COMP0114 - Coursework 4

Due: Tuesday April 17th, 2023 at 16:00

PART A

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

We will work with the Shepp-Logan phantom of resolution 128×128 (Figure 1 top-left). In this first part, we will investigate the Radon transform, so that we can use it in the next parts.
The principle of the Radon transform is that we send rays through the object we are scanning, we parameterize this line with an angle, and the value of this line is the integral of the values along it (Figure 2). This is suitable for tomography, where we send X-rays through the body to detect them on the other side, and we move the X-ray source around the patient to reconstruct the whole image.

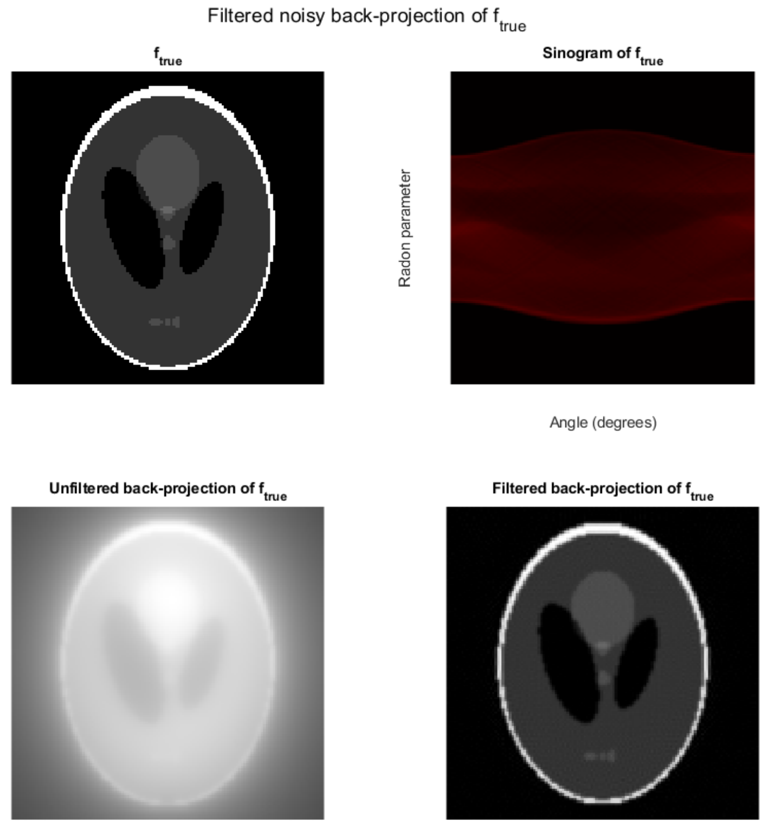

Filtered noisy back-projection of f_{true}
f_{true}
Sinogram of f_{true}
Radon parameter
Angle (degrees)
Unfiltered back-projection of f_{true}
Filtered back-projection of f_{true}

Figure 1: Radon transform of the Shepp-Logan phantom. f_{true} image (top-left), sinogram of f_{true} (top-right), unfiltered back-projection of f_{true} (bottom-left), filtered back-projection of f_{true} (bottom-right)

When we have sent enough rays (i.e covered all around the object), we can superimpose all the sinus functions obtained on a **sinogram**. We can see the sinogram of our Shepp-Logan phantom in Figure 1 (top-right). The size of the sinogram is 185×180 (180 measurements, values going in 185 bins). The sinogram is obtained using MATLAB's

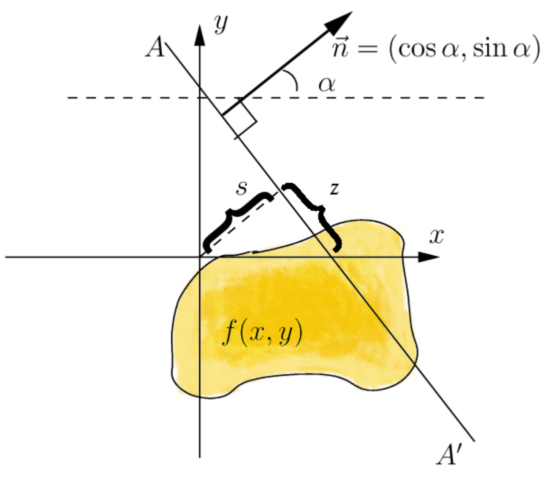


Figure 2: Image representing the principle of the Radon transform.
(Source: https://upload.wikimedia.org/wikipedia/commons/5/5d/Radon_transform.png)

`radon` function.
When we do not cover all the object, or that we do not make enough measurements during scanning, we can see that we either recover only part of the object, or we have artifacts on the reconstructed image (respectively top and bottom in Figure 3). We can clearly see from the shape of the artifacts/missing parts that we perform reconstruction based on sinus functions. The size of these sinograms is 185×45 (45 measurements, values going in 185 bins). The 4-degree separation method achieves a most informative reconstruction because it sees the whole object.

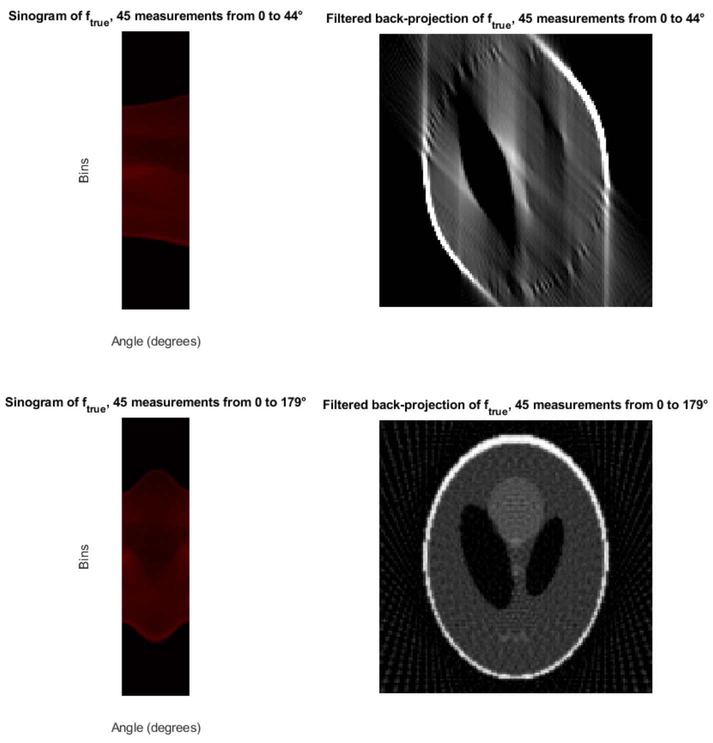

Sinogram of f_{true}, 45 measurements from 0 to 44°
Bins
Angle (degrees)
Filtered back-projection of f_{true}, 45 measurements from 0 to 44°
Sinogram of f_{true}, 45 measurements from 0 to 179°
Bins
Angle (degrees)
Filtered back-projection of f_{true}, 45 measurements from 0 to 179°

Figure 3: Radon transform of the Shepp-Logan phantom, but with incomplete measurements. 45 measurements from 0 to 40° (top): sinogram of f_{true} (top-left), filtered back-projection of f_{true} (top-right); 45 measurements from 0 to 179° (bottom): sinogram of f_{true} (bottom-left), filtered back-projection of f_{true} (bottom-right)

To check the reconstruction performance, we use MATLAB's `iradon`. We have the option to use unfiltered or filtered reconstruction, respectively shown at the bottom-left and bottom-right of Figure 1.
We can see that the unfiltered back-projection is really blurry, and that the filtered back-projection gives a much better result. We can deduce that the high-frequencies have been filtered out on the filtered back-projection, giving a more accurate result (see the documentation of `iradon`, there is a band-pass filter applied by default). We have two things to note. Firstly, despite being a good reconstruction, the filtered back-projection is not exact, and we can observe some blurring. Secondly, the unfiltered back-projection needs to be scaled,. We can perform an adjoint test to find the scaling factor. The image in Figure 1 shows the normalized image. We will precise in the following parts when we need to scale, and what scaling value we have used.

Let's now put to the test the performance of the filtered back-projection, by adding noise to the sinograms:

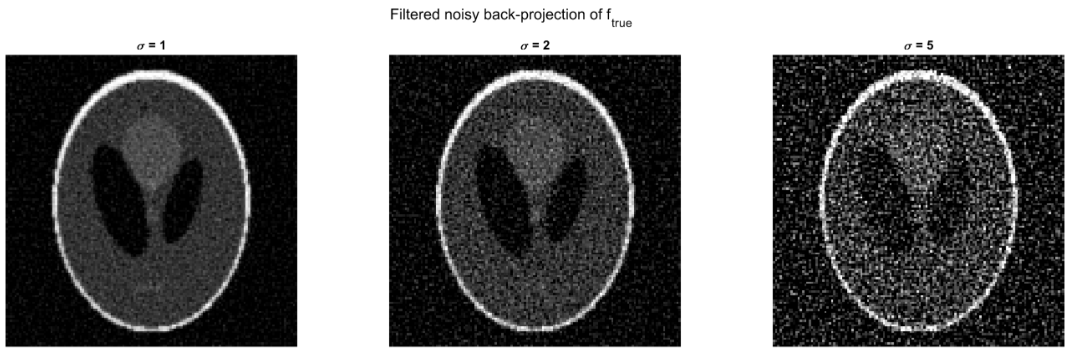


Figure 4: Filtered back-projection of noisy sinograms of f_{true}, for different noise levels.

As expected, we can see noisy outputs in Figure 4. Yet, the noise is spread evenly on the images, whereas we expected modifications in the shape of the phantom given that the noise could have altered some sinus functions. Maybe it is the filter applied by default that has smoothed the noise. Let's test for the unfiltered version of `iradon`:

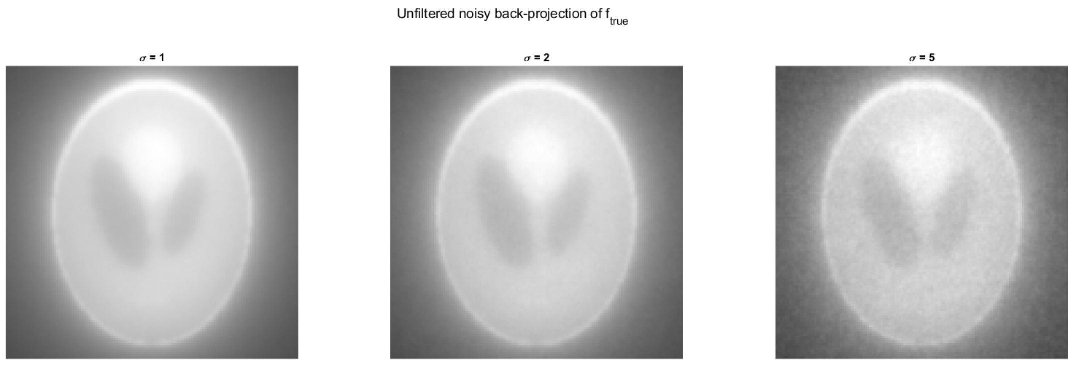
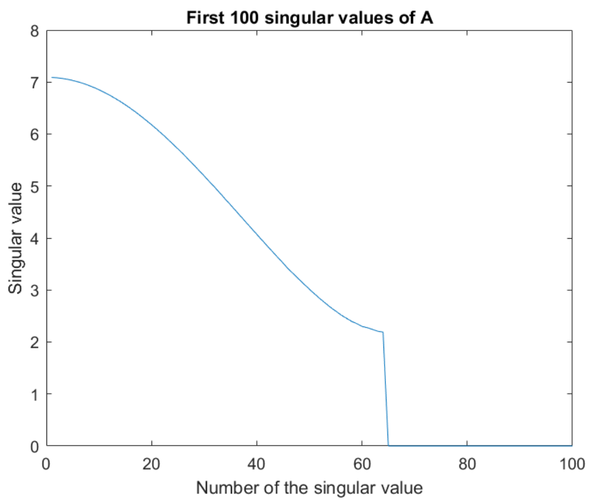
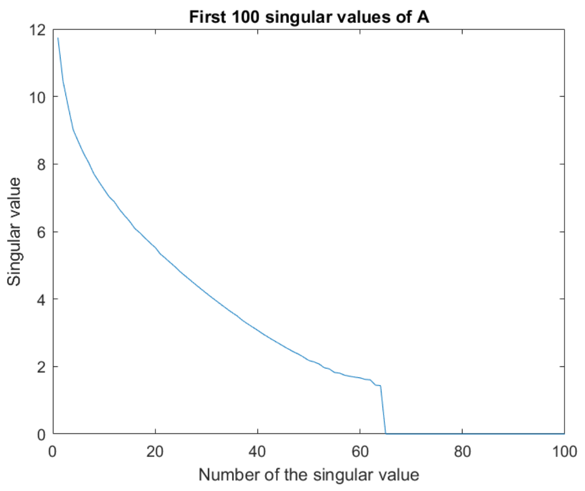


Figure 5: Unfiltered back-projection of noisy sinograms of f_{true}, for different noise levels.

The conclusion is similar in Figure 5. Even though this time we see less noise, probably because it blends with the original high-frequency blur of the unfiltered back-projection. We can therefore say that even though the error in reconstruction grows quite linearly with the measurement noise, it does not modify the main shapes of the input.



(a) 45 measurements between 0 and 45°



(b) 45 measurements between 0 and 179° (4° separations)

Figure 6: First 100 spectral values from the SVD of the explicit Radon transform matrix

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

By looking at the spectrum of the Radon transform explicit matrix, we will determine if it is a good idea to implement it directly in our methods.
First of all, if we stuck to 180 measurements (from 0 to 179°, like in part 1) and a 128×128 phantom, our Radon matrix would have dimensions 14800×16384, which is not ideal for storage and performance. So, we will limit ourselves to 45 measurements (either doing only part of a rotation from 0 to 44°, or doing a whole rotation but missing some measurements, from 0 to 179° with 4-degree separations) and downsize the phantom to a 64×64 resolution. Our explicit matrix is now 4275×4096 (13 times smaller). Let's add that because we have less measurements, and that the Radon transform of each individual pixel is a sinus function, we will use a sparse matrix, which now makes the explicit matrix a reasonable option. Let's take a look at its spectrum.

The main difference we can see between Figures 6a and 6b is the decay of the spectrum: it is exponential for the 0-179° range with missing measurements, and it is more stable for the 0-44° range (although not ideal either). But, the main point here is that when acquiring limited data there is a sudden drop in the spectrum after around 60 spectral values. This makes the matrix extremely ill-conditioned. As seen in previous courseworks, a solution could be to truncate/filter this SVD, but looking at the decay of the spectrum in either case, a better solution would be to implement a matrix-free solver for the Radon transform.

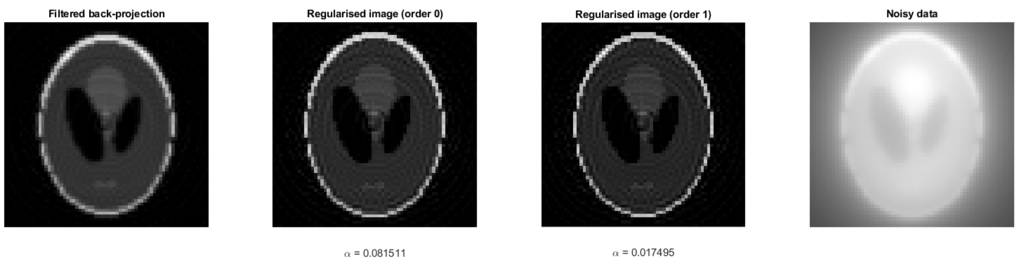
3 Implement a matrix-free regularised least-squares solver for the Radon Transform.

The equation we are implementing is:

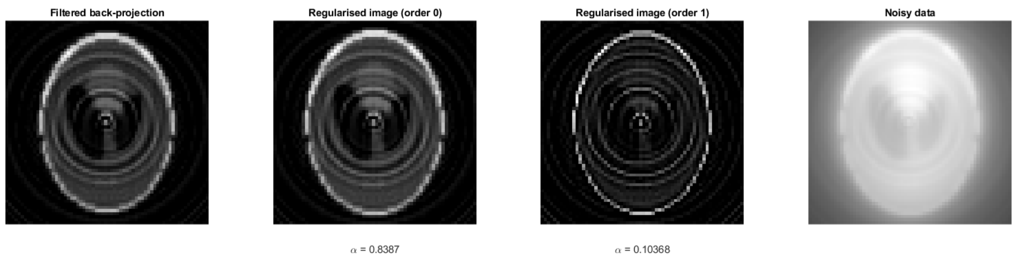
$$(A^T A + \alpha L)\boldsymbol{f}_+ = A^T \boldsymbol{g}$$

where A represents the Radon transform, A^T is the unfiltered Radon back-projection, α is the regularisation parameter, L the regularisation matrix (the identity matrix for the zero-order Tikhonov regularisation, and the gradient operator for the first-order), $\boldsymbol{f}_+$ is our estimate (for the first iteration, it is estimated as the zero vector), and $\boldsymbol{g}$ is our data i.e a noisy sinogram.
Just like in coursework 2, we will use the preconditioned conjugate gradients method (`pcg` in MATLAB). Therefore, for the left-hand side of the equation, we will implement a function handle in charge of applying the `radon`/`iradon` transforms to our estimate as well as adding the regularisation term. For the right-hand side, we just add noise to our sinogram, and we give it as an input for `pcg` to compare with the estimate at each iteration.
We use the Discrepancy Principle to determine the optimal value for α.
The following results will show the reconstruction of our regularized methods versus the Radon filtered back-projection. We have used different noise levels, and different measurements (presented in part 2).

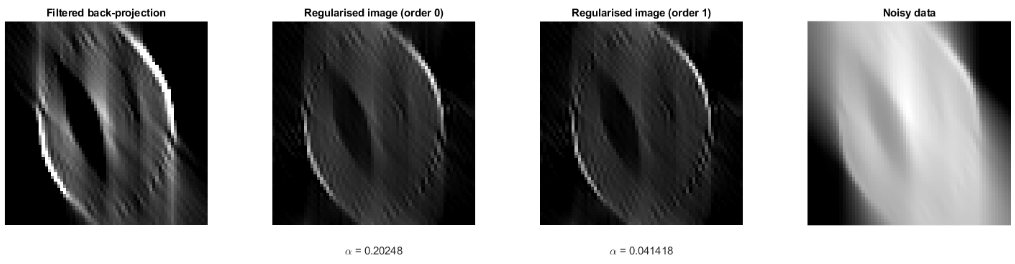
We can see that the noise manifests itself in the form of circles. It is because the noise is applied to the sinogram, and so it is back-projected as circles.
Overall, when comparing the filtered back-projection (Figure 7 column 1) with the zero-order regularised output of our method (Figure 7 column 2), we can see that our routine achieves less of a blurry image. We can say it is because it iteratively filters out the high-frequency components of the noisy data compared to the filtered back-projection that only filters once.
The noise is hard to remove above a certain level, here at $\theta = 0.5$, it becomes already hard to smooth out. Probably because when there is too much noise, the noise itself is not sparsely found in the sinogram, but rather creates new sinewaves, which are then considered as original parts of the sinogram.
Finally, the difference between zero and first-order regularisation (Figure 7 column 2 VS column 3) is that the latter seems sharper, which is expected given that we regularise all the more on the edges of the image. The noise is also reconstructed more sharply, but there is no major improvement.
Regularisation shows its limits in the context of noisy sinograms. Let's investigate another method.



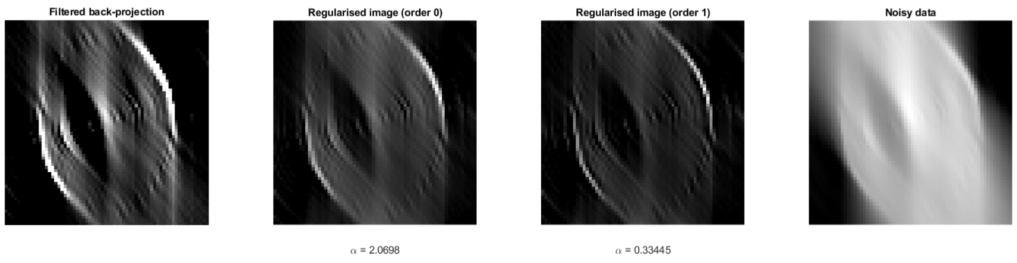
(a) Full measurements (0-179° range) with noise level $\theta = 0.1$



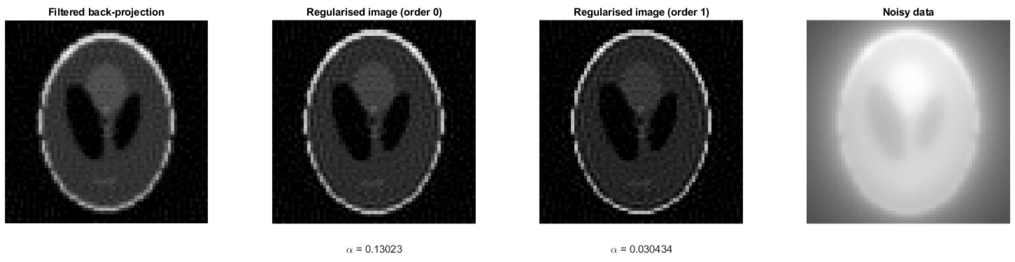
(b) Full measurements (0-179° range) with noise level $\theta = 0.5$



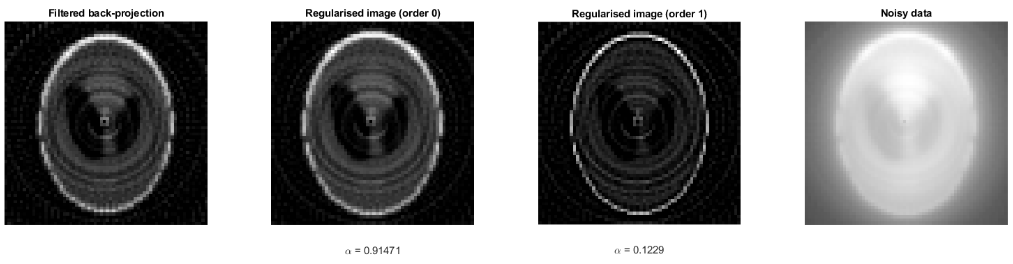
(c) Limited measurements (0-44° range) with noise level $\theta = 0.1$



(d) Limited measurements (0-44° range) with noise level $\theta = 0.5$



(e) Incomplete measurements (0-179° range with 4° separations) with noise level $\theta = 0.1$



(f) Incomplete measurements (0-179° range with 4° separations) with noise level $\theta = 0.5$

Figure 7: Comparison of our regularized reconstructions (column 2 for zero-order Tikhonov regularization, column 3 for first-order) versus the filtered Radon back-projection (column 1) for different measurements and noise levels. The noisy data is shown in column 4.

4 Write a Haar wavelet denoiser.

The new approach we can take is using a Wavelet transform and threshold its coefficients, and then taking the inverse transform. The Wavelet transform (in our case, the Haar wavelet transform, obtained with MATLAB's `haart2`) is a multi-resolution sparse representation of the details of an image. Here are for example the coefficients of the *cameraman256.png* image:

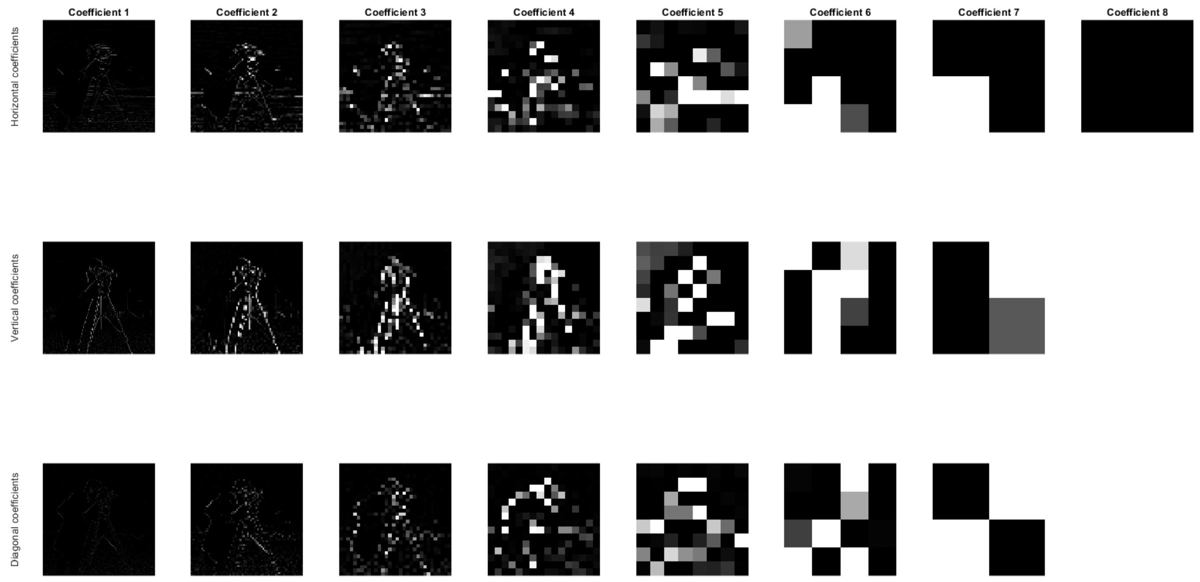


Figure 8: Representation of the Haar Wavelets coefficients for our image. Along the rows: horizontal, vertical, diagonal coefficients. Along the columns: precision of the coefficient.

Taking for example the first row of Figure 8, we can see the high-frequency details of the *cameraman256.png* in different resolutions: the pixels get bigger and bigger, and the representation more and more coarse.
When introducing noise, if kept to a reasonable level, there should be some pixels showing up in the coefficients, but not as bright as the high-frequency ones representing the main shapes of the image. Therefore, we can guess that thresholding the coefficients based on pixel intensity, will remove the low-intensity pixels corresponding to noise (and possibly the low-frequency parts of the image too).
To do so, we will follow the hint given in the coursework sheet by implementing a thresholding function. We gather all the coefficients' pixels, we sort them and we choose the thresholding value based on a given percentage of pixels we want to turn off. Because the Wavelet representation is sparse, we can clearly discriminate between high-frequency and noisy/low-frequency parts (the pixels are either high or low-intensity, there is not really a middle-ground). We then use MATLAB's `wthresh` function to threshold the coefficients directly (we can use soft or hard thresholding). We also add the option to choose which range of coefficients we want to threshold. Looking at Figure 8, the higher coefficients we modify, the more impact we will have: because for example in the first coefficient, the camera is represented in a 30×30 pixel window, whereas in coefficient 5, it is only one pixel (that could be lower in intensity than in coefficient 1). Therefore, we need to be cautious about our coefficient range, otherwise we will remove important details (we can maybe think about an adaptative thresholding value as an improvement for our method).
Here are our results for a varying range and different percentage values:

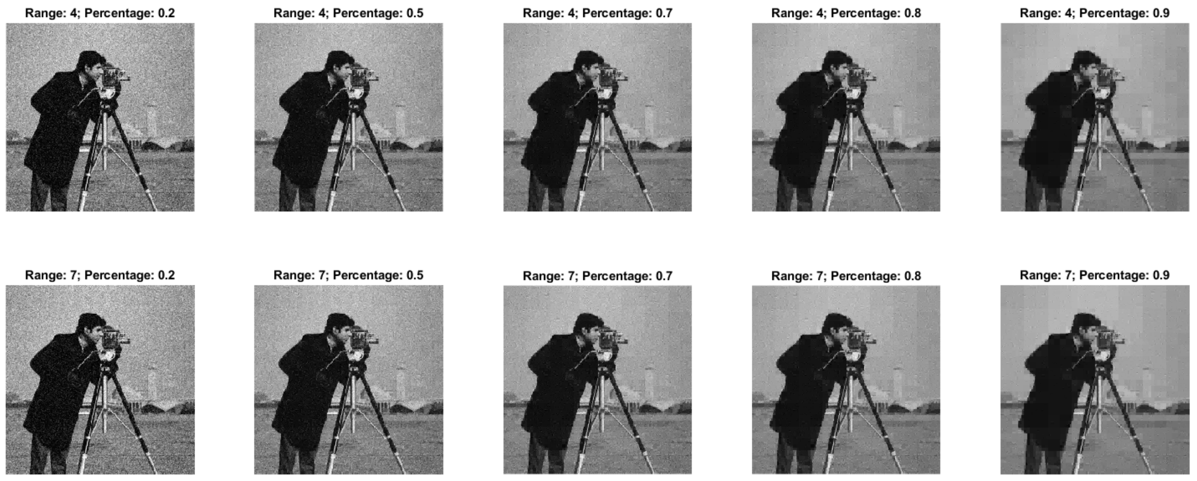

Range: 4; Percentage: 0.2
Range: 4; Percentage: 0.5
Range: 4; Percentage: 0.7
Range: 4; Percentage: 0.8
Range: 4; Percentage: 0.9
Range: 7; Percentage: 0.2
Range: 7; Percentage: 0.5
Range: 7; Percentage: 0.7
Range: 7; Percentage: 0.8
Range: 7; Percentage: 0.9

Figure 9: Output of our Wavelet coefficients thresholding function for different ranges and percentage values. Noise level: $\theta = 0.1$. Hard thresholding.

We can see on Figure 9 the effect of two things:

- **The percentage value:** we do not really see a difference below 70% of percentage value. This is because the Wavelet representation is very sparse, therefore the vast majority of pixel values are low, and do not carry much information. Starting at 70%, we start to see blur around the edges, and overall an impression of "squared" patches, rather than a continuous set of pixels.
- **The coefficient range:** This impression of "squared" patches is emphasized as we process more coefficients. When comparing the last column of Figure 9, we can see in the upper-right part of the image a grid of squares. The squares for the 7 range are almost twice the size of the squares for the 4 range. We understand that by thresholding the lower-resolution squares (cf the last 3 columns of Figure 8), we also affect the resolution of the output image.

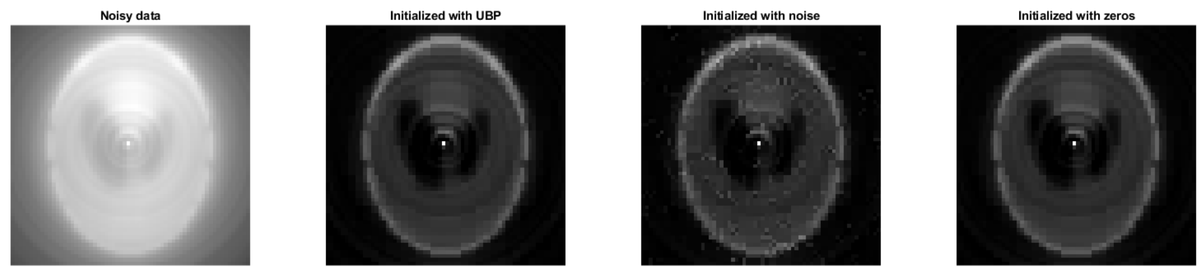
As far as denoising is concerned, it works. Starting at 70%, the amount of noise has decreased. The more we threshold, the less noise there is, so our assumptions about how sparse the representation is and how low-intensity the pixel values are related to noise are correct. Yet, choosing a higher range of coefficients does not achieve a better denoising performance. We can understand that because the lower the resolution, the less represented the noise is (e.g in a 30×30 pixel window, there could be a few pixels attributed to noise, but in a single low-resolution pixel it is impossible to distinguish between the high-frequency/low-frequency/noise contributions).
Let's now combine this denoising technique with an iterative method to improve what we have already seen in task 3.

5 Iterative soft-thresholding for X-ray tomography

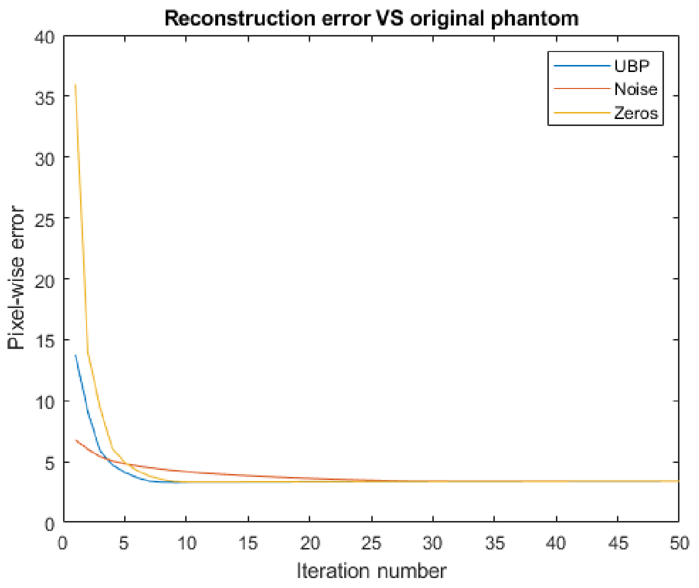
In this section, we will implement the following iterative algorithm:

$$\boldsymbol{f_{k+1}} = S_{\alpha,W}(\boldsymbol{f_k} - \lambda A^T(A\boldsymbol{f_k} - \boldsymbol{g})), \qquad S_{\alpha,W}(\boldsymbol{f}) = W^{-1}S_\mu W\boldsymbol{f}$$

where $\boldsymbol{f}$ is our estimate, $\boldsymbol{g}$ the data, A the Radon transform operator, W the Wavelet transform operator, k the iteration number, and S is our thresholding function introduced in task 4. λ, α and $\mu = \lambda\alpha$ are hyperparamters.



(a) Noisy data and outputs for different initializations.



(b) Reconstruction error compared to the original Shepp-Logan phantom for different initializations.

Figure 10: Results of our iterative algorithm for different initializations. Hyperparameters: $\lambda = 0.01$ and $\mu = 0.6$.

This is an algorithm to obtain an estimate for $\min \frac{1}{2}\|A\boldsymbol{f} - \boldsymbol{g}\|_2^2 + \alpha\|W\boldsymbol{f}\|_1$, where we recognize a classic optimization problem with the Wavelet transform playing the role of regularisation. The norm 1 indicates that we favor the sparse solutions (just like we have seen in coursework 3 with the paper from Candès et al.). Here, the soft-thresholding operator is used because the norm 1 is not differentiable, so we have to find a way around it.
We will start by creating a noisy sinogram, which will be our data, and we decide to try taking as the initial iterate $\boldsymbol{f_0}$ either the unfiltered back-projection of the noisy sinogram, or noise, or only zeros.
For the stopping criterion, we will compare our estimate at each step with either the ground truth (if we consider that we can have access to it) or the filtered back-projection. We can also stop when the difference between two estimates stays small over multiple iterations.
During the reconstruction with the inverse Wavelet transform, we apply an absolute value to the image so that we only have pixels with a positive value (prior information).
Let's first see how our algorithm performs with different initializations.

Based on the results from Figure 10b, we see that we generally take around 30 iterations to reach a stable output whatever the initialization strategy is. Though, by looking at the phantom outputs in Figure 10a, we can see that the best results are obtained

by initializing using the unfiltered back-projection (UBP) or zeros. The noisy initial estimate remains quite noisy. We will therefore stick with the UBP as our $\boldsymbol{f_0}$. Let's now experiment different hyperparameters.
Besides, our $\lambda = 0.01$ value seems adequate, the reconstruction is ok and the number of iteration reasonable, so we will also keep this value as is.
Here are the results for our soft-thresholding algorithm below, for a varying noise level, and different projection geometries:

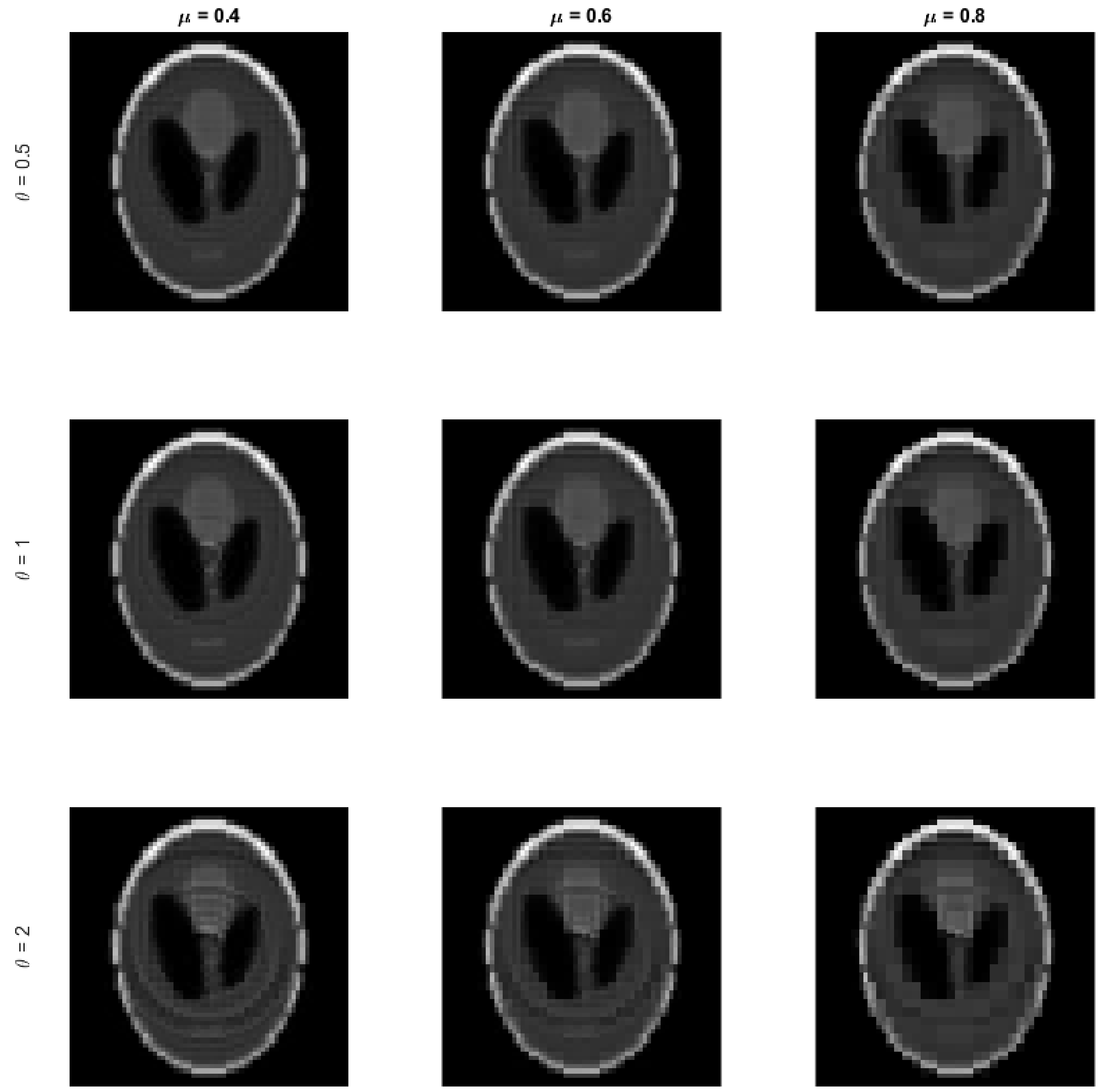


Figure 11: Results of our iterative algorithm for different noise levels and different hyperparameters. Measurements taken between 0 and 179°. $\lambda = 0.01$.

In Figure 11, we can see that the reconstructions are acceptable. For a low level of noise, taking a low μ coefficient gives us an output that almost matches the filtered back-projection in Figure 7a, which proves the efficiency of our algorithm. Taking a bigger μ makes the output more "squared", as if the resolution had decreased, which is an effect we had already highlighted in task 4.
For $\theta = 1$, we can see that taking a bigger μ makes the reconstruction better, because of

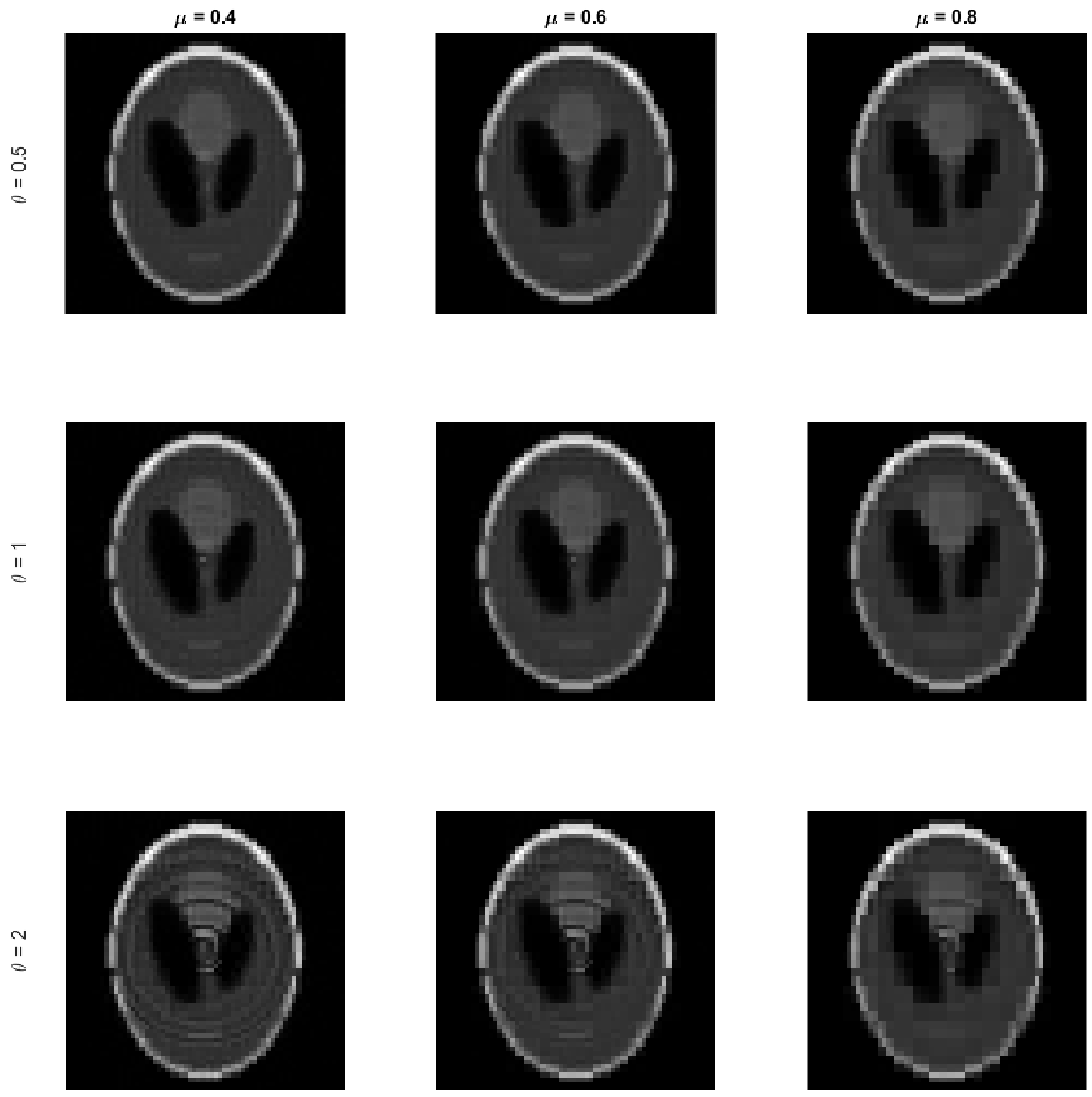


Figure 12: Results of our iterative algorithm for different noise levels and different hyperparameters. Measurements taken between 0 and 179° with a 4° separation. $\lambda = 0.01$.

the Wavelet denoising.
Even for $\theta = 2$, reconstructions are quite ok. We have proven that on a full set of measurements, our soft-thresholding method proves to be more efficient than the regularised least-squares solver by favoring sparsity.

What about it when we have missing measurements? In Figure 12, we can observe that for a reasonable level of noise (first two rows), the outputs are similar to the ones in Figure 11. It shows that our method works even on a low number of measurements (from 180 to 45 measurements in our case). This could be very useful given that scanning can be a long and stressful process for patients. Let's note that this method can be more vulnerable to high noise, as we can see in the third row.

Let's also add that, in order for us to take fewer measurements, we still have to respect one rule: the measurements need to span the whole image, and not only a part of it. We

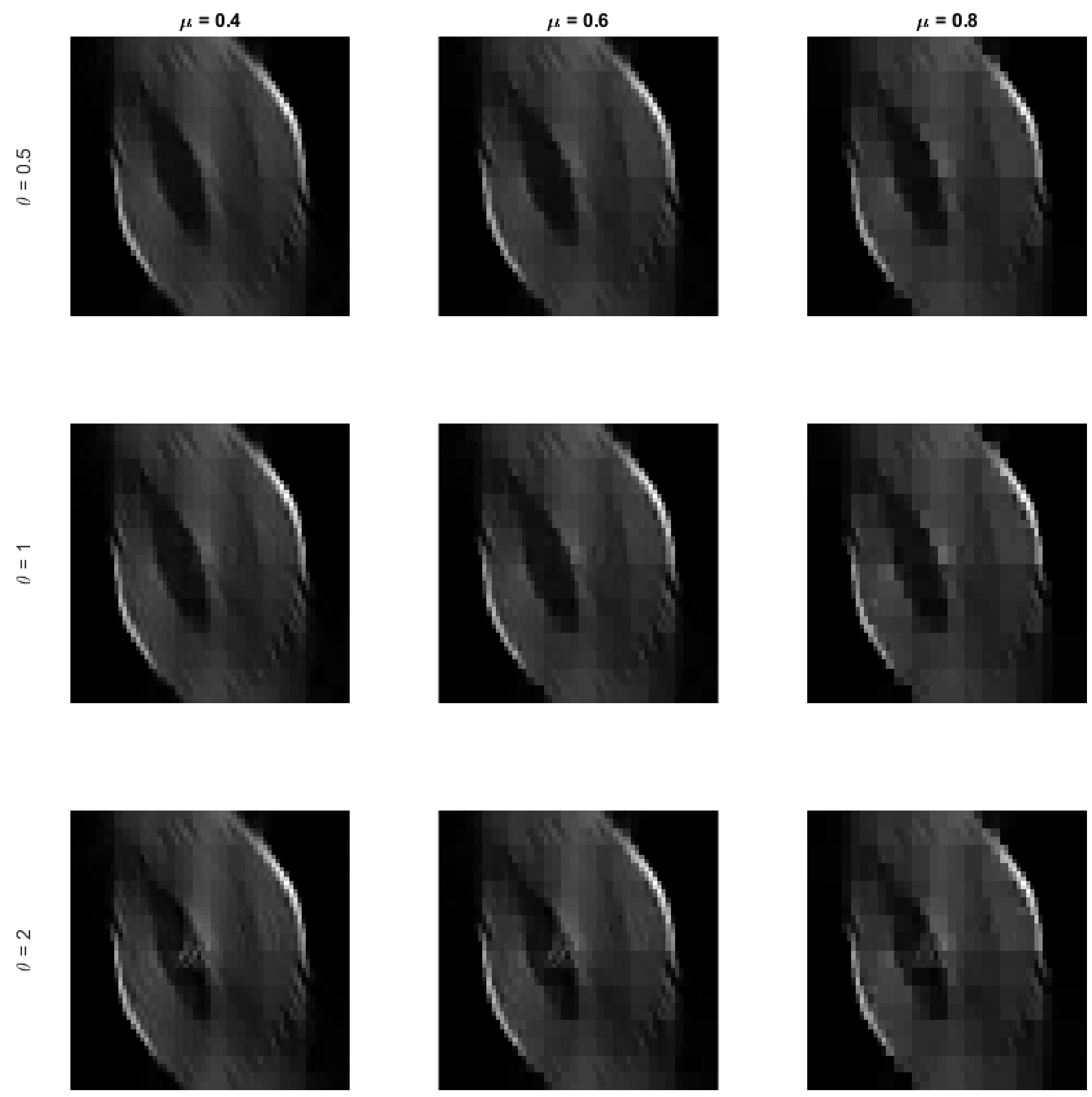


Figure 13: Results of our iterative algorithm for different noise levels and different hyperparameters. Measurements taken between 0 and 44°. $\lambda = 0.01$.

can clearly see in Figure 13 that this reconstruction is way less informative than the previous ones, because we have only taken measurements between 0 and 44°, not seeing all parts of the Shepp-Logan phantom. If we want to observe something contained in that part of the image, it may be useful, because we see that our iterative algorithm still performs well. But, we would need to introduce prior information about the ground truth so that our algorithm could try to reconstruct it in full.

To conclude, in this first part of the coursework, we have investigated ways to reconstruct an image from a sinogram obtained by Radon transform. Posing this problem as an optimization problem, we have explained and tested different algorithms to solve this. In the end, using the Wavelet transform as a denoiser and using a norm-1 regularisation to promote sparsity, we have achieved decent results with a soft-thresholding iterative algorithm, that allows us to obtain good results with a lower number of measurements, given that the measurements are informative enough.

PART B

We have chosen to study the advanced topic 1: Inpainting in Sinogram Space.

What is different in this part, is that instead of considering sinograms for limited and undersampled angles (that would be full sinograms but with smaller dimensions, refer to Figure 3), we consider the sinogram for the full range of angles (0-179°) but erasing some parts of it that would be considered as limited or undersampled angles. The sinogram would have its full size 95×190 but would carry less information. In part A, we focused on the image space. Here, we will try filling the gaps we have in the sinogram space. Refer to Figure 14 to see the original context (full phantom and sinogram) on the first row, the sinograms we will try to inpaint on the second row, and the mask identifying the regions of interest (ROI, to inpaint) on the third row.

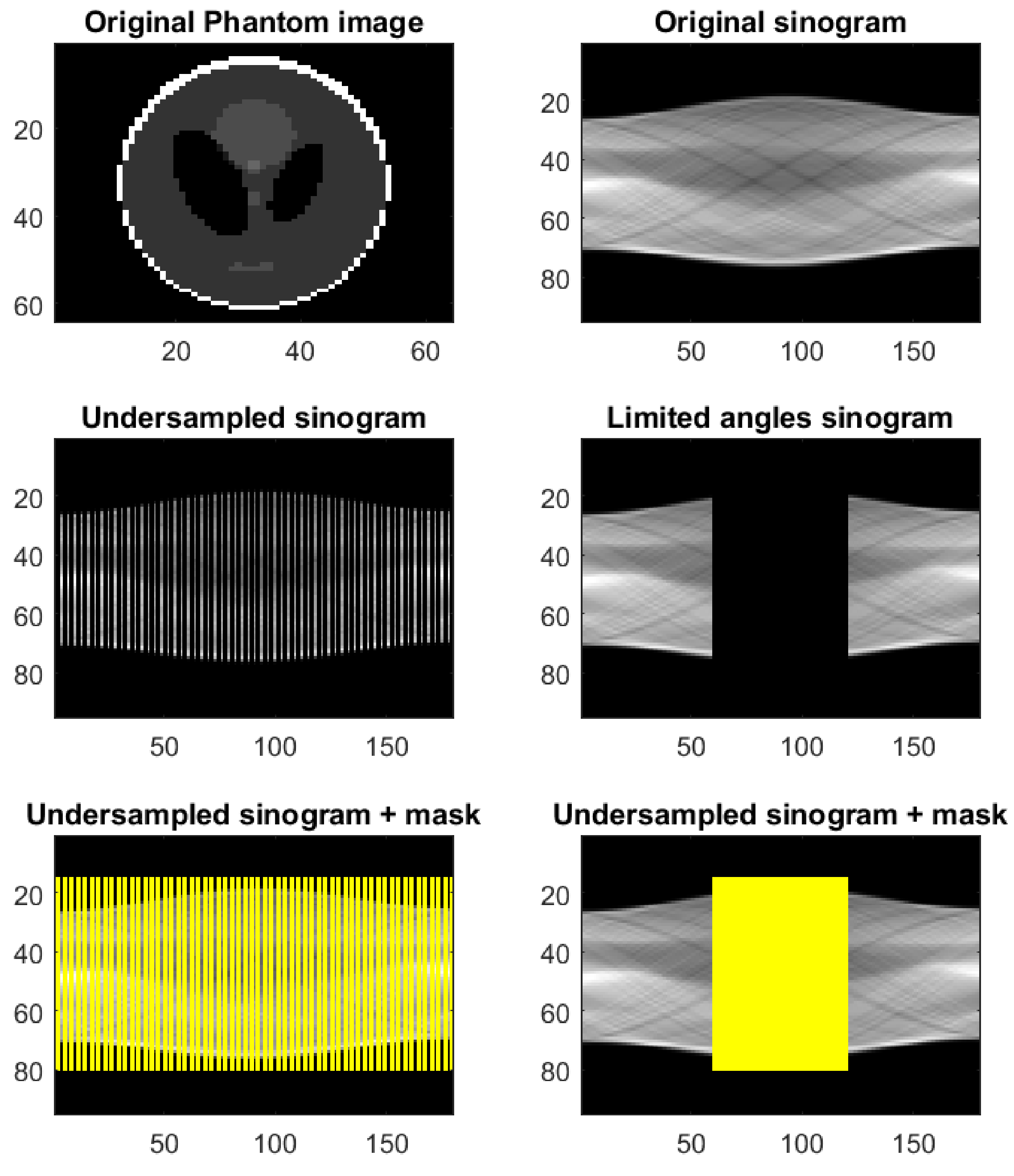
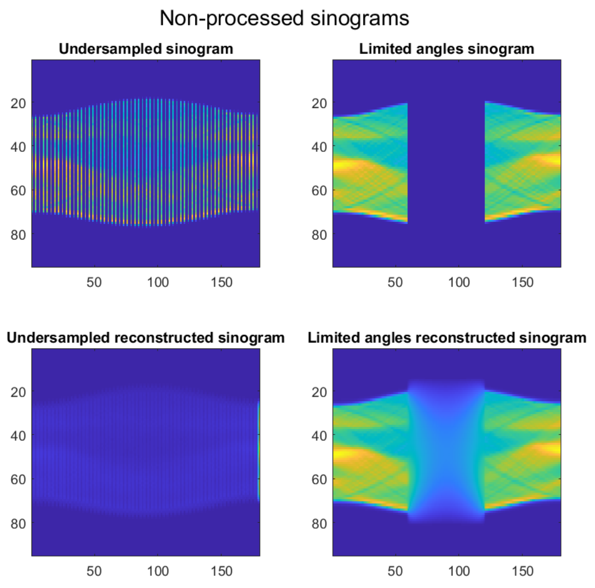
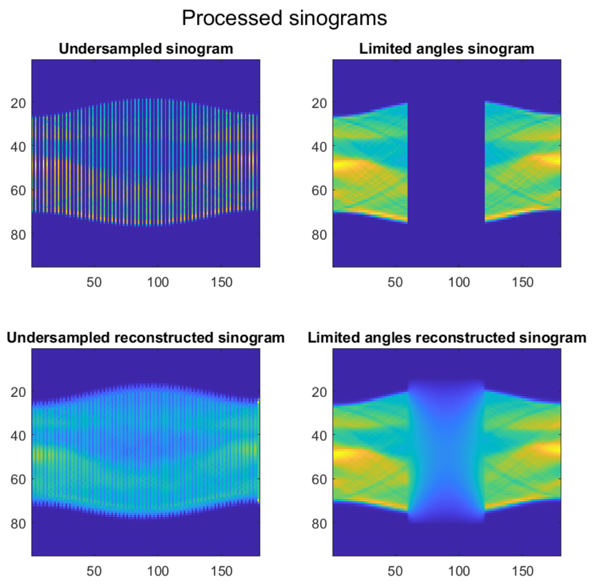


Figure 14: The data we are considering in this part



(a) Data and non-processed outputs



(b) Data and processed outputs

Figure 15: Sinograms obtained by our Laplacian inpainting

We will start by resorting to known inpainting methods, with tools we have already used in the past courseworks and lectures. Let's try the inpainting based on an isotropic Laplacian to fill in the missing data.

The form of the normal equation is the following:

$$(I_\Omega^T I_\Omega - \alpha\nabla^2)\boldsymbol{f} = I_\Omega^T \boldsymbol{g}$$

where $\boldsymbol{f}$ is the estimate, $\boldsymbol{g}$ is the data, α is the regularisation parameter, and I_Ω is the $(n_{pixels} - n_{missingpixels} \times n_{pixels}$ matrix obtained by deleting rows corresponding to missing pixels from the identity matrix. The notations are inspired by those in lecture 11 about inpainting.

Therefore, we implement a Krylov solver based on the equation above, using MATLAB's `pcg` to iteratively compute the estimate, and `del2` to compute the Laplacian of the estimate.

Our results can be found in Figures 15a, 15b and 16.

The first note we want to make is the fact that we need to process the output of our solver, especially in the undersampling case. We can see that the sinogram in the bottom-left corner is low in intensity, except at the very right (actually, the intensity at the right end of the sinogram is so high that the rest of the sinogram appears low-intensity in comparison). We think that it comes from the fact that we have an isotropic diffusion process: a portion of the inpainted part in each undersampled column has diffused each time to the right, to the point where it has now taken a high value. It cannot diffuse out of the bounds of the sinogram, so we end up with a column with much higher values than the others. To rectify this, we set the highest values to a comparatively similar value from the other inpainted parts. We also set the smallest values to zero (diffusion out of the masks). This effect is not seen in the limited angles sinogram because the ROI is in the middle of the image, and that eventually, when diffusion either from the left or from the right, as there is only zero values in the middle, the diffusion will quickly lose in intensity. We will encounter this diffusion intensity problem again in the later parts. Now, looking at the filtered back-projections of the sinograms, we cannot really see any major difference for the limited angles case. Looking closely at it, we can see that the yellow circle around the phantom is closing some gaps. To reconstruct the circle, we need information from missing angles, and that is exactly the case for our reconstructed sinogram. The intensity of the diffused inpainting may not be enough to clearly reconstruct the full circle. There is not that much added noise in our reconstruction. For the undersampled case, we can see much more intensity in the image overall, a thicker yellow circle, but contrary to the case of the limited angles, there is much more noise. This might also be due to diffusion: as the masks are really thin, the inpainting for each column necessarily diffuses on the left and right columns. The ROI is not respected. As a result, some diffusion values are now seen as noise. Still, we have now more information about the intensity and shapes of the final reconstruction. The two vertical lines on each side might be due to the last column we have had to process to get back to a normal intensity.

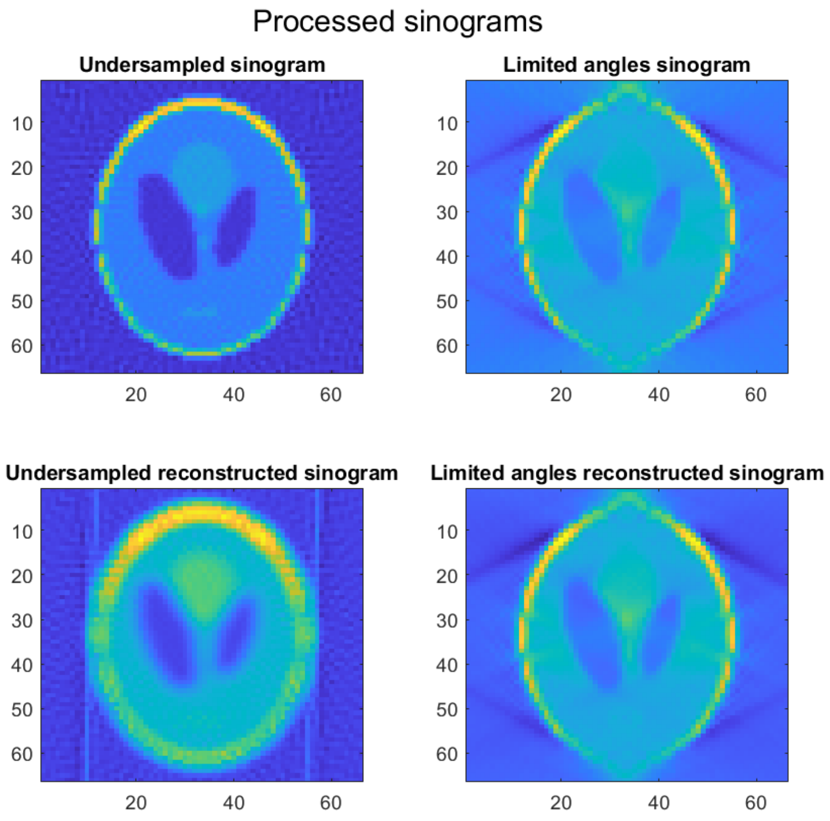


Figure 16: Data and filtered back-projection of the processed sinograms obtained in Figure 15b

To correct the noise added by the diffusion process, we decide to apply denoising techniques directly to the filtered back-projection. We will try out a Krylov solver based on the following equation: $(I^TI - \alpha I)\boldsymbol{f} = I^T\boldsymbol{g}$ (zero-order Tikhonov regularisation where the regularisation matrix is simply the identity. Here we take $\alpha = 2$). We will also try out our Wavelet denoiser from part A (range [1:4] and 80% of thresholding). Let's also take the Radon transform of the denoised image, to see if we have gained any information by performing denoising on our reconstruction. Our comparison can be found in Figure 17.

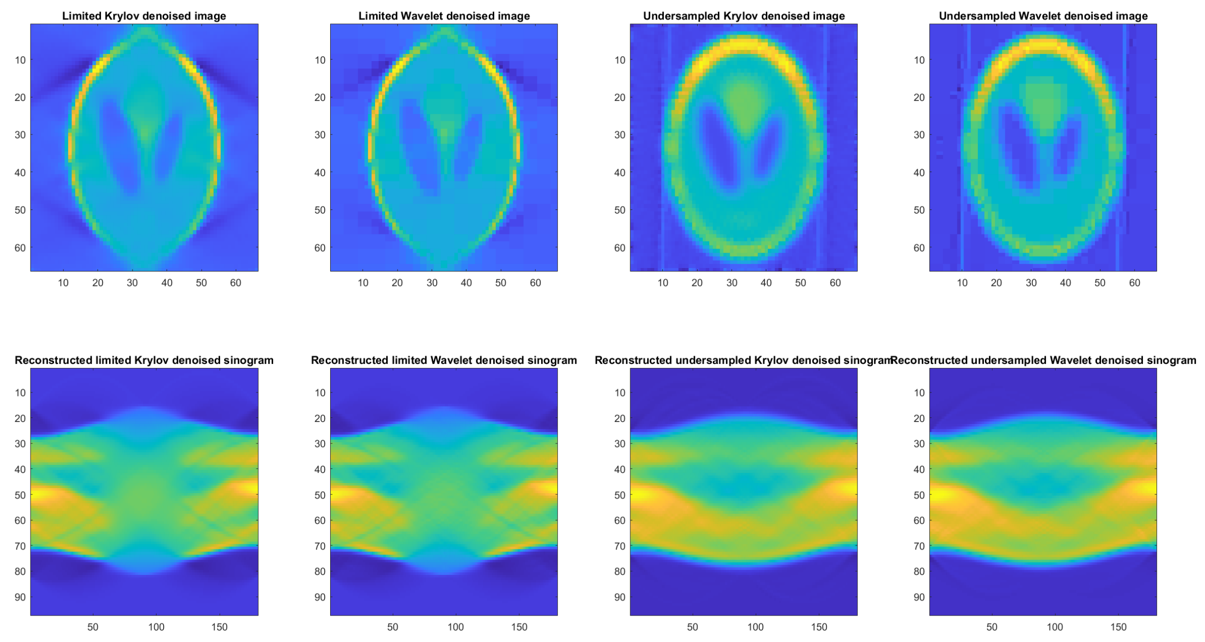

Limited Krylov denoised image
Limited Wavelet denoised image
Undersampled Krylov denoised image
Undersampled Wavelet denoised image
Reconstructed limited Krylov denoised sinogram
Reconstructed limited Wavelet denoised sinogram
Reconstructed undersampled Krylov denoised sinogram
Reconstructed undersampled Wavelet denoised sinogram

Figure 17: Comparison of denoising techniques (Krylov VS Wavelet) in the image space (top row), and the sinogram of each image (bottom row)

The first effect we can see is, as we had observed in part A, that the Wavelet-denoised images are more "squared" than the Krylov-denoised images. For the case of limited angles, this gives a better, more detailed look to the phantom. Conversely, the Wavelet-denoised undersampled image looks pixelated. Probably because the undersampled image is quite blurry to start, whereas the limited angles one exhibits more high-frequency parts. Both methods seem efficient.

This is also confirmed by the reconstructed sinograms. This is particularly telling for the undersampled case: the noise that was spread out between the columns has been smoothed out throughout the whole sinogram. As a result, we can see a full, consistent sinogram. The problem now is that we have lost the fine information: in the limited angles case, we can distinguish small curves inside the sinogram; it is much more blurry.

The conclusion we have from this denoising experiment is that denoising in the image space can effectively be used to rectify the noise spread out in the sinogram by the diffusion process. It helps to complete the reconstructed sinogram. But denoising can only revise noise, it cannot add new information: the finest and high-frequency parts of the sinogram still need to be reconstructed beforehand.

We will thus move on to the state-of-the-art method, which involves a Total Variation (TV) regularisation. In our first experiment, our regularisation was the L2-norm of the Laplacian. For TV, the regularisation is the L1-norm of the gradient. We implement the Gauss-Jacobi iterative scheme for linear systems described in section 6.2 of [1], where each pixel is approximated by the mean of its neighbors, and we enforce the Neumann boundary condition by defining a λ_e constant being null outside of the extended ring surrounding the ROI. Let's note that we have used the "lifted" weights implementation, where we use $\frac{1}{\sqrt{a^2+\|\nabla u_p\|^2}}$ (where a is a small constant) instead of $\frac{1}{\|\nabla u_p\|}$. This is because our ROI contains zero values, which would result in infinity values for the weights. The TV solutions can be seen in Figure 18. We have chosen $\lambda_e = 0.01$ for the undersampled case, $\lambda_e = 1$ for the limited angles case, and $\alpha = 0.001$.

The first remark we can make is that it works really well for the undersampled case.

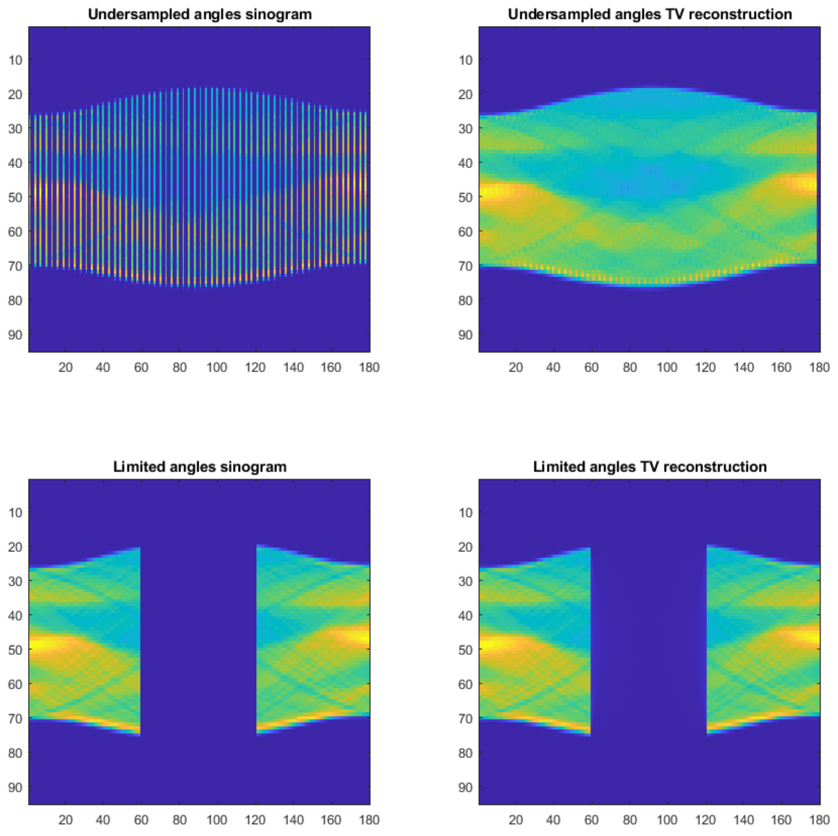
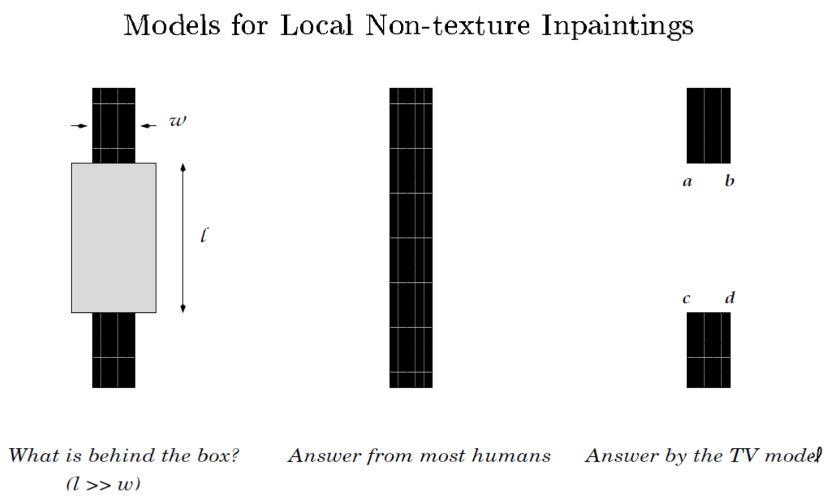


Figure 18: Comparison of the data (first column) and the reconstruction obtained by the TV algorithm from [1] (second column)

Compared to the sinograms in the bottom-right of Figure 17, here TV allows a good reconstruction of the finest details within the sinogram.



Models for Local Non-texture Inpaintings
w
l
a
b
c
d
What is behind the box? (l >> w)
Answer from most humans
Answer by the TV model

Figure 19: When $l \gg w$, the TV and segmentation based inpaintings both act against the Connectivity Principle of human perception. Humans mostly prefer to have the two disjoint parts connected, even when they are far apart [2] [3]

Then, how come it does not work for the limited angles case? The reason is explained in length in [1]: it is due to the Connectivity Principle. When we, as humans, see an occluded part of an object, we'd rather see the two extremities connected in the end. The problem with the TV algorithm is that it does not think like this: numerically, it is more interesting for it to let the space blank than to attempt to minimize a random shape in the ROI. An illustration of this principle is displayed in Figure 19.

We now need to inform the algorithm to reconstruct the missing part following a certain structure; we need to give prior information about it. We will implement the "Directional Total Variation regularisation" idea of [4]. What we did previously was averaging the values of the 4 neighbouring pixels (North, East, South, and West). But, when it comes to the case of limited angles, the further we go inside the ROI, the fewer intensity pixels we can find. Therefore, the inpainted pixels will have less and less intensity. The idea would be to inform the algorithm about the main directions of the image, so that it can average only the important pixels/the pixels that follow the same lines. As a result, the algorithm will favour the pixels restoring the prior structure of the image. Thus, the idea is to put coefficients before the horizontal and the vertical weights, and to adapt them depending on prior information. We will use the gradient of the full sinogram to detect the boundaries we want the reconstruction to follow, so that it does not get lost in areas where there is nothing to be reconstructed. We have thresholded the gradient to only get the most defining lines of the image. Following the suggestions from [4], we will put the vertical coefficient to a very small value near lines, and otherwise set both coefficients to one.

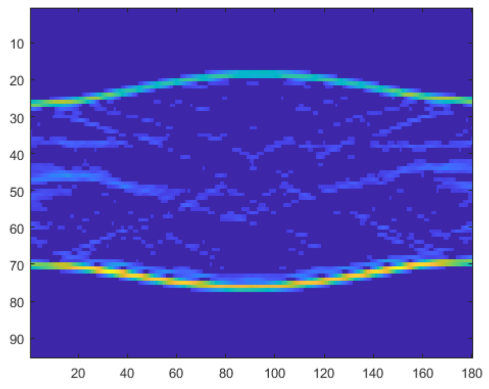


Figure 20: Gradient of the full sinogram

We have plotted the comparison between the traditional and the directional TV methods in Figure 21.

The most notable improvement can be seen for the limited angles case: adding prior geometrical information about the final shape helps in the inpainting process. Still, the recovery is not complete. As discussed above, the gap between the two ends of the sinogram must be too big to allow the TV method to average enough significantly high-intensity pixels. As for the undersampled case, the result is less natural with the DTV. The gaps between the non-missing parts are small and there is always data on the left and on the right to average correctly and to have a good reconstruction. It also makes sense that, when we have small spaces to fill, a diffusion method must be more effective and physically more viable than a constrained method. We can see some "ripples" between each gap, which makes the whole sinogram less smooth. Taking the filtered back-projection, we can see the DTV method has successfully removed the vertical lines which we saw in the diffusion results, as well as reaching a better resolution overall. For the case of limited angles, the recovered information with DTV is still not sufficient to have a better phantom. We would need even more prior information. We could argue that using the gradient from the original phantom is already unrealistic, so could we really add more prior information? Let's note that for the latter case, we have also made

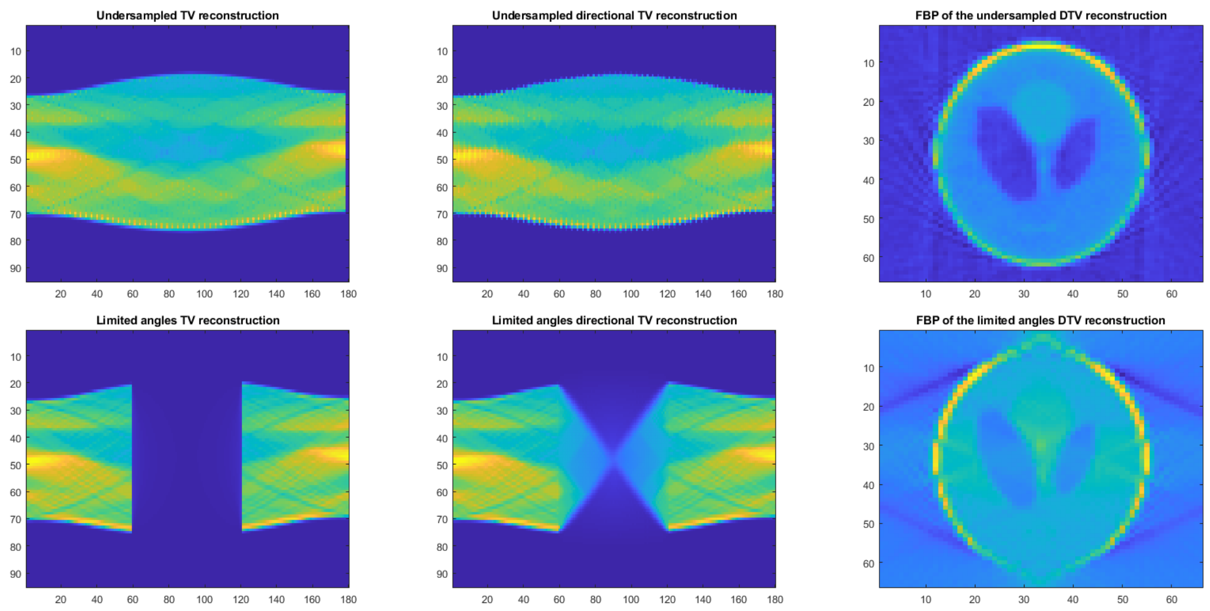


Figure 21: Comparison between the traditional and the directional TV methods for the undersampled and limited angles cases

the size of the boundary vary, but it has not achieved any meaningful result. Rather, it has diluted the information even more. We would need a more comprehensive and more adaptable process to encapsulate all the information from left to right.

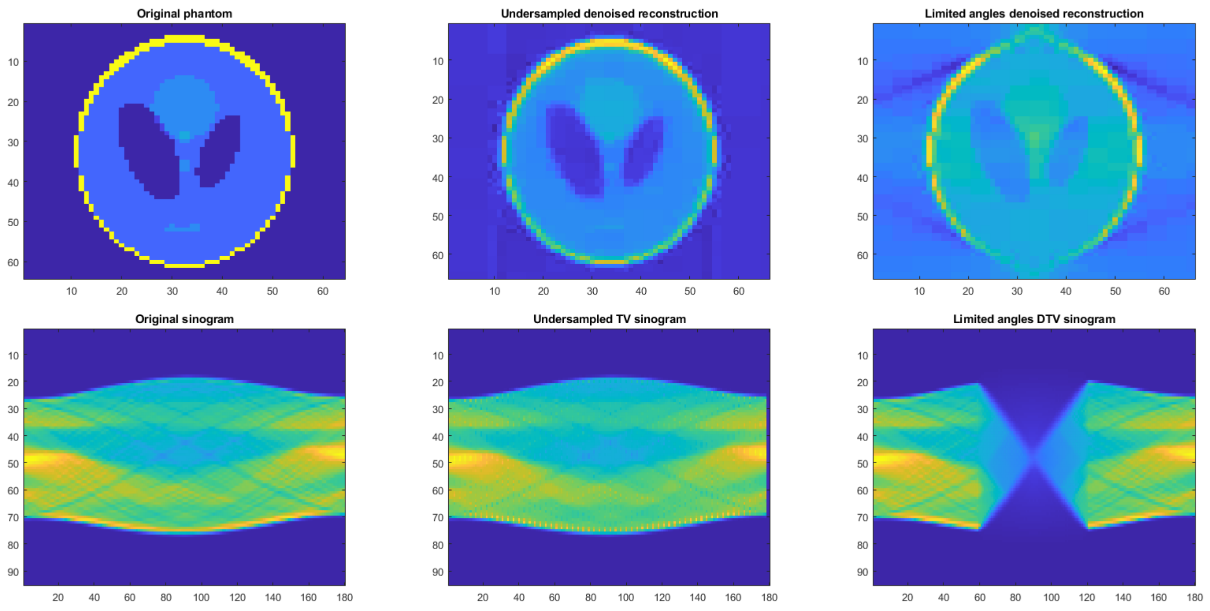


Figure 22: Summary of our results: the original and reconstructed sinograms (bottom row), and the original and denoised phantoms obtained from these sinograms (top row)

To conclude, recent works in inpainting, and especially inpainting in sinogram space, have seen the validation of the Total Variation regularisation as the state-of-the-art reconstruction method. Yet, attention still needs to be paid when designing the scanning experiment in the first place, because we have seen through our study that taking fewer samples during a full scan rotation is a situation we know how to deal with, but omitting a whole range of scanning angles is a way harder task; the latter is not so difficult in terms of computation load but rather in terms of machine understanding/vision principles. This begs the question of initialization and prior information: what are the main attributes of general sinograms? Can we use the already obtained sinograms to enforce conditions on the reconstructions?

The diffusion model is not to be overlooked either, as we have seen that this connectivity problems is less of an issue in this framework.

To sum up, as already suggested in [4], combining current reconstruction techniques with a further emphasis on priors could lead to improvements for inpainting in sinogram space.

References

[1] Tony F. Chan and Jianhong Shen. "Mathematical Models for Local Nontexture Inpaintings". In: *SIAM Journal on Applied Mathematics* 62.3 (2001), pp. 1019–1043. ISSN: 00361399. URL: http://www.jstor.org/stable/3061798 (visited on 04/15/2023).

[2] G. Kanizsa. *Organization in Vision: Essays on Gestalt Perception.* Praeger scientific. Praeger, 1979. ISBN: 9780275903732. URL: https://books.google.co.uk/books?id=iogQAQAAIAAJ.

[3] Mark Nitzberg, David Mumford, and Takahiro Shiota. "Filtering, Segmentation and Depth". In: *Lecture Notes in Computer Science.* 1993.

[4] Robert Tovey et al. "Directional sinogram inpainting for limited angle tomography". In: *Inverse Problems* 35.2 (Jan. 2019), p. 024004. DOI: 10.1088/1361-6420/aaf2fe.