Aluminium foam data reconstruction using CGLS and TV Regularisation - 100 and 200 projection data.

Wadeson, Nicola

2015

MIMS EPrint: **2015.25**

Manchester Institute for Mathematical Sciences
School of Mathematics

The University of Manchester

Reports available from: http://eprints.maths.manchester.ac.uk/
And by contacting: The MIMS Secretary
School of Mathematics
The University of Manchester
Manchester, M13 9PL, UK

ISSN 1749-9097

Aluminium foam data reconstruction using CGLS and TV Regularisation - 100 and 200 projection data.

Nicola Wadeson

1st August 2012

1 Introduction

In x-ray cone-beam CT we have the problem of reconstructing the attenuation coefficient of an object, $f(x)$ for $x \in \mathbb{R}^3$, given a discrete set of M line-integrals through the object,

$$b_i = -log(I_0/I) = \int_{x\cdot\theta=s} f(s\theta + x)\, dx, \tag{1}$$

for some $\theta \in S^2$, $s \in \mathbb{R}$, where $i = 1, \ldots, M$. The most common reconstruction algorithms used for cone-beam x-ray CT systems are variants of the filtered back-projection (FBP) algorithm. These algorithms have specific requirements, in terms of ray sampling, and in some cases data acquisition to suit these requirements can be time consuming.

Another approach is to perform an algebraic reconstruction [8]. Although these algorithms are much slower, require large amounts of computer memory and are less accurate when sampling requirements are met, they are more flexible in terms of data sampling. Advances in computer memory and processing speed over the years means that these methods, once infeasible, are now worth studying, particularly when reconstruction speed is not the top priority.

The Nikon Metris Custom Bay, situated in the Henry Moseley X-ray Imaging Facility at the University of Manchester, is a large walk-in 3D x-ray imaging system capable of scanning a large range of sample sizes. Typical scan times range from 20 to 120 minutes, and resolution ranges from 5 microns (for samples up to 1 cm) to ∼100 microns for the largest samples. The machine geometry consists of a single rotating x-ray source opposite a rotating, flat-panel, 2D detector array consisting of 2000 ×

2000 detector pixels. The scanning speed is limited by the mechanical rotation of the source and detector array, with typically 2000 projections being taken to acquire a full projection set.

This investigation is motivated by a desire to decrease the data acquisition time, thus allowing more data to be collected in a shorter space of time and lending itself to possible 4D reconstruction. This, in turn, should lead to a more desirable machine for customers, resulting in better value for money as well as exciting new possibilities for sample studies. The aim is to reduce the number of projections to as few as possible, thereby reducing the data set, with minimal reduction in image quality.

In these initial investigations the sample data used is of a cylindrical aluminium foam sample, with a 10mm diameter, which was imaged whilst being very slowly compressed over time. In order to apply the compression, the sample was placed inside a perspex cylinder which was not fully contained inside the field of view. Only one revolution of the projection data is considered here for reconstruction image quality comparisons and, due to the slow speed of compression, the sample is assumed to be static. All samples were imaged using a 75kVp polyenergetic x-ray beam from a Tungsten target.

2 Algebraic Reconstruction

To represent the problem algebraically, we imagine the image volume is discretised into N small voxels, x_j for $j = 1, \ldots, N$, each considered to have a constant value $f(x)$. If w_{ij} is the length of intersection of ray i with voxel j, assuming an infinitely thin ray, then an approximation to the integral equation is given by,

$$\sum_j w_{ij} x_j = b_i. \tag{2}$$

If ray i does not intersect voxel j then w_{ij} is zero. Thus the majority of w_{ij}'s are zero, as a particular ray only intersect a few of the voxels. If we consider all of the projection data, b_i, then we have a system of equations, $Ax = b$, where,

$$A = \begin{pmatrix} w_{11} & w_{12} & \cdots & w_{1N} \\ w_{21} & w_{22} & \cdots & w_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ w_{M1} & w_{M2} & \cdots & w_{MN} \end{pmatrix}$$

and A is known as the *projection* matrix. This matrix is of size $M \times N$ where M is

the number of rays and N is the number of voxels. Thus we have M equations in N unknowns. When reconstructing at a high resolution and limiting the number of projections, this matrix will be under-determined.

Since the system is too large to be solved by direct matrix inversion or related methods such as SVD/pseudo-inverse, iterative algorithms are necessary. There are a number of different possible methods, differing in accuracy and stability. Methods created specifically for x-ray CT are ART, SIRT and SART. In terms of established mathematical methods, ART is essentially an implementation of the generalised Kaczmarz method, SIRT is an implementation of Landwebber method and SART is a combination of the two algorithms to give better quality images and faster convergence. These methods have been developed with a view to efficiency rather than accuracy.

Approaching the problem from a mathematical direction there are a number of existing, more complex, algorithms that are suitable for solving this type of problem, with good accuracy and convergence properties, Since we only wish to reduce scanning time and not reconstruction time, we do not consider time to be an obstacle. The conjugate gradient least squares algorithm (CGLS) is commonly used for solving large systems of equations due to its good convergence properties. We begin by applying the CGLS algorithm to reduced projection data sets and then move on to Total Variation Regularisation (TV) which is an iterative de-noising algorithm, currently implemented to perform 5 iterations of CGLS as a starting point.

2.1 Forward and back projection

When we apply the projection matrix A to a vector x we are performing a *forward projection* (FP). That is, we are taking volume data (values provided for the voxels in the discretisation given above) and calculating ray, or projection, data. When we apply A^T to a vector b, we are performing a *back projection* (BP). That is, we are calculating the volume data given the ray data. Depending on the size of the vectors x and b, representing the number of voxels and rays respectively, the projection process can be memory and time consuming. The time taken to compute a reconstruction is largely dependent on the number of forward and back projections required in the reconstruction algorithm.

The projection process is parallelisable and its implementation, used in this study, utilises OpenMP to apply multithreading. All reconstructions were performed on a AMD opteron 2212 processor with a clock rate of 2GHz, with the number of threads set to eight.

3 Conjugate Gradient Least Squares Algorithm

Unlike the FDK algorithm, which approximates the rays to lie in a plane when using cone-beam data and performs reconstructions of 2D slices, CGLS can be used for direct 3D reconstruction. This requires a lot of memory as the whole data set is required at once, unlike FDK where each projection can be processed at a time.

The conjugate gradient method finds the minimum to the quadratic form

$$f(x) = \frac{1}{2}x^T Ax - b^T x + c, \tag{3}$$

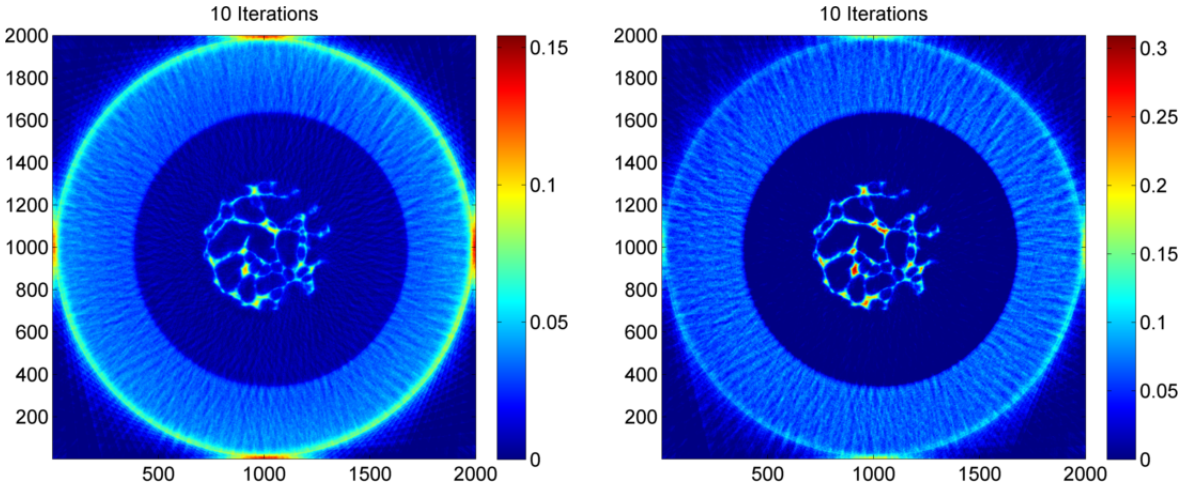
where c is a scalar constant. If the matrix A is symmetric, and positive-definite then this minimized solution is the solution to $Ax = b$. Any matrix, A, can be transformed into a symmetric positive definite matrix by multiplying by A^T. Thus instead we find the solution to $A^T Ax = A^T b$ which is the least squares solution (hence conjugate gradient least squares) which minimises the error.

A good description of the Conjugate gradient method, along with meaningful illustrations is given in [6]. A brief description is as follows: Pick a starting point, x_k, move in a search direction, d_k, by a step of length α_k and calculate the new value x_{k+1}, then repeat for a new search direction. The search directions, d_k, are chosen to be A-orthogonal and are calculated from the M linearly-independent vectors, or residuals, $r_k = A^T(b - Ax_k)$. This is essentially the method of steepest descent with A-orthogonal rather than orthogonal search directions.

In a perfect scenario with no rounding errors, convergence to the least squares solution would be obtain after M iterations (since $A^T A$ has size M). In reality the solution is obtained within some specified error bound, or when a chosen number of iterations have been computed.

The Algorithm

1. $x_0 = 0$
2. $d_0 = r_0 = A^T(b - Ax_k)$
3. $\alpha_k = \dfrac{r_k^T r_k}{d_k^T A^T A d_k} = \dfrac{r_k^T r_k}{(Ad_k)^T Ad_k}$
4. $r_{k+1} = r_k + \alpha_k A^T A d_k$
5. $\beta_{k+1} = \dfrac{r_{k+1}^T r_{k+1}}{r_k^T r_k}, \quad d_{k+1} = r_{k+1} + \beta_{k+1} d_k$



(a) (b)

Figure 1: A 2D CGLS reconstruction of the central slice, using the original 100 projection data with (a) no beam hardening correction and (b) beam hardening correction.

Repeat steps 2 - 5 until desired solution is achieved.

Algorithm Notes

- We choose $x_0 = 0$ such that $d_0 = A^T b$, which is just a backprojection of the data b.
- $A^T A$ is never actually calculated. First, calculate Ad_k (FP) and then apply A^T (BP) in 4. In calculating $d_k^T A^T A d_k$ in 2, since $d_k^T A^T = (Ad_k)^T$ we simply multiply Ad_k^T and Ad_k.
- A total of one initial BP and (one FP + one BP) per iteration is required. That is, 1 + 2*K projections for K iterations. As few as 10 iterations have been shown to give good results.
- As a benchmark for speed, when using 8 threads, the 2D CGLS reconstruction onto 2000 × 2000 voxels using 2000 × 100 rays took approximately 50 seconds.

4 CGLS Results: 100 projection data

A typical data set would consist of 2000 projections and be reconstructed onto a 2000 × 2000 × 2000 voxel grid, giving a voxel size of 30.8 μm. We begin by reducing this 20 times by taking only 100 projections per source revolution. Initially the raw data is reconstructed in 2D, considering the central slice only, using the full 2000 × 100 data set and retaining the reconstruction resolution of 30.8 μm (with 2000 × 2000 voxels).

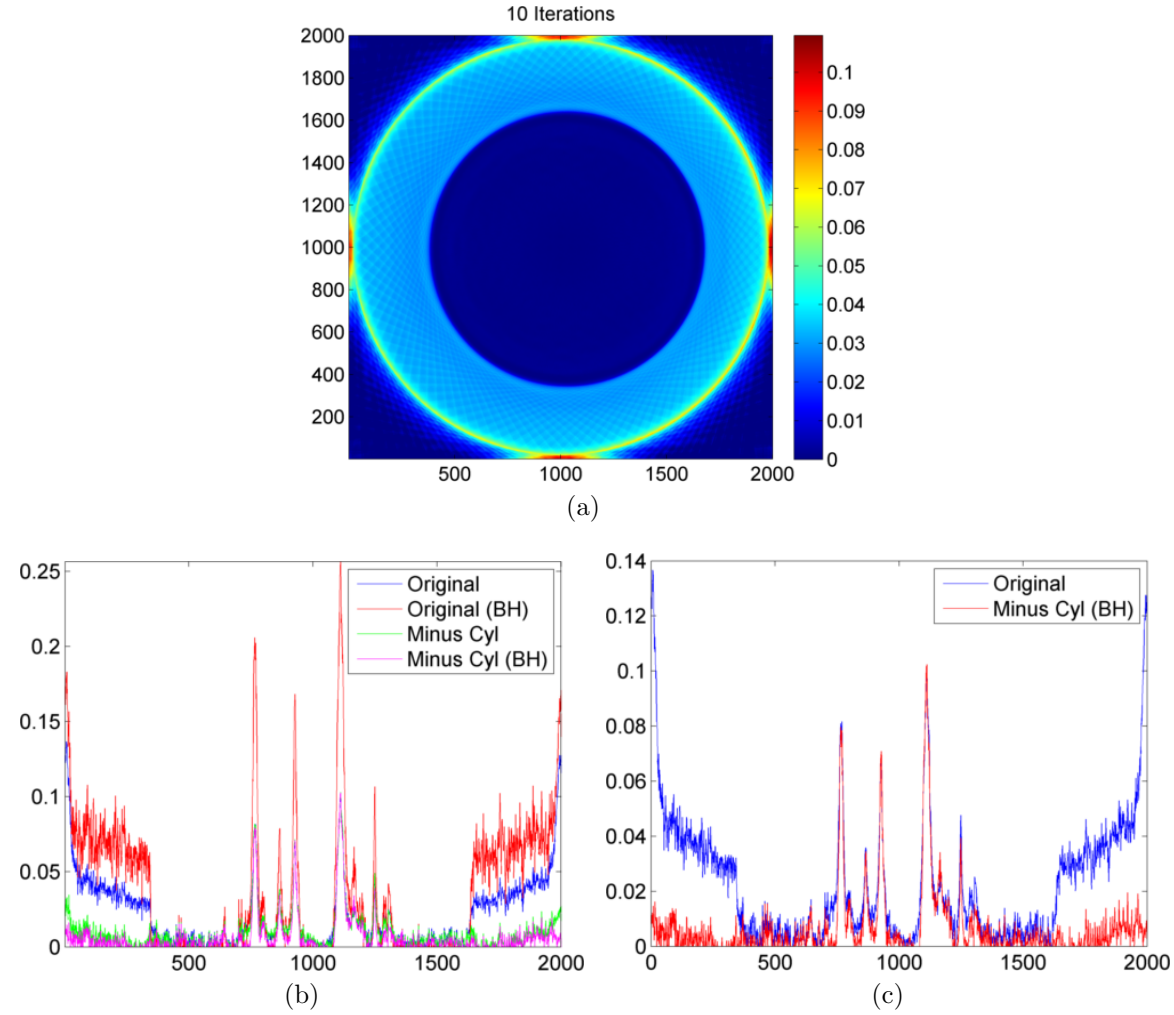


Figure 2: A 2D CGLS reconstruction of the cylinder data is shown in (a), and the central profiles (voxel row 1001) plotted in (b) for reconstructions of the original data; original data with beam hardening correction; data with cylinder subtracted; data with cylinder subtracted plus beam hardening correction. Fig.(c) plots only the profiles from the original data reconstruction and final data reconstructions for clarity.

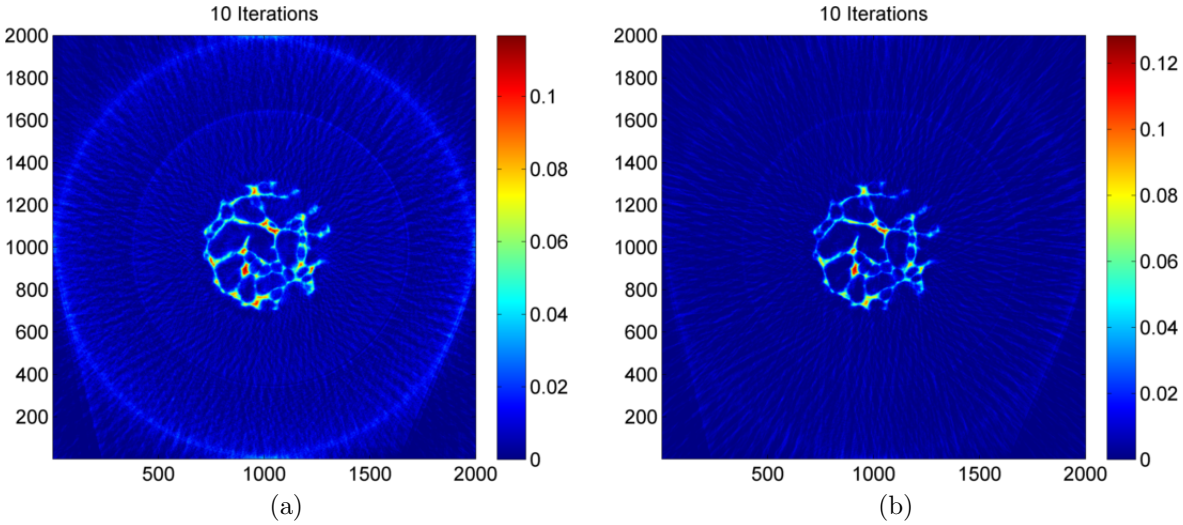


Figure 3: A 2D CGLS reconstruction of the central slice, using (a) the cylinder corrected data and (b) with additional beam hardening correction.

The solution, after 10 iterations, is displayed in Fig. 1a. Evidently this image is not very good as the values at the edge of the outer cylinder are much higher than elsewhere. This is attributed to two problems: Firstly, this is a limited data problem, as the cylinder containing the aluminium sample is not fully contained within the ROI and secondly, the results are subject to the effects of beam hardening.

The next step is to apply a simple b^2 beam hardening correction (i.e. the data is squared before reconstruction) and to reconstruct as before, see Fig. 1b. Significant changes are observed, with some of the outer ring artifacts removed, and an approximate doubling of the reconstructed values for the aluminium foam.

The outer perspex cylinder is approximated analytically to reduce the incomplete data problem. This is achieved by calculating the intersection length, d_c, of each ray, in each projection, with a cylinder with an inner radius of 20mm and an outer radius of 35mm (matching the true dimensions) of infinite length. The attenuation coefficient of the cylinder is calculated heuristically, resulting in a value of $\mu_c = 0.28\text{cm}^{-1}$, which relates to an effective energy of approximately 34.3 keV. Subtracting $\mu_c d_c$ from the original data b gives the final projection data for reconstruction. A 2D reconstruction of the central slice of the cylinder data is displayed in Fig. 2a.

The effect each of the corrections has on the reconstructed data is illustrated in Figs. 2b and 2c. The central profile, relating to voxel row 1001, for reconstructions of the original data, the original data with beam hardening, the data with the cylinder subtracted, and with cylinder removed plus beam hardening correction are plotted for 10 iterations of the CGLS algorithm. The final data that will be used in the reconstructions is that with cylinder removed plus beam hardening correction. Comparing this

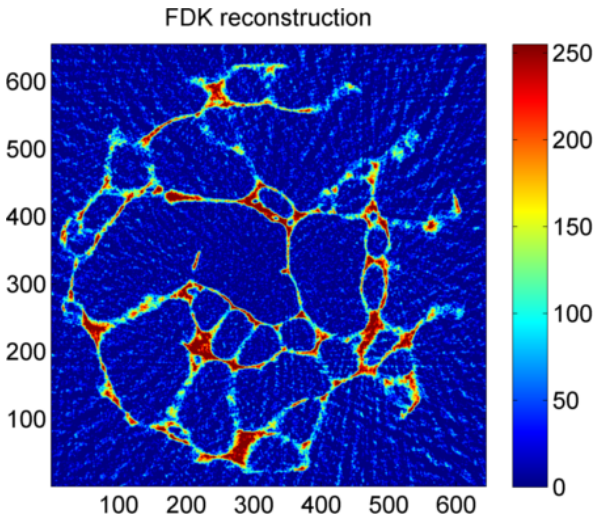


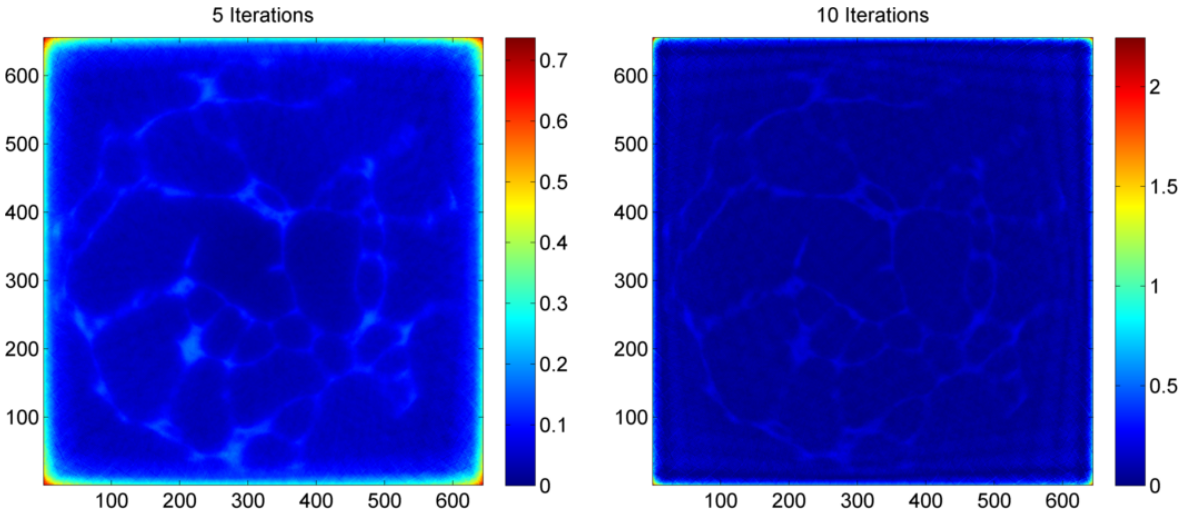
Figure 4: A slice of the 3D FDK sub-section reconstruction.

to the original data reconstruction, small changes are observed to the reconstructed attenuation coefficient of the aluminium foam sample, but removing the high outer values leads to better contrast between aluminium foam and background. This final data reconstructed with and without the beam hardening correction applied is displayed in Figs. 3a and 3b

Reconstructions are now obtained for direct comparison with the inbuilt FDK reconstruction. A slice of the 3D FDK reconstruction is displayed in Fig. 4, where evidently only a small region has been reconstructed. A beam hardening correction has been applied, but it is unknown if any additional filtering has taken place. This reconstruction suffers from a noisy background and it is difficult to discern the object from the background in some places, particularly at the sample edges.

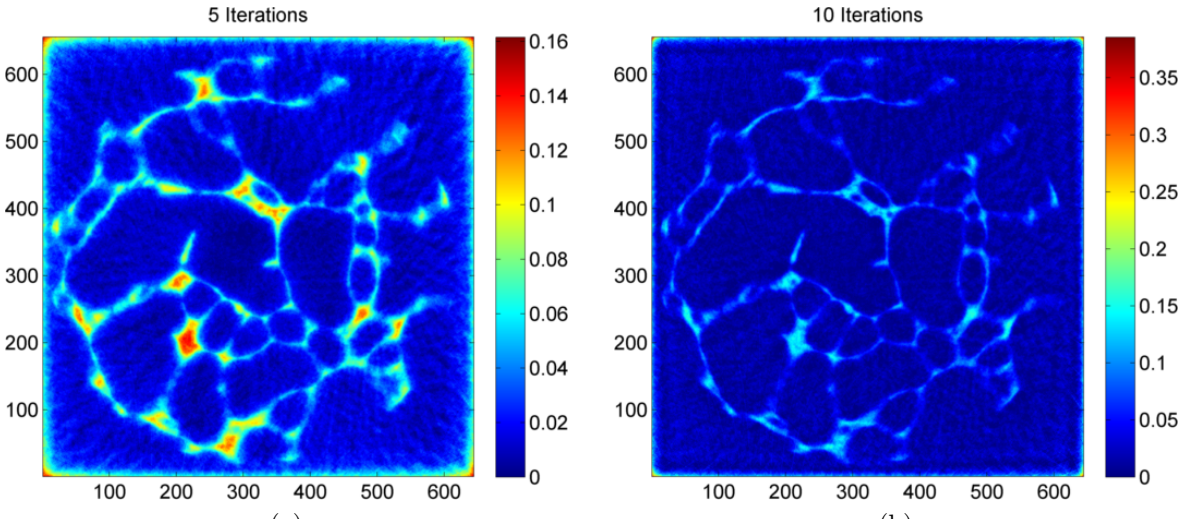
By reading the details of the FDK reconstruction in from file, to determine which rays and voxels to reconstruct and any shift in the data, a reconstruction can be performed using CGLS with the same parameters. A 2D CGLS sub-section reconstruction of the central slice, using the original raw data with 5 and 10 iterations is displayed in Figs. 5a and 5b. Although less voxels are used for the reconstruction the resolution is fixed at 30.8 μm.

Trying to directly reconstruct the data using CGLS onto the same sub-region as the FDK reconstruction is greatly affected by the incomplete data. This is mathematically inconsistent and does not converge to a solution as can be seen from the large jump in reconstructed values from 5 iterations to 10 iterations. The FDK reconstruction doesn't seem to suffer so much from the incomplete data problem, although we are not comparing reconstructed attenuation coefficients here.



(a) (b)

Figure 5: A 2D CGLS sub-section reconstruction of the central slice, using the original data with (a) 5 iterations and (b) 10 iterations.



(a) (b)

Figure 6: A 2D CGLS reconstruction of the central slice, using the data with analytic cylinder correction for (a) 5 iterations and (b) 10 iterations.

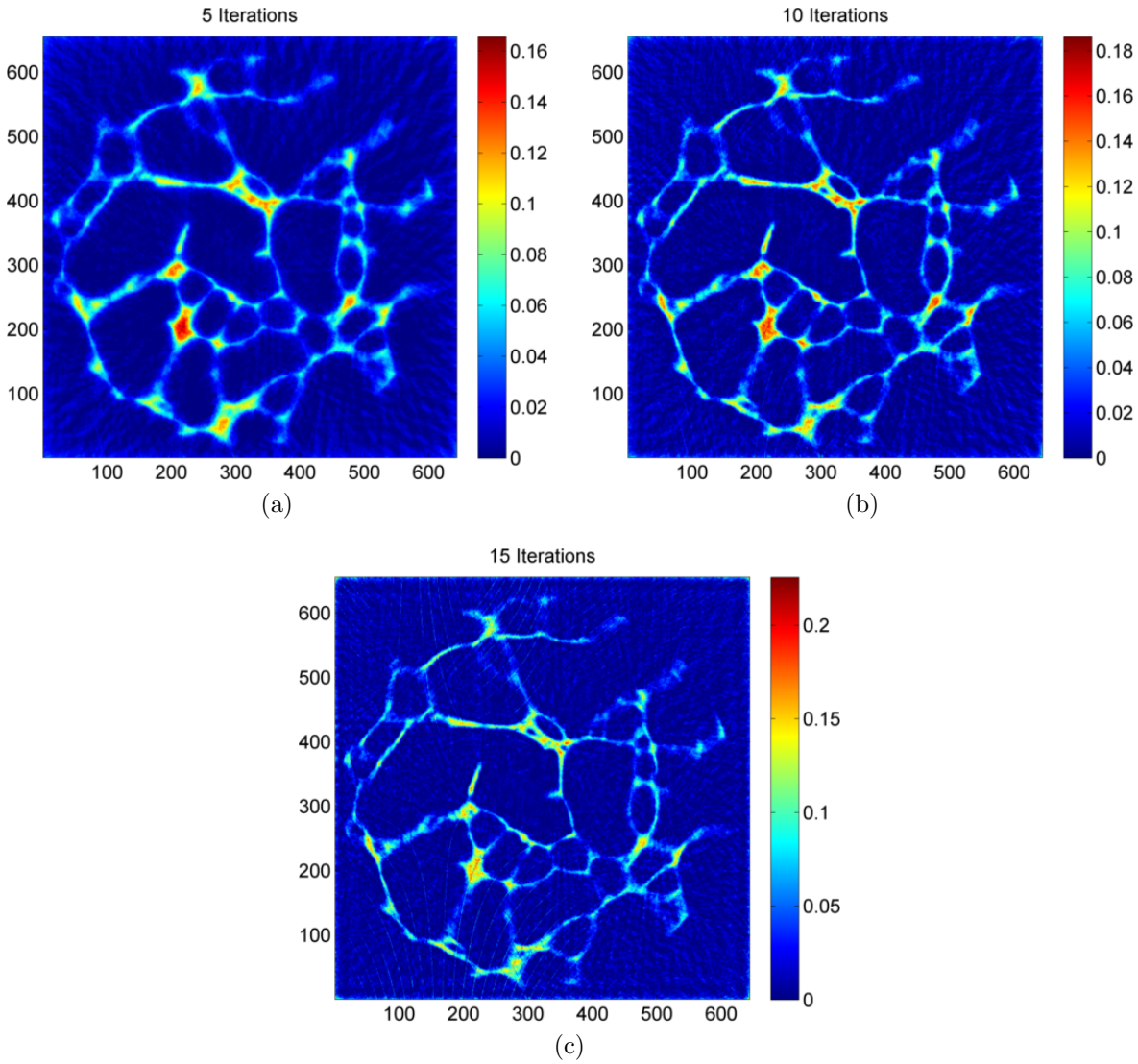


Figure 7: A 2D CGLS reconstruction of the central slice, using the data with analytic cylinder correction and simple beam hardening correction (x^2) for (a) 5 iterations, (b) 10 iterations and (c) 15 iterations.

Therefore, the data with the cylinder correction must be used for the CGLS reconstructions. Without any beam hardening correction the solution appears to be improved after 5 iterations, but quickly diverges as the iterations increase, Fig. 6. Therefore, a beam hardening correction is also necessary (Fig. 7), resulting in a solution that now changes gradually with increasing number of iterations, with 10 iterations providing the best solution.

With the number of iterations set to 10, a 3D CGLS reconstruction is now performed on the same sub-region as the FDK reconstruction, using the corrected data. Three slices of the 3D reconstruction by FDK and CGLS are compared in Fig. 8, with FDK on the left and CGLS on the right. The background in the CGLS images is less noisy which aids the segmentation process, but with as few as 100 projections it is difficult to separate the object from the background for both algorithms. CGLS may show increased performance relative to FDK for a larger sample.

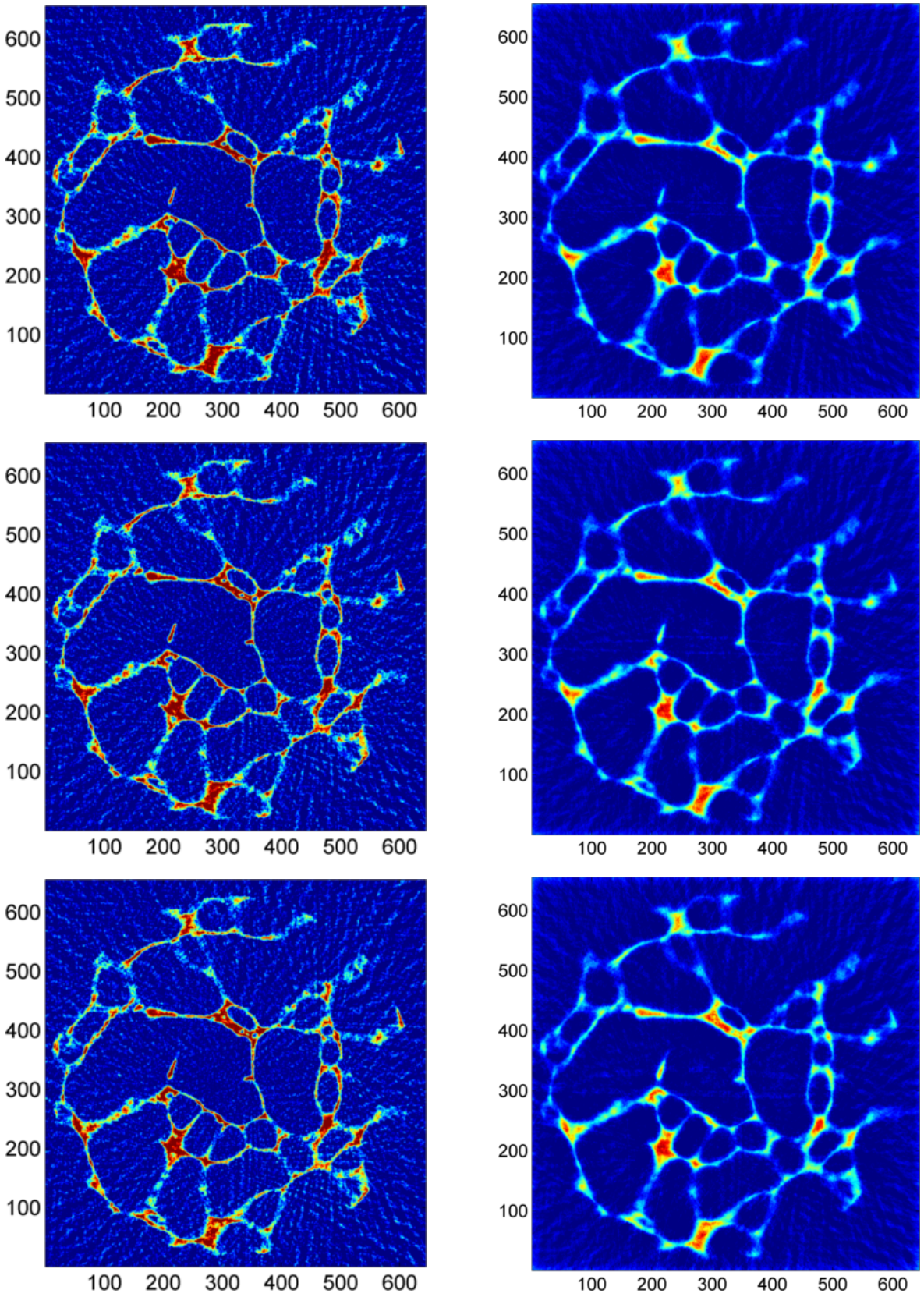


Figure 8: Three different slices of the 3D FDK reconstruction span the left column, with the 3D CGLS reconstruction, after 10 iterations, of the corresponding slices on the right.

5 Total Variation Regularisation

Total Variation (TV) Regularisation is a de-noising process that aims to compute meaningful reconstructions whilst preserving object edges. The Total Variation Regularisation algorithm used in this work is a practical implementation of Nesterovs optimal first-order method [4], implemented in MATLAB by Jensen et al., and described in the paper [2]. This implementation required the matrix A to be pre-calculated and stored, so the code has been adapted to use the forward and back projection algorithms described previously. That is, to calculate Ax and $A^T b$ directly.

The Total Variation (TV) of a function $g(t)$ (proposed by Rudin, [5]), with $t \in \Omega \subset \mathbb{R}^p$, is defined as,

$$T(g(t)) = \int_\Omega \|\nabla g(t)\|_2 \, dt. \tag{4}$$

Since this function is non-differentiable a smoothed version of the TV functional is instead used, with a discrete version given by,

$$T_\tau(x) = \sum_{j=1}^{N} \Phi_\tau(D_j x), \tag{5}$$

where Φ_τ is the Huber function,

$$\Phi_\tau(z) = \begin{cases} \|z\|_2 - \frac{1}{2}\tau, & \text{if } \|z\|_2 \geq \tau, \\ \frac{1}{2}\tau \|z\|_2, & \text{else,} \end{cases} \tag{6}$$

and $D_j x \in \mathbb{R}^3$ is the forward difference approximation to the gradient at $x \in \mathrm{R}^3$. The discrete TV regularisation problem is to find the minimum to the equation,

$$f(x) = \frac{1}{2}\|Ax - b\|_2^2 + \alpha T_\tau(x), \tag{7}$$

in the range [0, 1], where $\alpha > 0$ is the regularisation parameter applied to the TV term, $T_\tau(x)$. Setting $\alpha = 0$, and thus removing the regularisation term, gives the least squares solution. The idea is to stay close to this solution whilst decreasing the total variation, so the higher the value of α, the smaller the total variation and the further away we are from the least squares solution.

5.1 Parameters μ and L

The function $f(x)$ is strongly convex with strong convexity parameter $\mu > 0$ such that,

$$f(x) \geq f(y) + \nabla f(y)^T(x-y) + \frac{1}{2}\mu\|x-y\|_2^2, \quad \forall x, y \in \mathbb{R}^3. \tag{8}$$

The function $f(x)$ has Lipschitz continuous gradient with Lipschitz constant L, such that,

$$f(x) \leq f(y) + \nabla f(y)^T(x-y) + \frac{1}{2}L\|x-y\|_2^2, \quad \forall x, y \in \mathbb{R}^3, \tag{9}$$

which is a smoothness requirement on f. It is clear that $\mu \leq L$. Nesterov's optimal first-order method requires that these two parameters, μ and L, be known in advance. For $f(x)$ as defined in Eqn 8 we have,

$$\mu = \lambda_{min}(A^TA) = \begin{cases} 0, & \text{if rank}(A) < N, \\ \sigma_{min}(A)^2, & \text{else,} \end{cases} \quad \text{and} \quad L = \|A\|_2^2 + \frac{\alpha}{\tau}\|D\|_2^2, \tag{10}$$

where λ_{min} and σ_{min} denote the smallest eigenvalue and the smallest singular value respectively and $\|D\|_2^2 \leq 12$ in the 3D case. Due to the inequalities used in these calculations it is only possible to compute bounds on μ and L.

Since μ and L are not known in practice, the MATLAB implementation estimates μ and L during the iterations (μ_k and L_k). L_k is chosen to satisfy

$$f(x_{k+1}) \leq f(y_k) + \nabla f(y_k)^T(x_{k+1}-y_k) + \frac{1}{2}L_k\|x_{k+1}-y_k\|_2^2, \tag{11}$$

by using backtracking on L_k and

$$\mu_k = \min\{\mu_{k-1}, M(x_k, y_k)\}, \tag{12}$$

where

$$M = \begin{cases} \dfrac{f(x) - f(y) - \nabla f(y)^T(x-y)}{\frac{1}{2}\|x-y\|_2^2}, & \text{if } x \neq y, \\ \infty, & \text{else,} \end{cases} \tag{13}$$

that is, the largest μ_k that satisfies (8) for x_k and y_k. A restart procedure is written in the code as the estimate can be too large. So all that is required now is an initial estimate of $\bar{\mu}$ and $\bar{L}$, where,

$$\bar{\mu} \geq \mu, \quad \bar{L} \leq L \quad \text{and} \quad \mu \leq L, \tag{14}$$

currently implemented as,

$$\bar{L} = (\|A\|_2^2 + 12\frac{\alpha}{\tau})/100, \quad \text{and} \quad \bar{\mu} = \min\{L/50, \|A\|_2^2\}, \tag{15}$$

where $\|A\|_2^2$ is the squared largest singular value of A (replace 12 with 8 for the 2D case). In order to compute this singular value the stored matrix A is required as input. In the new implementation the matrix is not stored and $\bar{\mu}$ and $\bar{L}$ are currently chosen artibrarily, satisfying the requirements in (14) only. A central slice of the data is always reconstructed in 2D initially, to determine the preferred values of α and τ, with the final values of μ and L being output to screen. These final values, corresponding to the chosen values of α and τ, are then input as $\bar{\mu}$ and $\bar{L}$ for the final 3D reconstruction. The tight values of μ and L are not necessarily achieved in either case.

5.2 Functions P_Q and G_L

If we take a projected step of length p_k in the direction of steepest descent, with the additional constraint that the result lies in some set, Q, then we have the gradient projection method,

$$x_{k+1} = P_Q(x_k + p_k \nabla f(x_k)), \quad k = 0, 1, 2, \ldots \tag{16}$$

The function P_Q is the Euclidean projection onto the convex set Q, which is defined here as $Q = \{x \in \mathrm{R}^N \mid 0 \leq x_j \leq 1, \forall j\}$.

For constrained convex problems the *gradient map*, defined by,

$$G_L(x) = L(x - P_Q(x - L^{-1}\nabla f(x))), \tag{17}$$

is a generalisation of the gradient, $\nabla f(x)$, for unconstrained problems. The norm of the gradient map provides a measure of how far we are from the minimum, and is suitable for use in a stopping criteria such that,

$$\|G_L(x)\|_2 \leq \bar{\epsilon}, \tag{18}$$

where $\bar{\epsilon}$ is a user specified tolerance. With all the definitions complete, the algorithm is now given in Fig. 9.

Algorithm Notes

- As described previously, the algorithm now requires initial estimates of μ and

L as well as chosen values of α and τ. Two more initial values are required as input; the initial vector x_0 and the requested accuracy $\bar{\epsilon}$. The initial vector x_0 is currently implemented as 5 iterations of the CGLS algorithm and the default value of $\bar{\epsilon} = 1e^{-6}$ is used in all reconstructions.

- Each calculation of f requires one FP and each calculation of ∇f requires one BP. This results in three initial FP plus one initial BP and an additional a*FP, where a is the number of times the while loop is executed in initial backtracking. Subsequently each iteration requires two FP and two BP plus c_k*FP, where c_k is the number of times the while loop is executed in the backtracking for iteration k. This gives a total of $(4 + a) + (4 + c_k)$*K, where K is the total number of iterations. Since the number of iterations required can be in the hundreds and possibly the thousands, this is much slower that CGLS.

- As a benchmark for speed, when using 8 threads, one iteration of the sub-region TV Reg reconstruction took approximately 0.85 seconds per iteration.

The Algorithm
The algorithm consists of three parts; the main algorithm (UPN - Unknown Parameter Nesterov); the backtracking algorithm (BT) and the restart algorithm (RUPN),

UPN

1. Input: $x_0, \bar{\mu}, \bar{L}, \bar{\epsilon}$
2. $[x_1, L_0] = \mathrm{BT}(x_0, \bar{L})$
3. $y_1 = x_1, \quad \theta_1 = \sqrt{\frac{\bar{\mu}}{L_0}}$
4. $[x_{k+1}, L_k] = \mathrm{BT}(y_k, L_{k-1})$
5. $[\tilde{x}_{k+1}, \tilde{L}_{k+1}] = \mathrm{BT}(x_{k+1}, L_k)$
6. if $\|G_{\tilde{L}_{k+1}}(x_{k+1})\|_2 \leq \bar{\epsilon}$, Return $\tilde{x}_{k+1}$
7. if $\|G_{L_k}(y_k)\|_2 \leq \bar{\epsilon}$ Return x_{k+1}
8. $\mu_k = \min\{\mu_{k-1}, M(x_k, y_k)\}$
9. RUPN
10. $\theta_{k+1} = \frac{1}{2}\Big(-(\theta_k^2+q)+\sqrt{(\theta_k^2+q)^2+4\theta_k^2}\Big)$
11. $\beta_k = \dfrac{\theta_k(1-\theta_k)}{\theta_k^2+\theta_{k+1}}$
12. $y_{k+1} = x_{k+1} + \beta_k(x_{k+1} - x_k)$

Repeat steps 4 - 12 unless 6 or 7 is satisfied.

[x,L] = BT(y,$\hat{L}$))

1. $L = \hat{L}$
2. $x = P_Q(y - L^{-1}\nabla f(y))$
3. while equation 9 is not satisfied
4. $L = \rho_L L$
5. $x = P_Q(y - L^{-1}\nabla f(y))$

RUPN

1. $\gamma_1 = \theta_1(\theta_1 L_1 - \mu_1)/(1-\theta_1)$
2. if $\mu_k \neq 0$ and equation 19 is not satisfied
3. Abort UPN
4. Restart UPN with $x_0 = x_{k+1}$, $\bar{\mu} = \rho_\mu \mu_k$, $\bar{L} = L_k$

$$\frac{1}{2}\tilde{L}_{k+1}^{-1}\|G_{\tilde{L}_{k+1}}(x_{k+1})\|_2^2 \leq \prod_{j=1}^{k}\left(1-\sqrt{\frac{\mu_j}{L_j}}\right)\left(\frac{2}{\mu_k} - \frac{1}{2L_0} + \frac{2\gamma_1}{\mu_k^2}\right)\|G_{L_0}(x_0)\|_2^2 \qquad (19)$$

Figure 9:

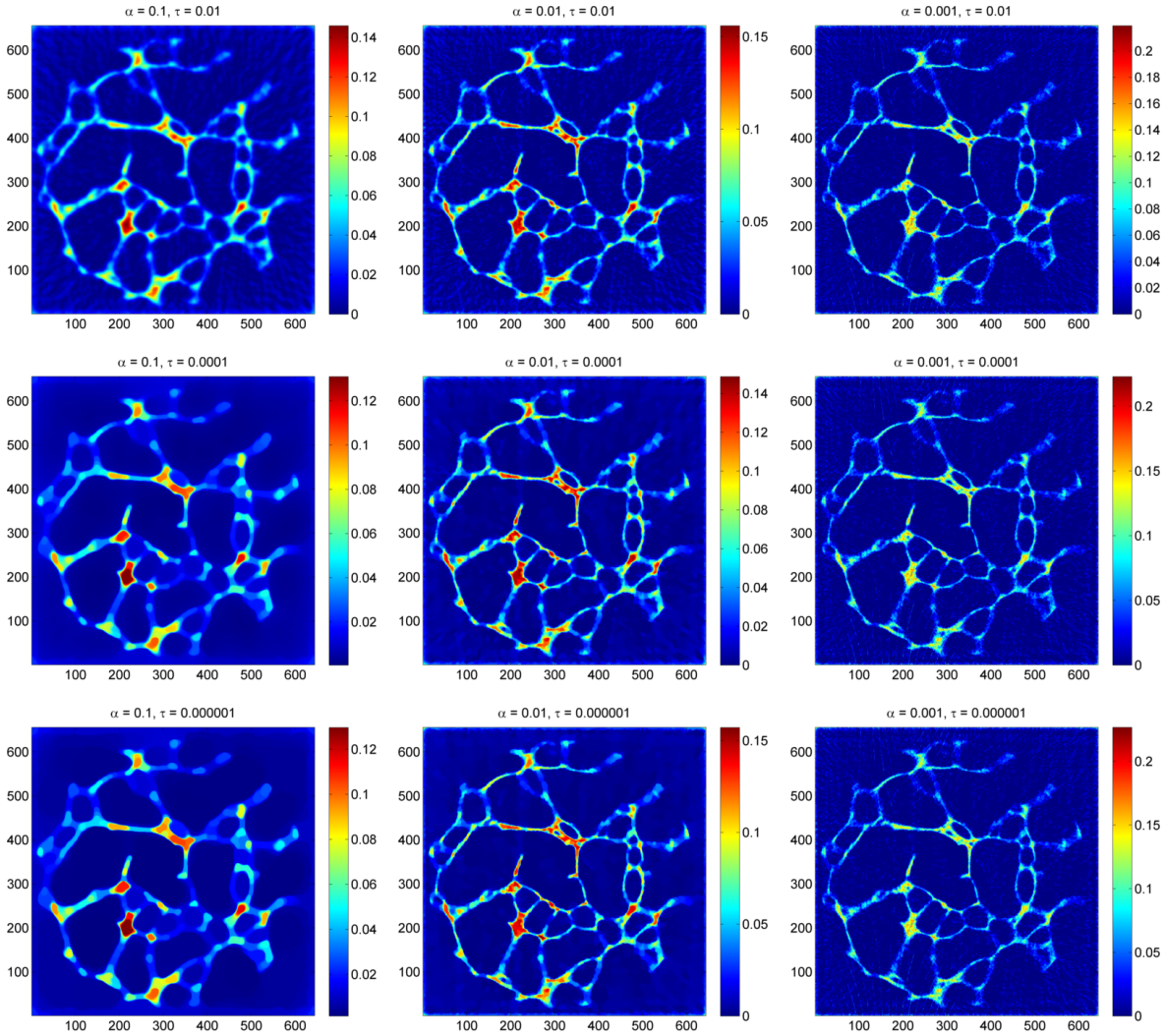


Figure 10: Total Variation Regularisation (TV Reg) 2D reconstructions of the central slice for varying parameters, α and τ.

5.3 TV Reg Results: 100 projection data

Total Variation regularisation reconstructions are performed on the cylinder and beam hardening corrected data. To test initial values of μ and L, as well as varying values of α and τ, we begin with 2D reconstructions of the central slice. Results are obtained for $\alpha = 0.1/0.01/0.001$ and $\tau = 1e^{-2}/1e^{-4}/1e^{-6}$, see Fig. 10. Decreasing α increases the image sharpness, whilst decreasing τ leads to a smoother background with less artifacts.

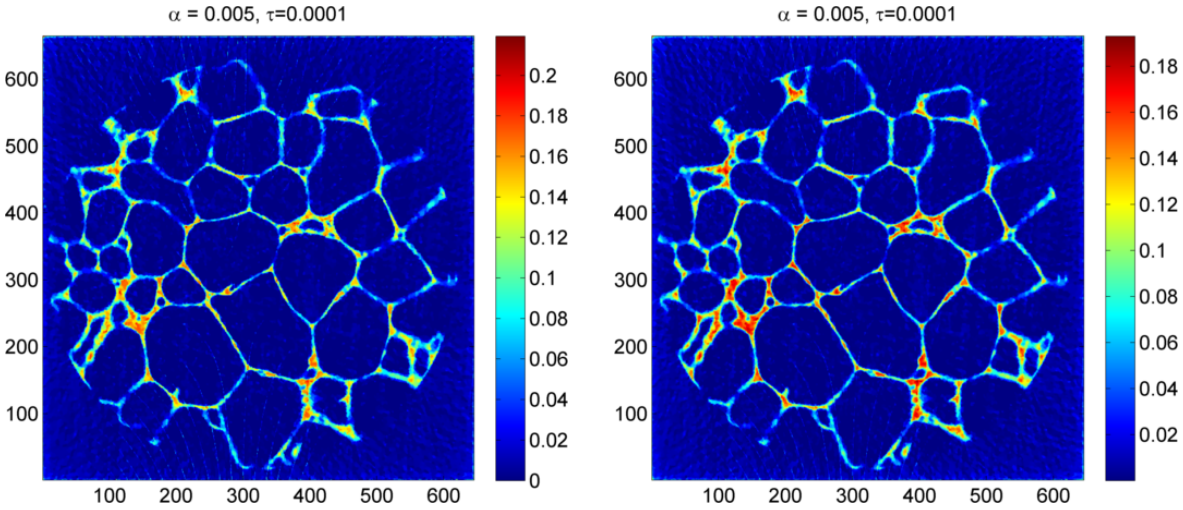
Initial input parameters and corresponding output parameters are displayed in Table 1, in the order that the reconstructions were performed, with K giving the total number of iterations. Values of $\bar{\mu}$ and $\bar{L}$ were chosen to be the output values μ and L from previously attempted reconstructions. As expected, better convergence rates resulted from smaller α and larger τ. Fig. 11 illustrates the different results obtained

Input parameters				Output parameters		
α	τ	$\bar{L}$	$\bar{\mu}$	L	μ	K
0.1	$1e^{-4}$	6930	1.5	6930	1.5	404
0.01	$1e^{-4}$	6930	0.75	6930	0.75	561
0.001	$1e^{-4}$	6930	0.148	6930	0.148	521
0.1	$1e^{-2}$	6930	1.46	6930	1.46	266
0.01	$1e^{-2}$	6930	0.75	6930	0.404	437
0.001	$1e^{-2}$	6930	0.148	6930	0.148	492
0.1	$1e^{-6}$	599000	1.46	779000	0.148	8641
0.01	$1e^{-6}$	779000	0.148	73500	0.148	1429
0.001	$1e^{-6}$	73500	0.148	73500	0.148	1826

Table 1: TV Reg reconstruction input and output parameters relating to images in Fig. 10.

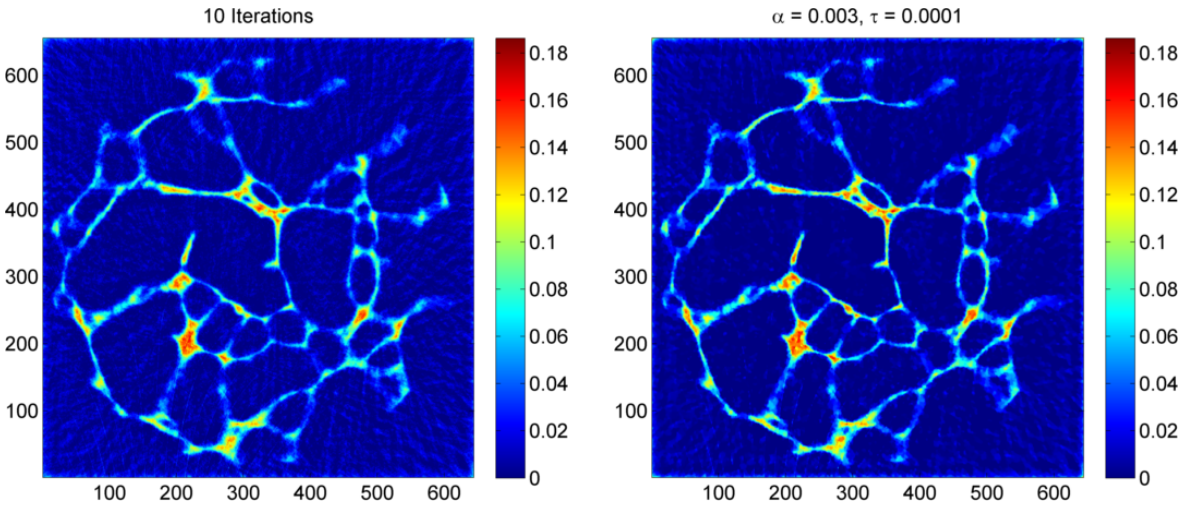
Input parameters				Output parameters		
α	τ	$\bar{L}$	$\bar{\mu}$	L	μ	i
0.005	$1e^{-4}$	6930	0.5	6930	0.5	536
0.0025	$1e^{-4}$	6930	0.5	6930	0.43	562
0.003	$1e^{-4}$	6930	0.5	6930	0.5	552
0.003	$1e^{-5}$	6930	0.5	6930	0.5	575
0.01	$1e^{-6}$	6930	0.5	15200	0.5	771

Table 2: Extra TV Reg reconstruction input and output parameters.



(a) $\bar{L} = L = 1000$, $\bar{\mu} = \mu = 0.01$ (b) $\bar{L} = L = 1000$, $\bar{\mu} = \mu = 0.1$

Figure 11:



(a) CGLS (b) TV Reg

Figure 12: Taking the CGLS solution as reference, the best TV Reg solution in terms of the reconstructed values is given for $\alpha = 0.003$ and $\tau = 0.0001$.

when different starting values of $\bar{L}$ and $\bar{\mu}$ lead to different final values of L and μ for fixed values of α and τ (for 200 projection data).

Since the reconstructed values change with different input parameters, the CGLS solution is used as reference to determine the best TV Reg solution. The values were chosen experimentally, with the best result given by $\alpha = 0.003$ and $\tau = 0.0001$, Fig. 12. The TV Reg solution provides a smoother background than CGLS for these parameters. As it may be of interest for further development of the algorithm, input and output parameters of other 2D TV Reg reconstructions that are not shown here are displayed in Table 2.

These values of $\alpha = 0.003$ and $\tau = 0.0001$ are then used to perform a 3D TV Reg reconstruction with the values of $\bar{\mu}$ and $\bar{L}$ chosen as the output parameters μ and L from the 2D reconstruction. The reconstructed volume size was reduced from 655 $\times$ 644 $\times$ 417 voxels (used in the FDK and CGLS 3D reconstructions) to 655 $\times$ 644 $\times$ 41, due to the long computation time for large volumes. The reconstruction converged after 184 iterations, with $\mu = \bar{\mu}$ and $L = \bar{L}$, and took approximately four hours to compute, using 8 threads. In contrast, the matching CGLS reconstruction onto the same number of voxels took only approximately four minutes.

Three slices of the 3D TV Reg reconstruction are displayed alongside the same slices produced by the FDK reconstruction in Fig. 13. The background is much smoother with less artifacts than the corresponding FDK slices, although the aluminium foam is still difficult to separate from the background at the edges. The same TV Reg slices are displayed alongside CGLS reconstructions in Fig. 14. Again the background does appear less distorted in the TV Reg reconstructions, but the difference is less

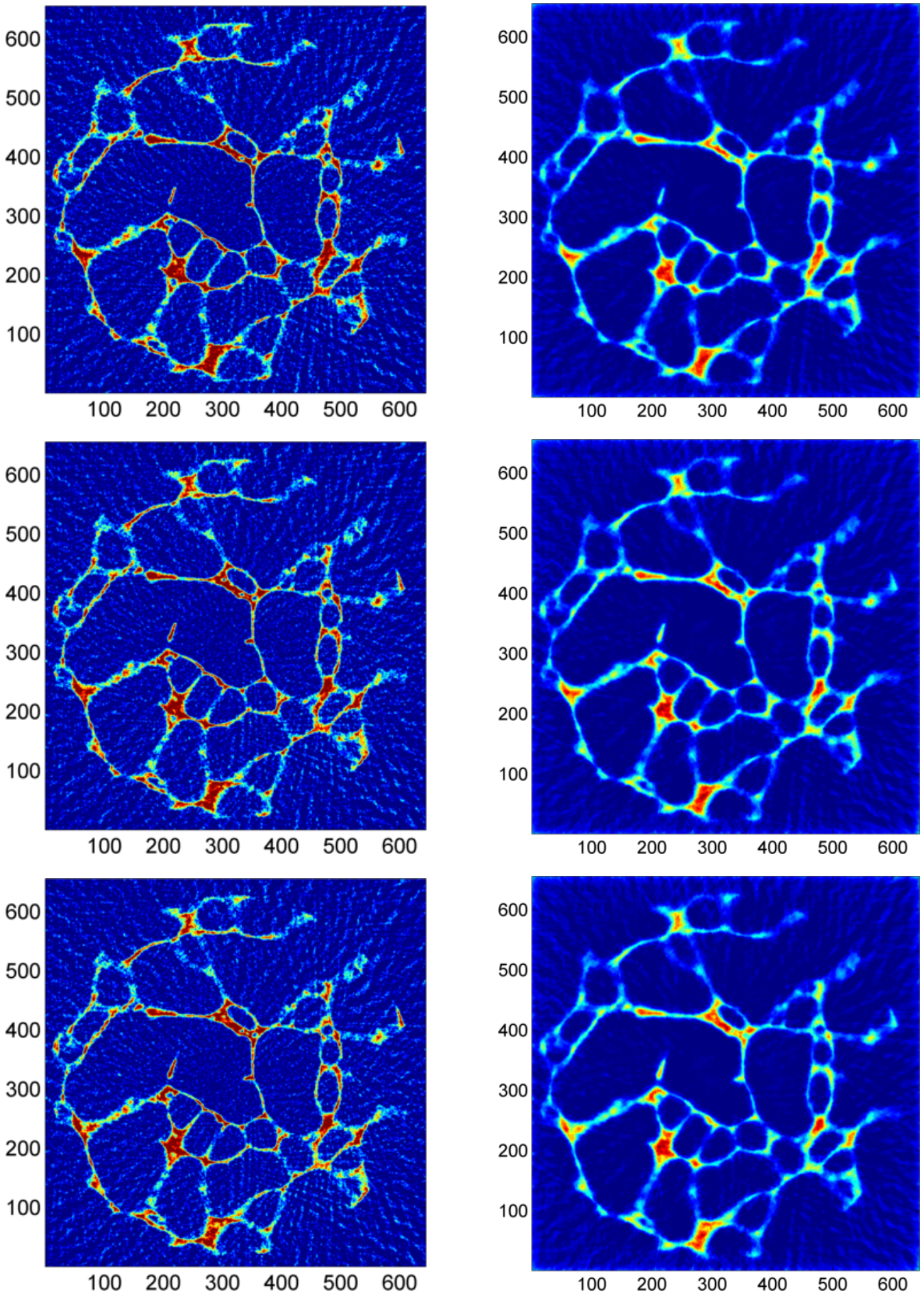


Figure 13: Three different slices of the 3D FDK reconstruction span the left column, with the 3D TV Reg reconstruction of the corresponding slices on the right ($\alpha = 0.003, \tau = 0.0001$).

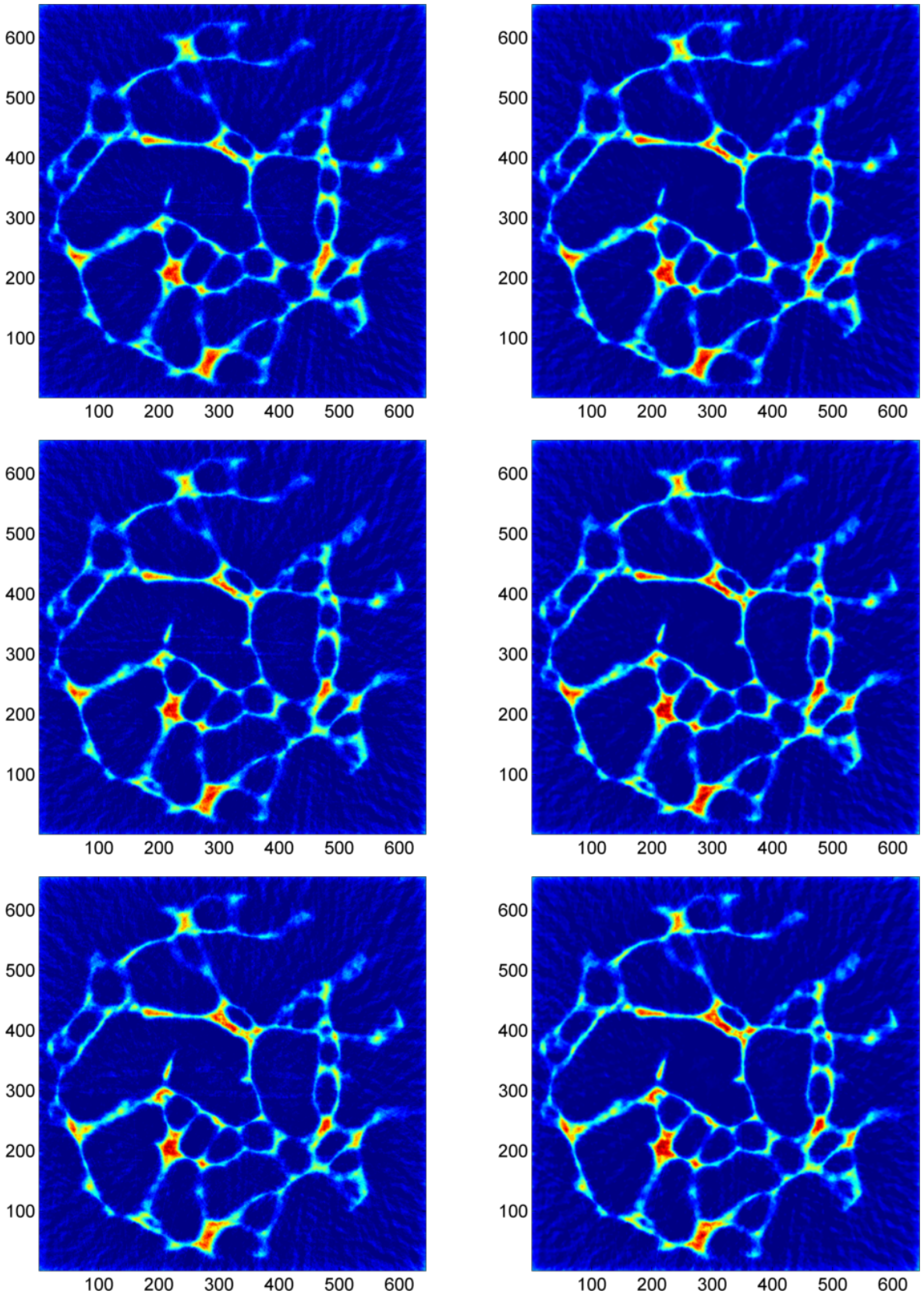


Figure 14: Three different slices of the 3D CGLS reconstruction (10 iterations) span the left column, with the 3D TV Reg reconstruction of the corresponding slices on the right ($\alpha = 0.003, \tau = 0.0001$).

pronounced as with the FDK.

5.4 Conclusions and Further work

The initial results show that both CGLS and TV Regularisation reconstructions show some improvement in the reconstruction of the aluminium foam sample in terms of background artifacts. Further quantitative analysis is needed, and more data sets must be reconstructed, before any firm conclusions can be drawn.

A further study of inital values $\bar{L}$ and $\bar{\mu}$ is required for TV Regularisation, to determine if the tight values are being found, and what effect these values have on the number of iterations required for convergence, along with the parameters μ and τ. If backtracking could be removed this would significantly reduce the number of time consuming forward and back projections required.

Code efficiency has not been investigated here either, so reconstruction speed is not optimised. An investigation into whether it is possible to speed up the TV Reg reconstruction process is imperative. The reconstruction times given in this report are only approximated and will depend on how many other processes were running at the same time. The machine used for reconstruction is a few years old and there have been significant increases in processing power and memory since then.

It may be the case that the quality of the FDK, CGLS and TV Reg reconstructions differs for different data sets, or that one is desirable over the others in different scenarios. Thus, it would, in the long term, be worthwhile offering the user the option of determining which reconstruction is best suited to their needs by reconstructing a 2D slice of their data using each of the algorithms. For TV Reg, multiple 2D reconstructions would be required to determine the most suitable input parameters. After their choice is made, a 3D reconstruction would be performed.

References

[1] F. Jacobs, E. Sundermann, B. De Sutter, M. Christiaens, and I. Lemahieu. A fast algorithm to calculate the exact radiological path through a pixel or voxel space. *Journal of computing and information technology*, 6(1):89–94, 1998.

[2] T.L. Jensen, J.H. Jørgensen, P.C. Hansen, and S.H. Jensen. Implementation of an optimal first-order method for strongly convex total variation regularization. *BIT Numerical Mathematics*, pages 1–28, 2011.

[3] T. Liu. Direct central ray determination in computed microtomography. *Optical Engineering*, 48:046501, 2009.

[4] Y. Nesterov and I.U.E. Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer, 2004.

[5] L.I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1-4):259–268, 1992.

[6] J.R. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain, 1994.

[7] R.L. Siddon. Fast calculation of the exact radiological path for a three-dimensional ct array. *Medical Physics*, 12:252, 1985.

[8] M. Slaney and A. Kak. Principles of computerized tomographic imaging. *SIAM, Philadelphia*, 1988.

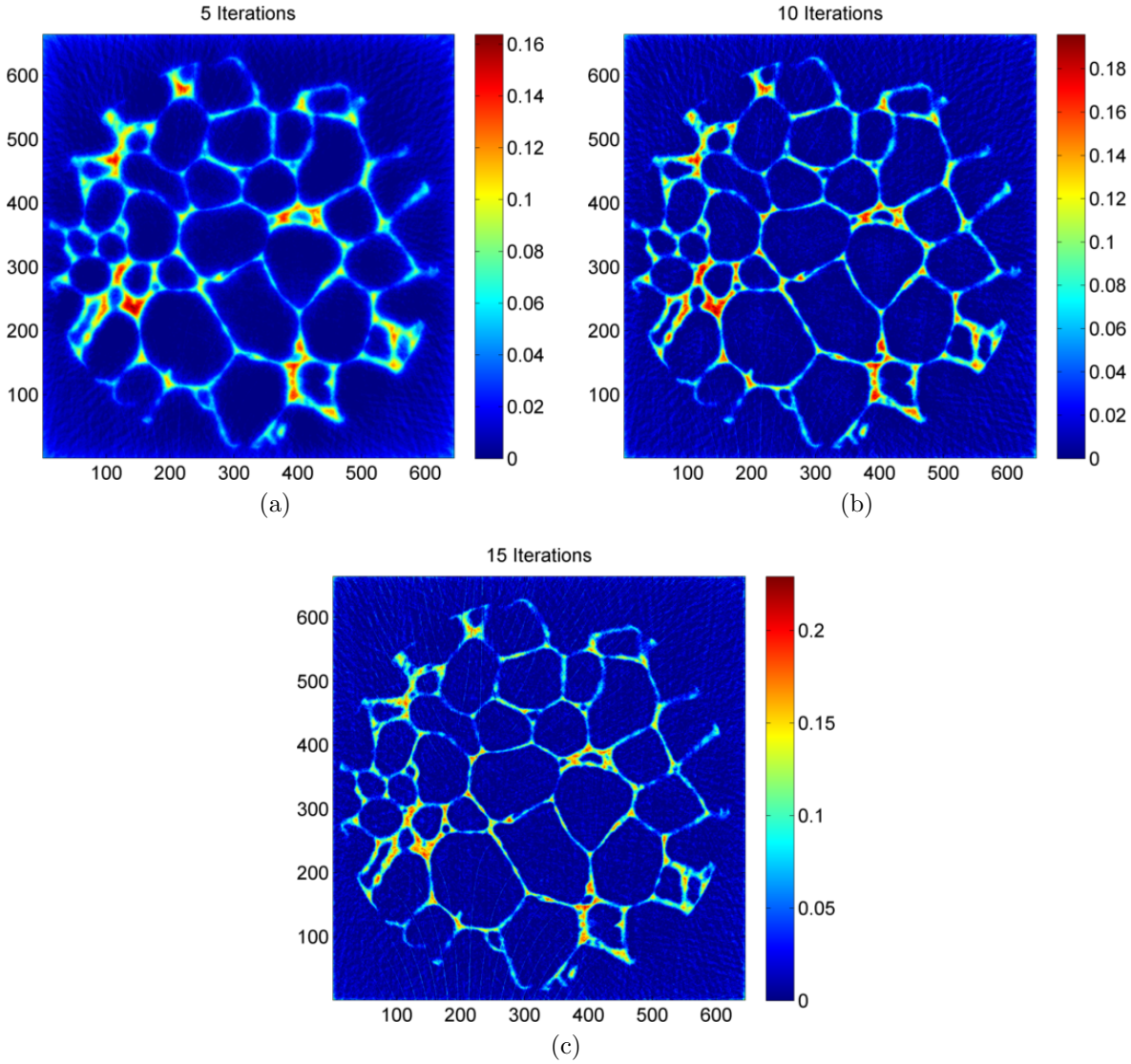


Figure 15: A 2D CGLS reconstruction of the central slice, using the 200 projection data with analytic cylinder correction and simple beam hardening correction (x^2) for (a) 5 iterations, (b) 10 iterations and (c) 15 iterations. The best solution is given after 10 iterations.

A CGLS and TV Reg Results: 200 projection data

The same reconstructions are also applied to 200 projection data and displayed in Figs. 15 - 20. Input and output parameters for all 200 projection reconstructions, including those displayed here, are given in Table. 3

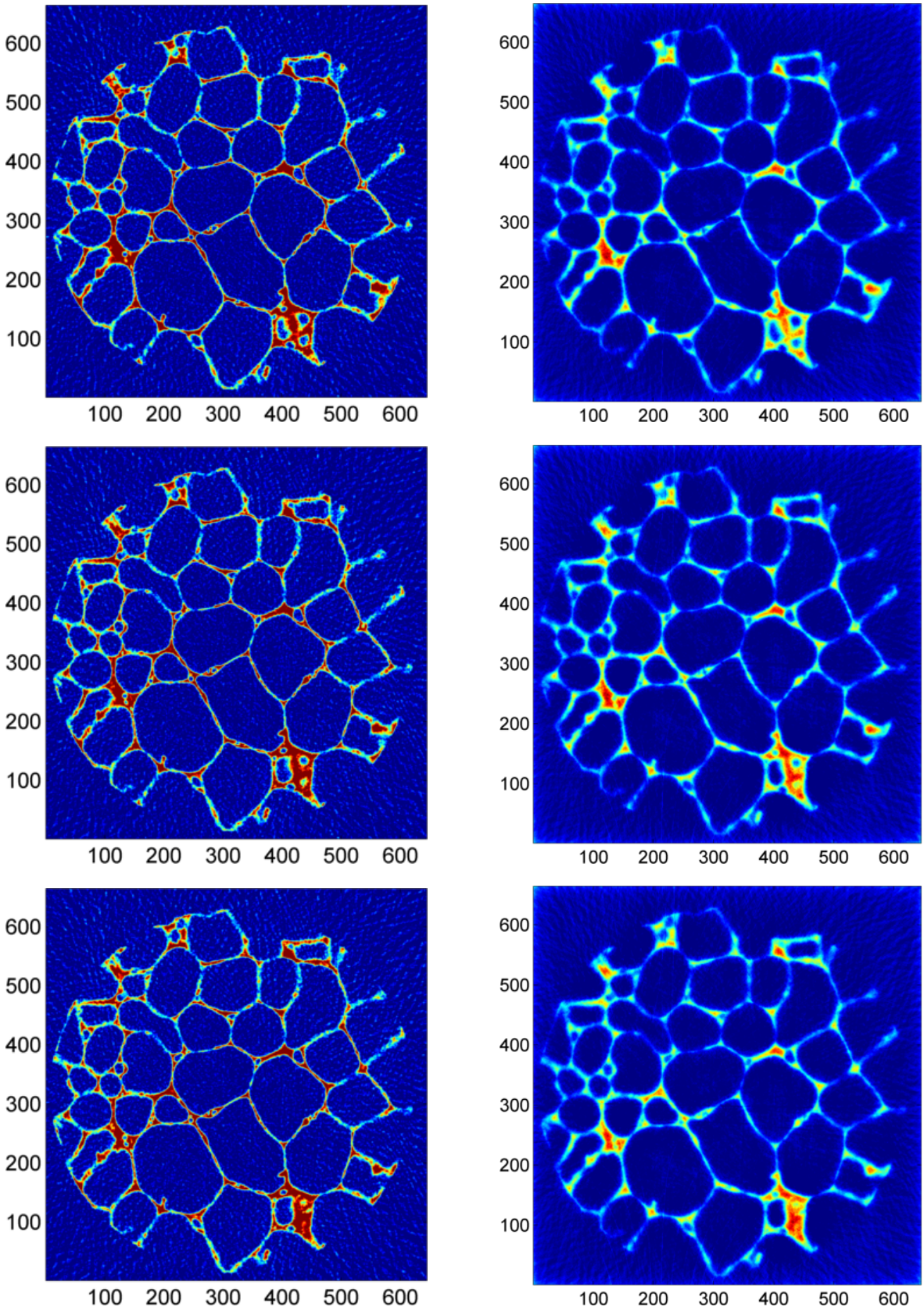


Figure 16: Three different slices of the 3D FDK reconstruction span the left column, with the 3D CGLS reconstruction of the corresponding slices on the right.

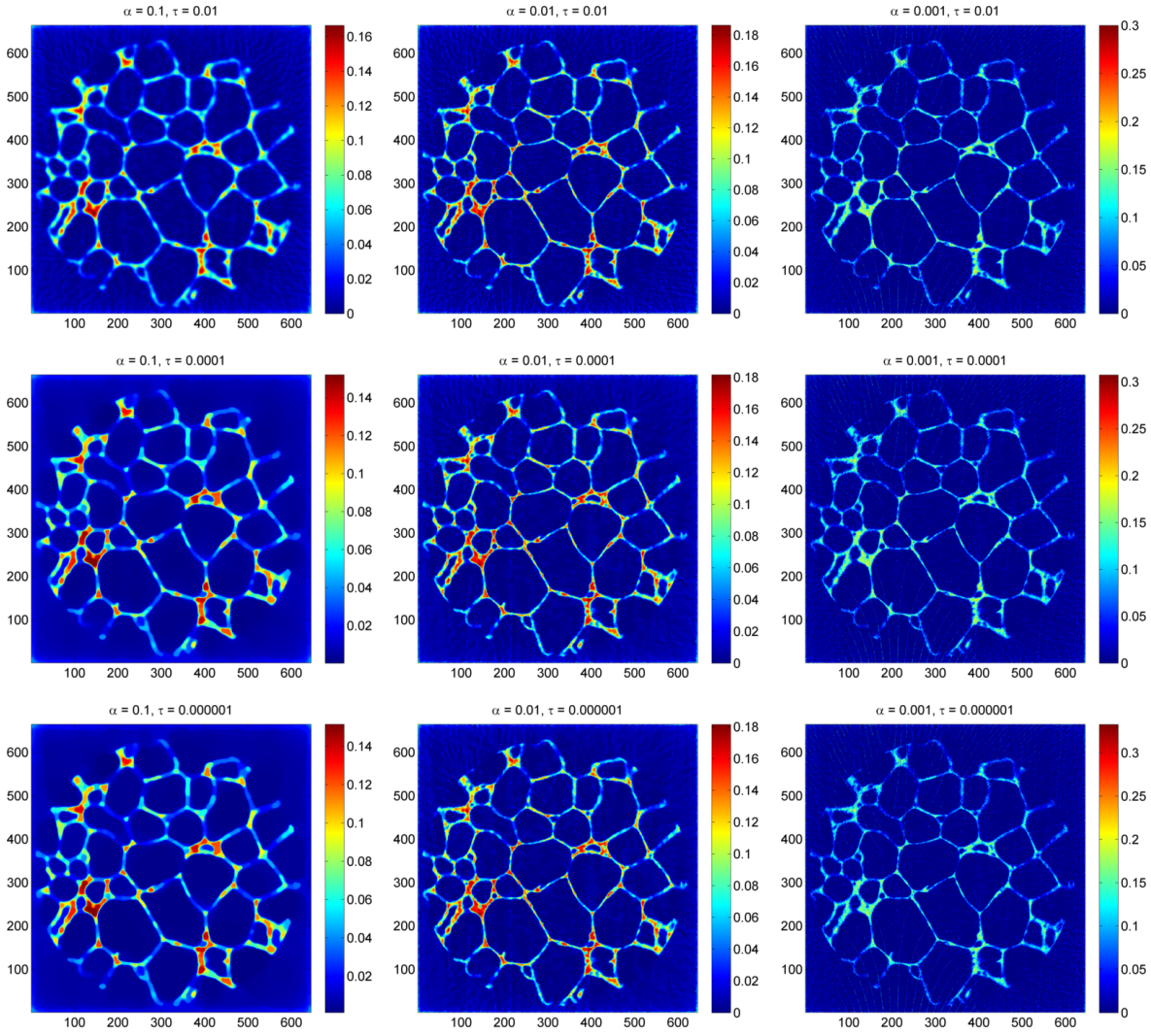
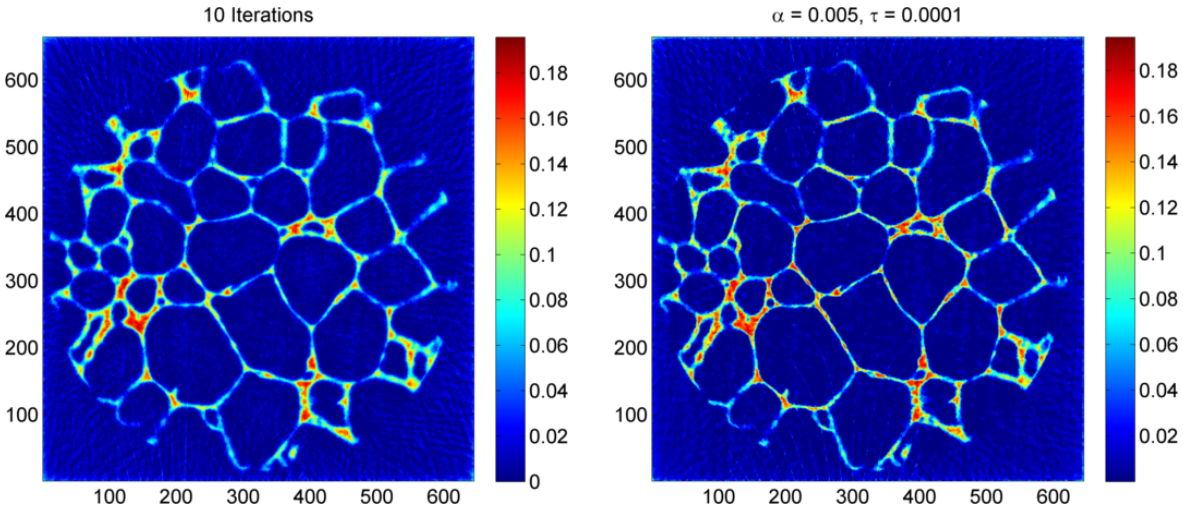


Figure 17: Total Variation Regularisation (TV Reg) 2D reconstructions of the central slice for varying parameters, α and τ. Decreasing α increases the image sharpness, whilst decreasing τ leads to a smoother background with less artifacts. The cylinder and beam-hardening corrected 200 projection data is used here.

Input parameters				Output parameters		
α	τ	$\bar{L}$	$\bar{\mu}$	L	μ	i
0.1	$1e^{-4}$	6930	2.08	6930	1.46	485
0.01	$1e^{-4}$	6930	1.46	6930	0.75	682
0.001	$1e^{-4}$	6930	0.75	6930	0.148	870
0.1	$1e^{-2}$	6930	1.46	6930	1.46	284
0.01	$1e^{-2}$	6930	0.75	6930	0.496	459
0.001	$1e^{-2}$	6930	0.5	6930	0.112	774
0.1	$1e^{-6}$	6930	1.46	599000	1.46	3687
0.01	$1e^{-6}$	59900	0.75	59900	0.75	1858
0.001	$1e^{-6}$	59900	0.148	59900	0.148	1991
0.003	$1e^{-4}$	6930	0.5	6930	0.5	237
0.0025	$1e^{-4}$	6930	0.5	6930	0.5	689
0.006	$1e^{-4}$	6930	0.5	6930	0.5	546
0.005	$1e^{-4}$	6930	0.5	6930	0.5	559

Table 3: TV Reg reconstruction input and output parameters relating to images in Fig. 10.



(a) CGLS (b) TV Reg

Figure 18: Taking the CGLS solution as reference, the best TV Reg solution in terms of the reconstructed values is given for $\alpha = 0.005$ and $\tau = 0.0001$.

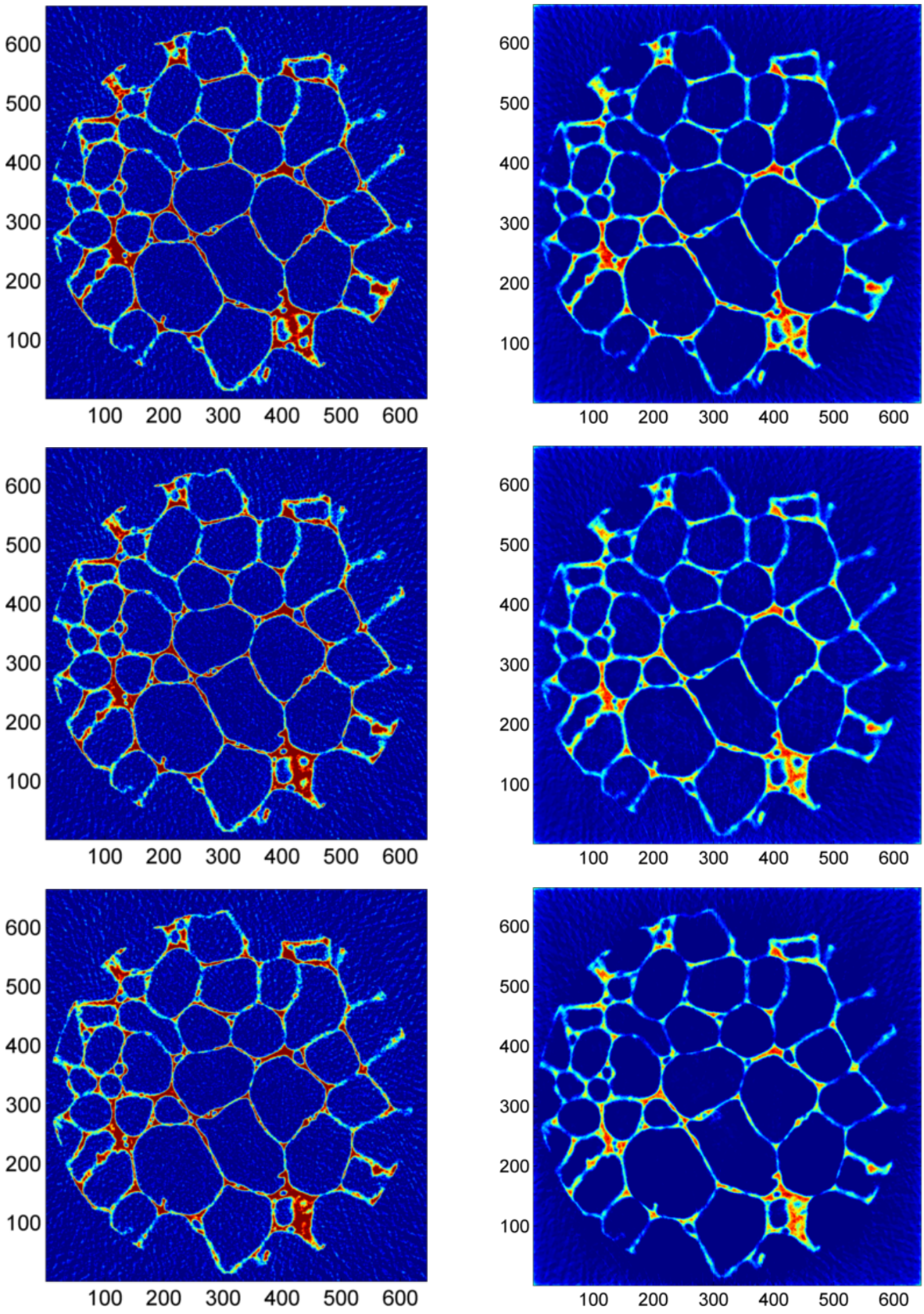


Figure 19: Three different slices of the 3D FDK reconstruction span the left column, with the 3D TV Reg reconstruction of the corresponding slices on the right ($\alpha = 0.005, \tau = 0.0001$).

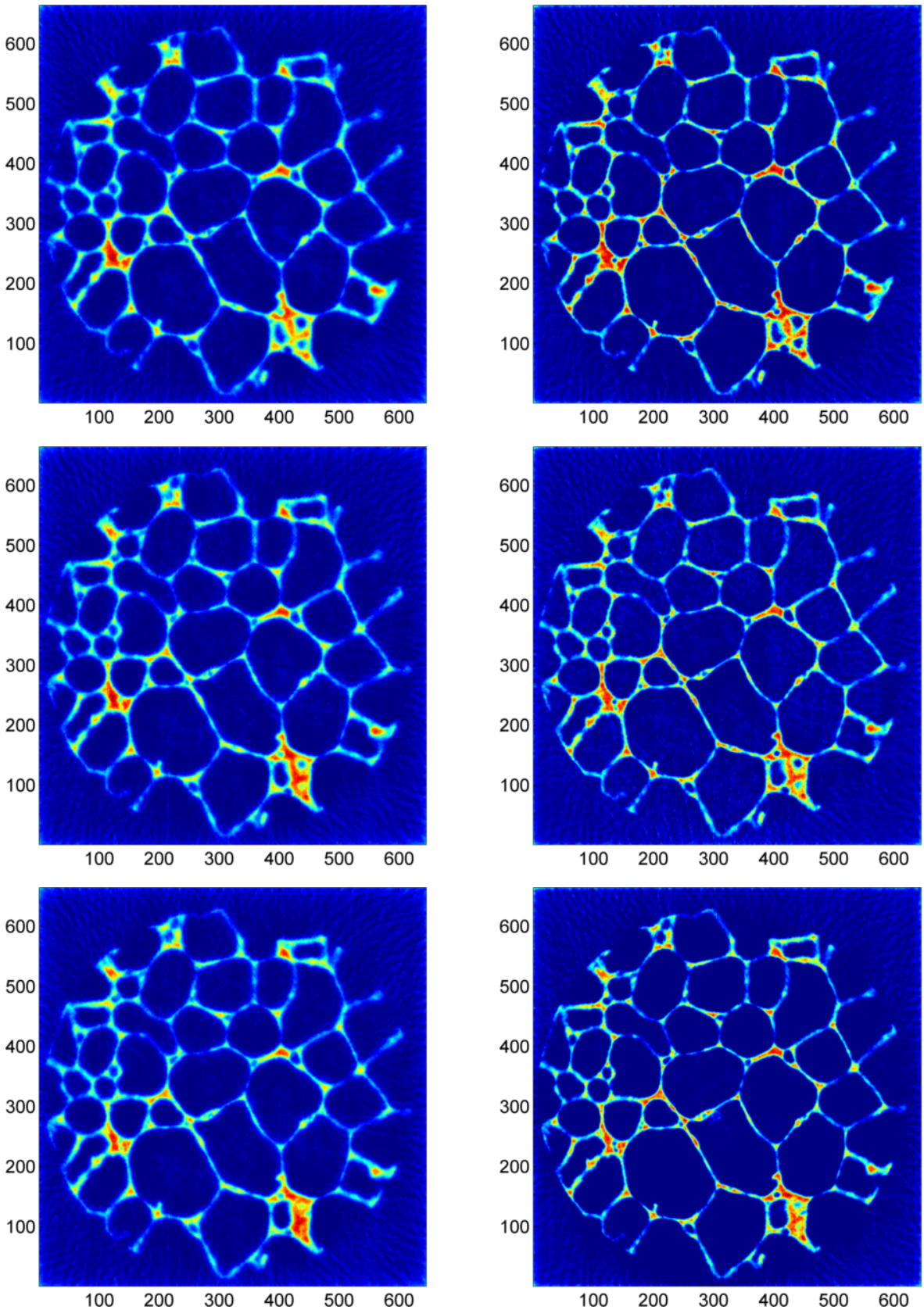


Figure 20: Three different slices of the 3D CGLS reconstruction (10 iterations) span the left column, with the 3D TV Reg reconstruction of the corresponding slices on the right ($\alpha = 0.003, \tau = 0.0001$).

B The CGLS Implementation

B.1 CGLS code description

The base directory is cgls_XTek. The main part of the code is contained in cgls_XTek_single.m, which contains the CGLS algorithm given in Section 3. The forward, Ax, and back, A^Tb, projection processes are implemented efficiently using a fast implementation of Jacob's ray tracing method, [1], [7]. A description of the code is given below, with more information provided within each file itself.

In cgls_XTek folder:

- cgls_XTek_single.m // The main function
- setup.m // Script for building mex files

- reconPhantomTest.m // Quick test to check algorithm is working
- reconPhantom.m // Reconstruct a 3-cube phantom with small Xtek geometry
- recon2D.m // Reconstruct the central slice in 2-D using data from file
- recon3D.m // Reconstruct in 3D using data from file

In cgls_XTek/tools folder:

- create_phantom.m: // Create phantom data and set-up geometry of small XTek machine
- load_data.m // Read data and geometry parameters in from file
- convert2D.m // Convert the data to 2D (central slice only)
- centre_geom.m ⇒ find_centre.m // Find the centre of rotation (give link to paper)
- CBproject_single.m: // This calls the mex function CBproject_single_newgeom_c.c which // performs the forward projection Ax
- CBbackproject_single.m: // This calls the mex function CBproject_single_newgeom_c.c which // performs the back projection A^Tb
- scrollView.m: // View the data with a slider to scroll through the last dimension

In cgls_XTek/c folder:

- CBproject_single_newgeom_c.c ⇒ project_singledata.c // Perform forward projection using Jacobs ray tracing
- CBbackproject_single_newgeom_c.c ⇒ backproject_singledata.c // Perform back projection using Jacobs ray tracing
- jacobs_rays.h // Header file for projections

Implementation Notes

- The original ray tracing code (and ScrollView) was written by David Szotten
- This code was adapted to perform forward and back projections by Will Thompson
- Further changes have been made by myself (Nicola Wadeson)

B.2 Using the Code

Notes

- Warning... You will require a lot of RAM...
- Muliple processors are desirable
- TV Reg also uses 5 iterations of CGLS code as a starting point
- Instructions are given for 64bit linux machines. If using windows, refer to document links below for how to compile mex files.

Mex Files
For an introduction to mex files, and information on setting up and using mex files, follow the links below:

`http://www.mathworks.co.uk/help/techdoc/matlab_external/f29322.html` (Introducing mex files)
`http://www.mathworks.co.uk/help/techdoc/matlab_external/f23674.html` (Building mex files)
`http://www.mathworks.co.uk/help/techdoc/apiref/bqoqnz0.html` (List of mex functions)

OpenMP
Uses open MP for parallel programming for forward and backward projections. If **T** is the number of threads you wish to use then before opening MATLAB type the following command in the bash shell (OpenSuse):

$ export OMP_NUM_THREADS=T

MATLAB commands: Setting up the code and testing with phantom data

- Load matlab
- Change to cgls_Xtek folder
- If this is a first time using mex files type:

  >> **mex -setup**

  and select the compiler you wish to use.
- Build the mex files by running

  >> **setup**

  to build the code, with largeArrayDims option and flags for OpenMP. Re-run this command any time changes are made to mex files
- Test everything works by running the example script:

  >> **reconPhantomTest**

  This is a test script for the cgls algorithm with few rays and voxels. (open the script for more information - try changing the parameters)
- For a more realistic test with number of rays as in small Xtek machine run:

  >> **reconPhantom**

  The geometry is that of the small Xtek machine with number of rays relating to number of detectors. The phantom consists of three cubes, and the data is created by forward projection (using ray tracing methods) and then adding noise to this data. This data (b) is then passed to the cgls algorithm to solve for A.

- Play with the number of reconstruction voxels - the reconstruction volume is fixed so this will change the size of the voxels.

MATLAB commands: Reconstructing real data

- It is advisable to reconstruct the central slice in 2D first to check everything is ok.
- The .xtekct file contains all the parameters used in the data collection process (with projection angles contained in _ctdata.txt). This can be found in the folder along with the projection data. The pathname and filename of the data to be reconstructed must be set before running recon2D/recon3D. The pathname being the absolute path to the data directory and the filename being the name of the .xtekct file (without the extension), in this directory. (examples are given)

  >> **pathname = '/disk2/Aluminium_foam/FoamLoadSpeed1/';**

  >> **filename = 'FoamLoadSpeed1_0.05mmmin_3';**
- When the data is FDK reconstructed, another file ending in *reconstruction.xtekct is created with details of the parameters used in the reconstruction, such as number of voxels, voxelSize, etc... If you wish to reconstruct the data with the same parameters for direct comparison then set the path and filename of this file to pathname2 and filename2, for example,

  >> **pathname2 = '/home/nwadeson/Aluminium_foam/FDK_reconstructions/';**

  >> **filename2 = 'FoamLoadSpeed1_0.05mmmin_3_reconstruction';**
- This second path and file name is optional. There are two options for the call to the load_data function in recon2D and recon3D depending on whether you are using one file or two. You must uncomment the relevant one.
- Perform the chosen reconstruction:

  >> **recon2D**

  OR

  >> **recon3D**
- Play with the parameters at the beginning of recon*.m for different reconstructions.

recon2D.m (Brief description: See file for more details)

- Sets parameters at the beginning of the file
- Loads data and geometrical parameters from file
- Converts data to 2D (central slice only)
- Finds the centre of rotation (This uses the method given in [3])
- Optionally performs simple beam hardening correction
- Calls cgls main function (Final output is cglsOut).

recon3D.m (Brief description: See file for more details)

- Sets parameters at the beginning of the file
- Loads data and geometrical parameters from file
- Finds the centre of rotation
- Optionally performs simple beam hardening correction
- Calls cgls main function (Final output is cglsOut).

If using optional second file, the voxel size and number of voxels will be read in from file, so the parameter specified in recon*.m is overwritten. If NOT using optional second file, the number of voxels is that given as a parameter.

C The TV Regularisation Implementation

C.1 TV Reg code description

The base directory is newTV_Reg. The original MATLAB implementation, developed by Tobias Lindstrøm Jensen et.al, can be downloaded from `http://www2.imm.dtu.dk/~pch/TVReg/`, along with detailed documentation. The adapted implementation used here no longer reads in a matrix A, but uses the forward and back projection methods as in the CGLS implementation, to allow much larger 3D volume calculations. The CGLS code described in the previous section is also called here, with 5 iterations providing a starting vector x_0. The main part of the code containing the algorithm described in Fig. 9 is contained in newTV_Reg/c/tv_core.c. A description of the code is given below, with more information provided within each file itself.

In newTV_Reg folder:

- tvreg_upn.m // The main MATLAB function
- install_linux64.m // Script for building mex files

- recon2D_TV.m // Reconstruct the central slice in 2-D using data from file
- recon3D_TV.m // Reconstruct in 3D using data from file

In newTV_Reg/c folder:

- tvreg_upn_c.c // Mex function which calls the main TV Reg function
- tv_core.c // The main TV Reg function
- project_singledata.c // Perform forward projection using Jacobs ray tracing
- backproject_singledata.c // Perform back projection using Jacobs ray tracing
- tools.c // Contains a number of functions used in tv_core
- jacobs_rays.h // Header file for projections
- tools.h // Header file for tools
- tv_core.h // Header file for tv_core

Implementation Notes

- The original code was created by Tobias Lindstrøm Jensen, Jakob Heide Jørgensen, Per Christian Hansen, and Søren Holdt Jensen.
- Implementation amended to remove stored matrix A by Nicola Wadeson.

C.2 Using the Code

For details of mex file setup and multiple threading of projections see Section B.2. Change the file paths (addpath commands) in recon2D_TV.m and recon3D_TV.m to point to your cgls_XTek folder.

MATLAB commands: Setting up the code and reconstructing real data

- Load matlab
- Change to newTV_Reg folder

- Build the mex files if necessary by running

  >> **install_linux64**

  and re-run each time changes are made to the code.

- Reconstruct the central slice in 2D first with different values of α, τ, L and μ. These parameters are set in the script recon2D_TV.m or recon3D_TV.m. More details are provided in these files.

- As in Section. B.2 enter the path and filename of the data to be reconstructed, for example,

  >> **pathname = '/disk2/Aluminium_foam/FoamLoadSpeed1/';**
  >> **filename = 'FoamLoadSpeed1_0.05mmmin_3';**

- Add the (optional) path and filename of the FDK reconstruction file,

  >> **pathname2 = '/home/nwadeson/Aluminium_foam/FDK_reconstructions/';**
  >> **filename2 = 'FoamLoadSpeed1_0.05mmmin_3_reconstruction';**

- Uncomment the relevant load_data function call in recon2D or recon3D.

- Perform the chosen reconstruction:

  >> **recon2D_TV**
  OR
  >> **recon3D_TV**

In order to reconstruct on the same number of voxels as FDK in x and y (using second file option) but with a smaller number of z voxels (for speed), you must uncomment these lines in tvreg_upn.m and cgls_XTek_single.m:

else
geom.nVoxels(3) = 41;
replacing 41 with the number of z voxels you desire.