TOmographic MOdel-BAsed Reconstruction (ToMoBAR) software for high resolution synchrotron X-ray tomography

Daniil Kazantsev and Nicola Wadeson

Abstract—**ToMoBAR is an open-source software which contains a library of model-based regularised iterative reconstruction methods with a plug-and-play capability. With the help of flexible data models and the selection of regularisers, one can achieve a quantifiable image quality for a variety of challenging reconstruction problems. The main aim of ToMoBAR is to solve big real data reconstruction problems in a short period of time with computational efficiency. In this paper, we outline the main functionality of the package in application to simulated data and real tomographic data obtained at Diamond Light Source (UK) synchrotron.**

Index Terms—**iterative methods, splitting methods, regularisation, open-source software**

I. Introduction

THE challenge of processing big data effectively and efficiently is crucial for many synchrotron facilities which can collect up to several petabytes of data annually. At Diamond Light Source (DLS) synchrotron (UK), the data is collected from over 30 beamlines and integrated facilities. Although the tomographic data is processed using the MPI protocols, it remains important for reconstruction tools to be computationally efficient.

It is known that the image reconstruction of big data (e.g. $(4k)^3$ voxels in size) is a demanding computational task. Many synchrotrons, including DLS, routinely use direct reconstruction methods, such as Direct Fourier Inversion and Filtered Back-projection (FBP) [1]. Notably, there are efficient realisations of direct methods, namely: the Gridrec method [2] (enables fast polar-Cartesian coordinates interpolation for inverse FFT) from the TomoPy package [3] and the GPU-based ray-tracing FBP methods of the ASTRA toolbox [4]. While direct reconstruction enables a fast parameters-free recovery, it performs poorly for undersampled, noisy and erroneous data. Iterative reconstruction (IR) methods, however, can deal much better with problematic measurements by more realistic data modelling. The drawbacks of IR methods are significantly more computationally demanding implementations and the variety of parameters requiring optimisation to achieve a better reconstruction. Fortunately, with the growth of computing capabilities it becomes possible to perform complex IR tasks on CPU/GPU hardware architectures within reasonable time.

There are a number of leading reconstruction software packages which deliver optimised low-level routines, such as forward and back-projection as well as various direct and IR methods (see Table I). In this paper, we primarily focus on IR software for X-ray computed tomography with the exception of STIR for PET/SPECT imaging [6]. In Table I, the standalone low-level open-source packages are presented with core elements implemented in: C, C++, and CUDA languages. Many of these packages have wrappers in Matlab and/or Python providing easy access to efficient core modules. The packages support different reconstruction geometries and have a set of IR algorithms that are either algebraic, based on solving the set of linear equations, (ART, SIRT, CGLS, POCS) or statistical (MLEM, OSEM).

TABLE I
STANDALONE PACKAGES FOR TOMOGRAPHIC RECONSTRUCTION

Software Version	Stands for	IR methods Geometry	Compilers Interpreters
ASTRA [4] 1.9.9.dev	All Scale Tomographic Reconstruction Antwerp	SART, SIRT, CGLS, MLEM parallel, fan, cone-beam	C++, CUDA Matlab, Python
TIGRE [5] 0.1.8	Tomographic Iterative GPU-based REconstruction	SART, OS-SART, SIRT, CGLS MLEM, TV-POCS parallel, fan, cone-beam	CUDA Matlab, Python
TomoPy [3] 1.7.0	Tomographic Reconstruction in Python	ART, BART, MLEM, OSEM, PML, OSPML, TV-methods parallel	C, CUDA Python
STIR [6] 4.0	Software for Tomographic Image Reconstruction	MLEM, OSEM, OSL, OSSPS specific to PET/SPECT	C++

In Table II, we present a group of packages which can be loosely categorised as high-level tomographic software. Normally, these packages have a dependency on low-level software and are written using interpreted languages. Notably, Python remains a first choice for many high-level packages due to convenient object-oriented elements and open-source nature. Most of the packages in Table II use Python classes to wrap low-level modules of software in Table I.

Additionally the IR methods available with high-level packages are more complex and tend to deal with various challenging aspects of non-linear optimisation. There is a

D. Kazantsev and N. Wadeson are with Diamond Light Source Ltd. Diamond House, Harwell Science and Innovation Campus, Fermi Ave, Didcot OX11 0DE; e-mail: daniil.kazantsev@diamond.ac.uk

TABLE II
HIGH-LEVEL SOFTWARE FOR TOMOGRAPHIC RECONSTRUCTION

Software Version	Stands for	IR methods	Interp.
ODL [7] 0.7.0	Operator Discretization Library	primal-dual ADMM, PDHG CGLS, BFGS with **ASTRA** projectors	Python
SIRF [8] 19.10	Synergistic Image Reconstruction Framework	methods inherited from **STIR, CIL**	Python
CIL [9] 19.10	Core Imaging Library	PDHG, SDPDHG, ADMM, FISTA, GD, CGLS with **ASTRA, STIR** projectors	Python
ToMoBAR [10] 20.01	TOmographic Model-BAsed Reconstruction	FISTA, OSFISTA ADMM with **ASTRA** projectors	Python, Matlab
Savu [11] 2.4	Tomography Reconstruction and Processing Pipeline	methods inherited from **ASTRA, TIGRE, TomoPy, ToMoBAR**	Python

substantial effort to implement and employ various splitting algorithms of primal-dual nature to deal with nondifferentiable functionals [12]. Due to the flexibility of dealing with complicated multi-term objective functions, methods such as Alternating Direction of Multipliers (ADMM) [13], Primal-Dual Hybrid Gradient (PDHG) [12], a Fast Iterative Shrinkage-Thresholding Algorithm (FISTA) [14], and others are proven to be successful for advanced tomographic reconstruction.

The presented ToMoBAR package also contains the implementation of splitting algorithms, however the main purpose is *to reconstruct large tomographic data efficiently*. In contrast to ODL and CIL packages which aim to solve inverse problems with mathematical rigor, ToMoBAR is driven mostly by computational efficiency and the challenge to minimise various imaging artifacts through nonlinear, and in some cases non-convex, data models [15]. Because of this, for some cases the global solution is not guaranteed, however in practice, for problematic data cases it is usually possible to obtain quantifiable results of a better quality. Apart from various modifications to accelerate convergence and compensate for artifacts, ToMoBAR has also the traditional algorithms with proven convergence which can be used for benchmarking. Additionally, ToMoBAR performs regularisation through the CCPi Regularisation Toolkit [16] which contains more than 10 different CPU/GPU regularisers. With various data models and regularisers one can construct multiple variations of IR algorithms using ToMoBAR.

Being a part of Savu [11] (MPI-based tomographic application for large data), ToMoBAR has an access to multiple CPU/GPU nodes of a computing cluster. With parallel-beam geometry setup at DLS one can iteratively reconstruct $(2k)^3$ voxels volumes using ToMoBAR in under three minutes time.

II. TOMOBAR MODELS AND SOFTWARE DESCRIPTION

ToMoBAR is an open-source Python and Matlab toolbox to perform tomographic data processing and image reconstruction tasks. Inherently it uses the low-level forward and back-projection modules of ASTRA toolbox [4]. The direct and IR methods from ASTRA are also wrapped for ToMoBAR for a quicker access. At the moment, ToMoBAR supports only parallel beam geometry since the main aim is to reconstruct synchrotron tomographic data. It can be generalised, however, to any geometry which ASTRA allows. To install ToMoBAR in Python one can use Conda environment and for Matlab one can simply copy the source code. ToMoBAR has dependencies on ASTRA toolbox [4] and the CCPi Regularisation Toolkit [16].

ToMoBAR v.20.01 consists of two main IR reconstruction algorithms: FISTA and ADMM. While the ADMM algorithm follows a classical implementation [13], FISTA has been modified in a number of ways. With the ordered-subset strategy, FISTA has been accelerated resulting in OSFISTA [17], to give up to 10 times faster convergence than the original method. A number of different data models has been added and due to separability of the objective function one can select a suitable regulariser from the CCPi Regularisation Toolkit [16].

The reconstruction task is to recover the unknown attenuation coefficient distribution (the absorption map) $\mathbf{x} \in \mathbb{R}^N$ using the log-corrected normalised tomographic projection data $\mathbf{b} = -\ln\left(\frac{\mathbf{Y}}{\mathbf{I_0}}\right)$, where $\mathbf{Y} = \{y_j\}_{j=1}^M$ are raw measurements and $\mathbf{I_0}$ is the intensity of the incoming beam.

The following regularised optimisation problem needs to be solved:

$$\min_{\mathbf{x}} f(\mathbf{Ax}, \mathbf{b}) + \beta g(\mathbf{x}),$$

where $\mathbf{A} \in \mathbb{R}^{M \times N}$ is the system projection matrix, $f : \mathbb{R}^M \to \mathbb{R}_+$ is continuously differentiable data misfit term, $g : \mathbb{R}^N \to \mathbb{R}_+$ is a convex penalty, and β is the regularisation parameter.

Let $\mathbf{r} = (\mathbf{Ax} - \mathbf{b})$, then for the least-squares (LS) data model we have $f(\mathbf{Ax}, \mathbf{b}) = \mathbf{r}^\top \mathbf{r}$ and for penalised weighted least-squares (PWLS) model $f(\mathbf{Ax}, \mathbf{b}) = \mathbf{r}^\top \mathbf{\Lambda}^{-1} \mathbf{r}$. Here the noise covariance matrix $\mathbf{\Lambda} = \sigma^2 \mathbf{I}$ and one usually select $\sigma_j^2 = y_j$. For PWLS model the weights can be also absorbed as $\mathbf{A} \equiv \sqrt{\mathbf{\Lambda}}\mathbf{A}$ and $\mathbf{b} \equiv \sqrt{\mathbf{\Lambda}}\mathbf{b}$ [15]. Additionally one can consider a more realistic Poisson noise model and minimise Kullback-Leibler divergence: $f(\mathbf{Ax}, \mathbf{b}) = \sum_{j=1}^{M} (b_j (\log b_j - \log[\mathbf{Ax}]_j))$.

In Table III we provide various data and regularisation models available in OSFISTA method of ToMoBAR v.20.01. Note that apart from the LS and PWLS data models, other models are not mutually exclusive and can be stacked together to form more complex functionals. For instance, one can use the Group-Huber data penalty [18], [15] to minimise ring artifacts together with the Huber data penalty and PWLS forming the following cost function:

$$\min_{\mathbf{s}} \tfrac{1}{2} \|\mathbf{s} - \mathcal{L}^\top \rho(\mathbf{r})\|_{\mathbf{\Lambda}}^2 + \beta g(\mathbf{x}) + \lambda \|\mathbf{s}\|_1.$$

TABLE III
ToMoBAR MODELS WITH OSFISTA ALGORITHM

Data model	$f(\mathbf{Ax}, \mathbf{b}), \mathbf{r} = \mathbf{Ax} - \mathbf{b}$	Regulariser	$g(\mathbf{x})$
LS	$\frac{1}{2}\Vert\mathbf{r}\Vert^2_{\mathbf{\Lambda}=\sigma^2\mathbf{I}}$	Rudin-Osher-Fatemi TV	$\Vert\nabla\mathbf{x}\Vert_{TV,\epsilon}$
PWLS	$\frac{1}{2}\Vert\mathbf{r}\Vert^2_{\mathbf{\Lambda}=\mathbf{Y}}$	primal-dual / fast gradient projection / split-Breg exact TV	$\Vert\nabla x\Vert_{TV}$
KL	$\sum_{j=1}^{M}(b_j(\log b_j - \log[\mathbf{Ax}]_j))$	Nonlinear diffusion	$\int\phi(\Vert\nabla\mathbf{x}\Vert^2)$
Huber	$\rho(\mathbf{r}) = \sum_{j=1}^{M}\phi(r_j)$ $\phi(r) = \begin{cases}\frac{1}{2}r^2, \lvert r\rvert \leq \sigma, \\ \sigma(\lvert r\rvert - \frac{1}{2}\sigma)\end{cases}$	Total General. Variation	$\alpha_1\Vert\nabla\mathbf{x} - \mathbf{v}\Vert$ $+ \alpha_0\Vert\mathcal{E}(\mathbf{v})\Vert$
Student's t	$\rho(\mathbf{r}) = \sum_{j=1}^{M}\phi(r_j)$ $\phi(r) = \log(1 + (r/\sigma)^2)$	ROF-LLT	$\alpha_1\Vert\nabla\boldsymbol{x}\Vert_{TV,\epsilon}$ $+ \alpha_2\Vert\nabla^2\boldsymbol{x}\Vert_{\epsilon_2}$
Group-Huber	$\min_{\mathbf{s}}\frac{1}{2}\Vert\mathbf{s} - \mathcal{L}^\top\mathbf{r}\Vert_2^2$ $+\lambda\Vert\mathbf{s}\Vert_1$, $\boldsymbol{\mathcal{L}} = (\mathbf{I}_{m_1}\otimes\mathbf{1}_{m_2})/\sqrt{m_2}$ with $\mathcal{L}^\top\mathcal{L} = \mathbf{I}_{m_1}$	Nonlocal TV	$\Vert\nabla_\epsilon(\omega)\mathbf{x}\Vert_{TV}$

Any suitable regulariser $g(\mathbf{x})$ can be chosen from Table III using the CCPi Regularisation Toolkit [16]. The resulting objective can be optimised using a splitting algorithm and in this instance we used FISTA [14].

In Fig. 1, we demonstrate a Python code snippet from ToMoBAR which enables the OSFISTA algorithm with the PWLS data model and TV regularisation.

```
1 from tomobar.methodsIR import RecToolsIR
2 # Set scanning geometry parameters and initiate a class object
3 tomo = RecToolsIR(DetectorsDimH = detHoz,  # Horizontal detector dimension
4                   DetectorsDimV = None,  # Vertical detector dimension (3D data)
5                   CenterRotOffset = None,# Center of Rotation scalar (3D data)
6                   AnglesVec = AnglesRad, # Array of projection angles in radians
7                   ObjSize = N,           # Reconstructed object dimensions (scalar)
8                   datafidelity='PWLS',   # Data fidelity, choose from LS, KL, PWLS
9                   device_projector='gpu')
10 ####################### Creating the data dictionary: #######################
11 _data_ = {'projection_norm_data' : dataNorm, # Normalised projection data
12           'projection_raw_data' : dataRaw,   # Raw projection data
13           'OS_number' : 6                    # Number of ordered subsets
14           }
15 lc = tomo.powermethod(_data_) # calculate Lipschitz constant
16 ## Creating the regularisation dictionary using the CCPi regularisation toolkit: ##
17 _regularisation_ = {'method' : 'PD_TV',       # Selected regularisation method
18                     'regul_param' : 0.000001, # Regularisation parameter
19                     'iterations' :80,         # Number of regularisation iterations
20                     'device_regulariser': 'gpu'}
21 ####################### Creating the algorithm dictionary: #######################
22 _algorithm_ = {'iterations' : 25,          # Number of algorithm iterations
23                'lipschitz_const' : lc}
24 # RUN THE FISTA RECONSTRUCTION METHOD:
25 RecFISTA_os_tv_pwls = tomo.FISTA(_data_, _algorithm_, _regularisation_)
```

Fig. 1. The snippet of a Python code in ToMoBAR enabling the OSFISTA algorithm with the PWLS data-model and Primal-Dual TV [12] penalty term. There are four main components for IR methods in ToMoBAR which need to be set: 1) the data-model and the geometry class (line 3); 2) the data dictionary (line 11); 3) the regularisation dictionary (line 17) and 4) the algorithm dictionary (line 22). One can access the detailed information on parameters by typing *help(RecToolsIR)* in Python command line.

In the next section we demonstrate several numerical examples of how the stacked models of ToMoBAR work synergistically to ensure better reconstruction quality.

III. NUMERICAL EXPERIMENTS

For our experiments with simulated data we use the TomoPhantom software [19], which is a convenient tool to access an extensive library of enumerated 2D-4D phantoms and their analytical Radon transforms. TomoPhantom can also model various imaging artifacts and noise. In this example we use model no. 14 and to the data we add Poisson noise, zingers (dead pixels) and detector offsets. Zingers and offsets can happen due to scattering and detector miscalibration processes, respectively. In the reconstructed images they lead to extensive streaks and ring artifacts [15]. These nonlinear data errors do not fit to the expected linear data model which is used routinely for IR reconstruction. For such cases, one needs to either preprocess (filter) the data or to use a more suitable data model. Since we use the IR methods, one might want to incorporate more realistic data models rather than modifying the raw data which can lead to various corruptions of the reconstructed images.

In Fig. 2 one can see the results of the reconstructed 512×512 phantom using only 256 noisy projections with imaging errors. Expectantly, the FBP reconstruction is noisy and contains multiple errors. The regularised LS OSFISTA-TV algorithm is able to successfully suppress noise and some minor artifacts, however one can clearly see (also noticeable in the image reconstruction error) the streaks and ring artifacts. Applying the Huber data penalty (see Table III) one can suppress the contribution of zingers (streaks) but the ring artifacts overall remain. Clearly one needs a better data model to be able to suppress ring artifacts. One such model has been proposed in [18] and we refer to it as a Group-Huber model [15] (see Table III). In Fig. 2 we employ such model on top of the existing Huber data model. One can see that it is possible to suppress effectively ring artifacts as well as streaks and noise. The resulting RMSE quality metric also shows the smallest error using the Huber-GH-TV model.

In Fig.3 we demonstrate the use of ToMoBAR in application to real data of dendritic growth collected at DLS synchrotron [20]. Here we show how one can use the KL data term together with the GH model and different regularisers. Since the data is seriously undersampled and also low-dose, the FBP reconstruction is very noisy. Applying OSFISTA algorithm with the TV regulariser increases the quality of the image significantly. The ring artifacts, however, remain a serious problem for the subsequent image analysis. Applying the GH data fidelity one can suppress the ring artifacts substantially. One can also argue that the piecewise-constant nature of TV regularisation is not very suitable for the dendritic surfaces. Thus it can be easily substituted to piecewise-smooth regularisation model [16] resulting in a new LK-GH-ROF/LLT reconstruction method.

IV. CONCLUSION

We have presented an open-source ToMoBAR software which is a practical and efficient tool to deal with the challenging data of X-ray tomography. With a variety of data and regularisation models it is possible to construct a suitable objective for a particular problem. We demonstrate the Python

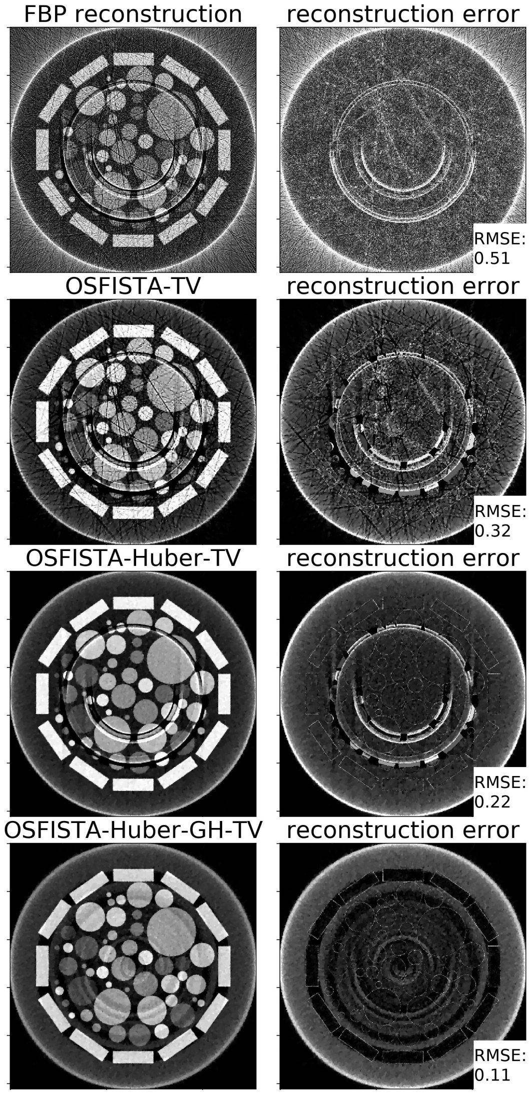

Fig. 2. ToMoBAR reconstructions using the simulated projection data with noise and imaging artifacts. Various data models exploited to show the difference.

syntax of ToMoBAR which makes it easier to construct a desirable cost function.

References

[1] A. C. Kak and M. Slaney, *Principles of computerized tomographic imaging*, N.Y.: IEEE Press, 1988.

[2] F. Marone and M. Stampanoni, *Regridding reconstruction algorithm for real-time tomographic imaging*. Journal of synchrotron radiation, 19(6), pp.1029–1037, 2012.

[3] D. Gürsoy, F. De Carlo, X. Xiao and C. Jacobsen, *TomoPy: a framework for the analysis of synchrotron tomographic data*. Journal of Synchrotron Radiation, 21(5), pp. 1188–1193, 2014.

[4] W. van Aarle, W. J. Palenstijn, J. Cant, E. Janssens, F. Bleichrodt, A. Dabravolski, J. De Beenhouwer, K. J.Batenburg, J. Sijbers, *Fast and flexible X-ray tomography using the ASTRA toolbox*. Optics Express, 24(22), pp. 25129–25147, 2016.

[5] A. Biguri, M. Dosanjh, S. Hancock and M. Soleimani,*TIGRE: a MATLAB-GPU toolbox for CBCT image reconstruction*. Biomedical Physics & Engineering Express, 2(5), p.055010, 2016.

[6] K. Thielemans, C. Tsoumpas, S. Mustafovic, T. Beisel, P. Aguiar, N. Dikaios and M. W. Jacobson, *STIR: software for tomographic image reconstruction release 2*. Physics in Medicine & Biology, 57(4), p.867, 2012.

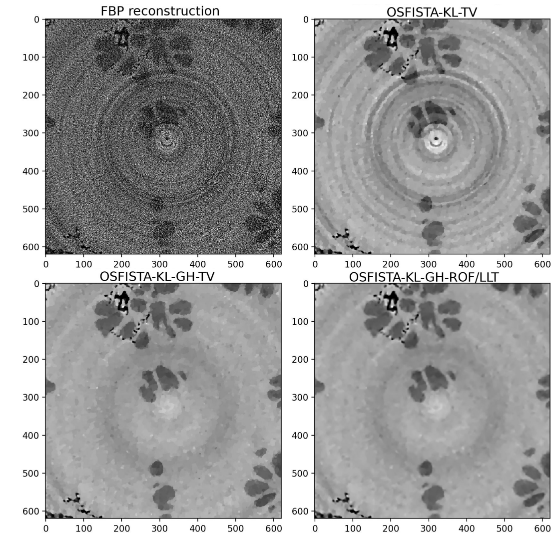

Fig. 3. ToMoBAR reconstructions in application to the real data using KL and GH data models with different regularisers.

[7] J. Adler, H. Kohr and O. Oktem. *Operator discretization library (ODL)*. Software available from https://github.com/odlgroup/odl, 2017.

[8] E. Ovtchinnikov, R. Brown, C. Kolbitsch, E. Pasca, C. da Costa-Luis, A. G. Gillman, B. A. Thomas, N. Efthimiou, J. Mayer, P. Wadhwa and M. J. Ehrhardt, *Sirf: Synergistic image reconstruction framework*. Computer Physics Communications, p.107087, 2019

[9] E. Ametova, G. Fardell, J. Jorgensen, D. Kazantsev, S. Nagella, E. Pasca, E. Papoutsellis, *Core imaging library for image reconstruction*. Software available at https://www.ccpi.ac.uk/CIL, 2019

[10] D. Kazantsev, *TOmographic MOdel-BAsed Reconstruction software*. Software available at https://github.com/dkazanc/ToMoBAR, 2019.

[11] N. Wadeson and M. Basham, *Savu: A python-based, mpi framework for simultaneous processing of multiple, n-dimensional, large tomography datasets*. arXiv preprint arXiv:1610.08015, 2016

[12] A. Chambolle and T. Pock, *A first-order primal-dual algorithm for convex problems with applications to imaging*. Journal of mathematical imaging and vision 40(1), 2010.

[13] B. Wahlberg, S. Boyd, M. Annergren and Y. Wang, *An ADMM algorithm for a class of total variation regularized estimation problems*. IFAC Proceedings Volumes, 45(16), pp.83–88, 2012.

[14] A. Beck, M. Teboulle, *A fast iterative shrinkage-thresholding algorithm for linear inverse problems*. SIAM journal on imaging sciences, 2(1), pp.183–202, 2009.

[15] D. Kazantsev, F. Bleichrodt, T. Van Leeuwen, A. Kaestner, P. J. Withers, K. J. Batenburg and P. D. Lee, *A novel tomographic reconstruction method based on the robust Student's t function for suppressing data outliers*. IEEE Transactions on Computational Imaging, 3(4), pp.682–693, 2017.

[16] D. Kazantsev, E. Pasca, M. J. Turner and P. J. Withers, *Ccpi-regularisation toolkit for computed tomographic image reconstruction with proximal splitting algorithms*. SoftwareX, 9, pp.317–323, 2019.

[17] Q. Xu, D. Yang, J. Tan, A. Sawatzky and M. A. Anastasio, *Accelerated fast iterative shrinkage thresholding algorithms for sparsity-regularized cone-beam CT image reconstruction*. Medical physics, 43(4), pp.1849–1872, 2016.

[18] P. Paleo, A. Mirone, *Ring artifacts correction in compressed sensing tomographic reconstruction*. Journal of synchrotron radiation, 22(5), pp.1268–1278, 2015.

[19] D. Kazantsev, V. Pickalov, S. Nagella, E. Pasca, P. J. Withers, *TomoPhantom, a software package to generate 2D-4D analytical phantoms for CT image reconstruction algorithm benchmarks*. SoftwareX, 7, pp.150–155, 2018.

[20] E. Guo, S. Shuai, D. Kazantsev, S. Karagadde, A. B. Phillion, T. Jing, W. Li and P. D. Lee, P.D, *The influence of nanoparticles on dendritic grain growth in Mg alloys*. Acta Materialia, 152, pp.127–137, 2018.