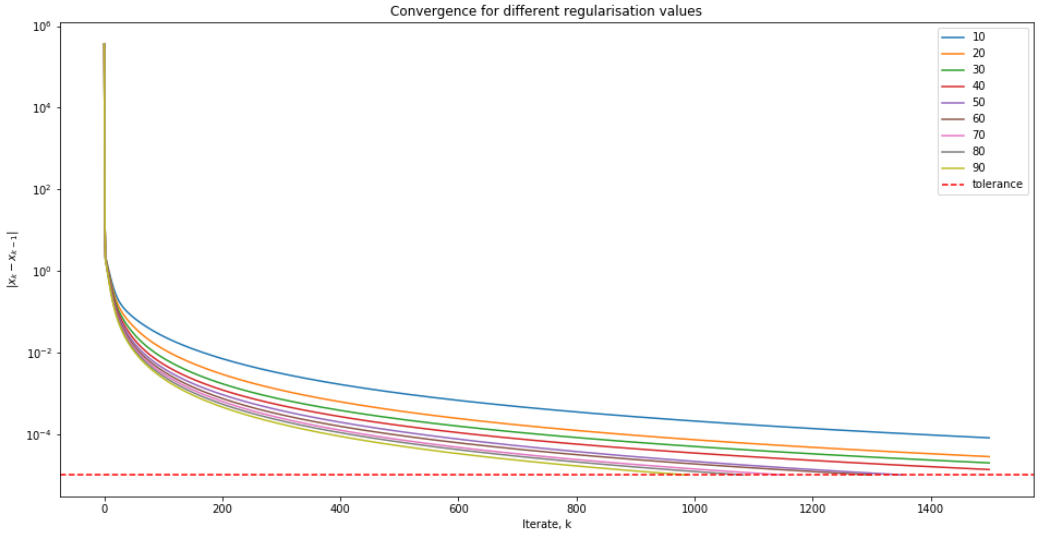

Figure 34: Convergence for varying value of regularisation paramater α

The reconstructions associated with each regularisation value are displayed below.

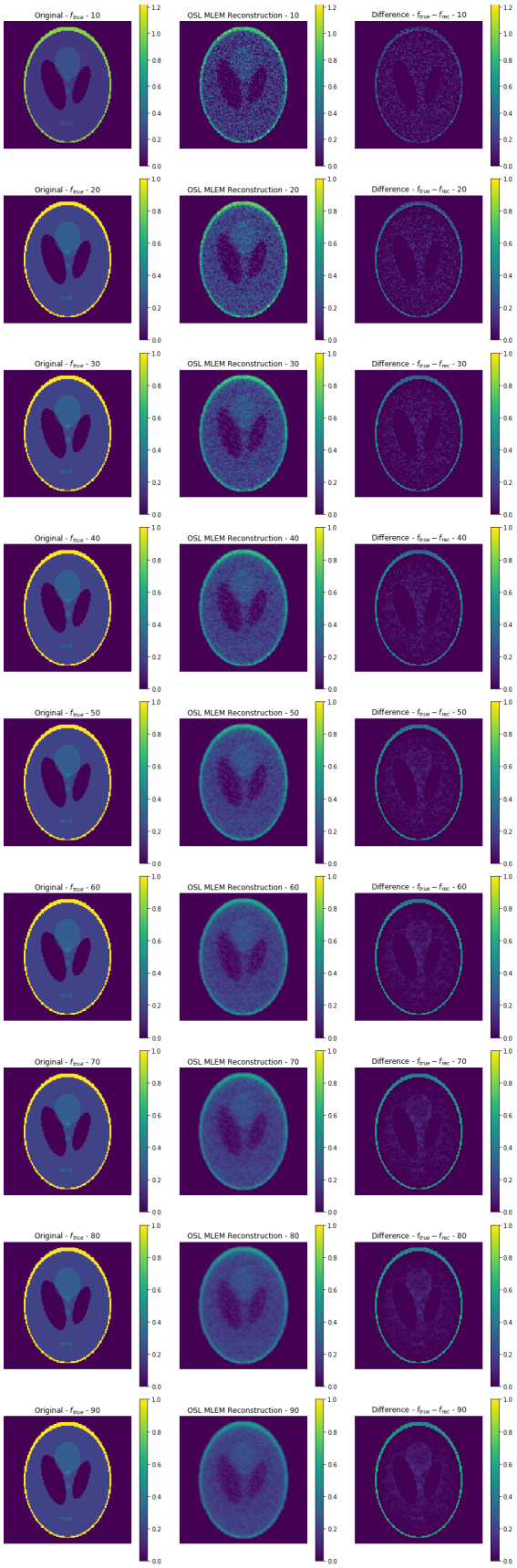

Figure 35: Reconstructions for varying value of regularisation paramater α

Appendix

Q4: Code for Haar wavelet thresholding function

```
1
2  def thresholdFunction(coeffs,tRange,tVal):
3
4      max_tRange = len(coeffs)
5
6      if tRange > max_tRange:
7
8          tRange = max_tRange
9
10     else:
11
12         pass
13
14     coeffs_threshold = list()
15
16     coeff_array, slices = pywt.coeffs_to_array(coeffs)
17
18     tVal_n = np.quantile(abs(coeff_array.flatten()),tVal)
19
20     for cs in coeffs[tRange:]:
21
22         ct = tuple(pywt.threshold(a,tVal_n) for a in cs)
23
24         coeffs_threshold.append(ct)
25
26     coeffsT = coeffs[:tRange] + coeffs_threshold
27
28     return coeffsT
```

Part B: Code for one step late MLEM

```
1
2  from scipy import optimize
3
4  class EM():
5      def __init__(self,g,A,f0,alpha):
6
7          self.g = g
8          self.f0 = f0
9
10         self.alpha = alpha
11
```

```
12          self.A = A
13
14          self.one = np.ones(g.shape)
15
16          self.f_k = None
17
18      def rl_operator(self,f):
19
20          return (f/(np.dot(self.A.T,self.one)))*(np.dot(self.A.T,(self.g/np.dot(
                self.A,f))))
21
22      def osl_operator(self,f):
23
24          return (f/(np.dot(self.A.T,self.one) + self.alpha*f))*(np.dot(self.A.T,(
                self.g/np.dot(self.A,f))))
25
26      def richardson_lucy_update(self):
27
28          f_k1 = self.rl_operator(self.f_k)
29
30          self.f_k = f_k1
31
32      def osl_MLEM_update(self):
33
34          f_k1 = self.osl_operator(self.f_k)
35
36          self.f_k = f_k1
37
38      def solve(self,maxIter,tol,update_method,opt_method):
39
40          if update_method == 'richardson-lucy':
41
42              r = optimize.fixed_point(self.rl_operator, self.f0,xtol = tol,maxiter
                    = maxIter,method = opt_method)
43              self.result = r
44
45          elif update_method == 'osl':
46
47              r = optimize.fixed_point(self.osl_operator, self.f0,xtol = tol,maxiter
                     = maxIter,method = opt_method)
48              self.result = r
49          else:
50              pass
51
```

```
52
53      def solve_it(self,maxIter,tol,method):
54
55          if method == 'richardson-lucy':
56
57              stop_condition = True
58              self.nIter = 0
59              self.f_k = self.f0
60
61              while (stop_condition & (self.nIter < maxIter)):
62
63                  f_k_1 = self.f_k
64
65                  self.richardson_lucy_update()
66
67                  stop_condition = np.linalg.norm(self.f_k - f_k_1) > tol
68
69                  self.nIter = self.nIter + 1
70
71
72          elif method == 'osl':
73
74              stop_condition = True
75              self.nIter = 0
76              self.f_k = self.f0
77              self.difference_k = np.inf
78              nDiv = 0
79
80              residuals = list()
81
82              while (stop_condition & (self.nIter < maxIter)):
83
84                  f_k_1 = self.f_k
85
86                  self.osl_MLEM_update()
87
88                  if np.linalg.norm(self.f_k - f_k_1) > self.difference_k:
89
90                      nDiv = nDiv + 1
91
92                  else:
93                      pass
94
95                  self.difference_k = np.linalg.norm(self.f_k - f_k_1)
```

```
96
97                  residuals.append(np.linalg.norm(self.f_k - f_k_1))
98
99                  stop_condition = self.difference_k  > tol
100
101                 self.nIter = self.nIter + 1
102
103             self.residuals = residuals
104
105         else:
106
107             pass
```