A general software framework for regularisation

This note discusses various issues and opportunities associated with building up a general software framework for regularisation in tomographic reconstruction. The main motivation is to avoid implementing all necessary software components from the ground up, *i.e.*, to make use of existing forward model and optimisation softwares.

1 Mathematical setting

Most tomographic imaging problems can be formulated in a common framework as an inverse problem. Now, an image can be represented by real-valued functions, *e.g.*, a 3D image is represented by real-valued functions that models a 3D spatial distribution of some quantity of interest and temporal variation of this spatial distribution adds another dimension. Hence, in general a k-dimensional image is represented by a function $f\colon \Omega \to \mathbb{R}$ where $\Omega \subset \mathbb{R}^k$. The goal is now to *reconstruct* the function (*image*) $f_{\text{true}} \in \mathscr{X}$ from *measured data* $\boldsymbol{g} \in \mathbb{R}^m$ assuming

$$\boldsymbol{g} = \boldsymbol{\mathcal{T}}(f_{\text{true}}) + \boldsymbol{g}_{\text{noise}}. \tag{1}$$

In the above, we have the following:

- $\mathscr{X}$ (*reconstruction space*) is the vector space of feasible images.
- $\mathscr{Y}$ (*data space*) is a vector space of all possible continuum data (*i.e.*, data assuming no discretisation due to sampling), $Y \subset \mathbb{R}^m$ is the corresponding set of digitsed data, and $P_{\text{data}}\colon \mathscr{Y} \to Y \subset \mathbb{R}^m$ models the actual digitisation of data.
- The mapping $\boldsymbol{\mathcal{T}}\colon \mathscr{X} \to Y$ is the *(partly) discretised forward model*

  $$\boldsymbol{\mathcal{T}} := P_{\text{data}} \circ \mathcal{T}$$

  where $\mathcal{T}\colon \mathscr{X} \to \mathscr{Y}$ (*forward model*) describes how an image gives rise to continuum data in absence of noise and digitisation.
- The array $\boldsymbol{g}_{\text{noise}} \in \mathbb{R}^m$ is a sample of a random variable representing the noise component of data (note that it may the case that $\boldsymbol{g}_{\text{noise}} \notin Y$).

Many inverse problems in imaging are ill-posed, *i.e.*, (1) lacks a unique maximum likelihood solution and/or such a solution is overly sensitive to small variations in data (instability). There are various approaches for solving such ill-posed problems, but a common feature is that useful methods all involve a regularisation that introduces uniqueness and stability, *e.g.*, by penalising 'image complexity'.

The project will primarily focus on *variational regularisation methods* where the true (unknown) image $f_{\text{true}} \in \mathscr{X}$ is estimated by solving an optimisation problem:

$$\min_{f \in \mathscr{X}} \lambda \mathcal{S}(f) + \mathcal{D}\big(\boldsymbol{\mathcal{T}}(f), \boldsymbol{g}\big). \tag{2}$$

In the above:

- $\mathcal{S} \colon \mathscr{X} \to \mathbb{R}_+$ (*regularisation functional*) introduces uniqueness and stability by encoding a priori information about f_{true}.
- $\mathcal{D} \colon Y \times Y \to \mathbb{R}_+$ (*data discrepancy*) quantifies the goodness-of-fit. It is often given as a norm, but preferably it should be chosen so that minimising it should be equivalent to minimising the negative log-likelihood of measured data. Then, solving (2) with $\lambda = 0$ would correspond to seeking an maximum-likelihood (ML) solution.
- $\lambda \geq 0$ (*regularisation parameter*) quantifies the trade-off between stability and goodness-of-fit.

The data discrepancy is in some sense dictate by the physics of the experimental setup and choice of regularisation parameter is given by the noise level in data. On the other hand, the choice of the regularisation functional $\mathcal{S}$ is governed by what image features that are considered important and what reliable a priori information one has about f_{true}. Thus, its choice/design needs to be adapted to the imaging task at hand.

Fully discrete setting Note that unless the vector space $\mathscr{X}$ is finite dimensional, (2) is an optimisation problem over an infinite dimensional vector space. At some stage, such an optimisation problem needs to be recast to an optimisation over a finite dimensional vector space $\mathbb{R}^n$. Thus, we introduce the *image digitisation operator*

$$P_{\text{image}} \colon \mathscr{X} \to X \subset \mathbb{R}^n$$

where X is the set of arrays that correspond to feasible digital images in $\mathscr{X}$.

Next, define the *fully discrete forward operator* as the mapping $\boldsymbol{T} \colon X \to \mathbb{R}^m$ that corresponds to $\boldsymbol{\mathcal{T}}$ in the sense that

$$\boldsymbol{T} \circ P_{\text{image}} \approx \boldsymbol{\mathcal{T}} \quad \text{on } \mathscr{X}.$$

Likewise, we define the *discretised regularisation functional* as the mapping $\boldsymbol{S} \colon X \to \mathbb{R}_+$ that corresponds to $\mathcal{S}$ in (2) in the sense that

$$\boldsymbol{S} \circ P_{\text{image}} \approx \mathcal{S} \quad \text{on } \mathscr{X}.$$

Finally, we define the *discretised data discrepancy functional* as the mapping $\boldsymbol{D} \colon X \times Y \to \mathbb{R}_+$ that corresponds to $\mathcal{D}$ in (2) in the sense that

$$\boldsymbol{D}(\boldsymbol{f}, \boldsymbol{g}) \approx \mathcal{D}\big(\boldsymbol{T}(\boldsymbol{f}), \boldsymbol{g}\big) \quad \text{on } X \times Y.$$

The finite dimensional optimisation problem corresponding to (2) now reads as

$$\min_{\boldsymbol{f} \in X} \lambda \boldsymbol{S}(\boldsymbol{f}) + \boldsymbol{D}\big(\boldsymbol{T}(\boldsymbol{f}), \boldsymbol{g}\big). \tag{3}$$

2 Software framework for variational regularisation

We here outline design principles for a software framework for variational regularisation of inverse problems of the type (1). Focus is on image reconstruction from tomographic data in medical imaging. The structure in (2) and its relation to (3) clearly points to a natural modularization for such a software framework.

2.1 Data pre-processing software

Measured raw data often needs to undergo some pre-processing. Hence, there is often need for software components that perform such pre-processing, which typically is given as a number of data-to-data mappings, *i.e.*, transformations that map an array representing raw data into another array representing data $\boldsymbol{g} \in Y$ in (1).

2.2 File I/O library

Software components to read and properly interpret both raw and pre-processed data. There are a variety of data formats to consider that depend on the imaging modalities that are used.

2.3 Forward model software

Software components for evaluating the action of the fully discrete forward operator, *i.e.*, to evaluate the map $\boldsymbol{f} \mapsto \boldsymbol{T}(\boldsymbol{f})$ given a digital image $\boldsymbol{f} \in X \subset \mathbb{R}^n$. Ideally, the same library can also evaluate the action of the adjoint of the derivative of this map.

2.4 Data discretisation library

Software components for generating arrays that correspond to digitisations of data consistent with a specific data acquisition protocol. Data is here assumed to be pre-processed. This library must make use of the file I/O library and there are several different types of data digitsations to consider.

2.5 Image discretisation library

Highly optimised software components for generating arrays of varying dimensions that correspond to different image digitisations. There are several different types of image digitsations to consider.

2.6 Optimisation software

Software components for solving optimisation problems of the type in (3). Here, one is given pre-processed digitised data $\boldsymbol{g} \in Y$ as well as routines for evaluating the discretised regularisation and data discrepancy functionals and their derivatives, and a method for choosing the regularisation parameter.

2.7 Regularisation library

This is a software framework for representing (not solving) inverse problems of the type (1) and associated regularisation methods of the type (2). The actual numerical routines for solving (2) through (3) are mostly performed by external software libraries, such as those in sections 2.1–2.6.

Such a framework needs to have certain properties, some of which we list below. Section 5 has a more detailed discussion.

2.7.1 Functional analysis library

This is a middle-ware that provides a system of types for numerical functional analysis that enables one to *define* and *represent* infinite dimensional variational problems of the type (2) independently of implementation related details, which require explicit data and image digitisations.

More specifically, the library must offer types and data structures for representing elements in infinite dimensional vector spaces, operators and functionals defined on such spaces, and operations between such structures. Some specific features are:

1. Representing data digitisation operators $P_{\text{data}} \colon \mathscr{Y} \to Y \subset \mathbb{R}^m$, *i.e.*, types and data structures for handling continuum data (element in infinite dimensional data space $\mathscr{Y}$) while retaining the couplings to the correct data digitisation routines (section 2.4) that correspond to how pre-processed measured data is formed from continuum data.

2. Representing image digitisation operators $P_{\text{image}} \colon \mathscr{X} \to X \subset \mathbb{R}^n$, *i.e.*, types and data structures for handling images (element in infinite dimensional reconstruction space $\mathscr{X}$) while retaining the couplings to the correct image digitisation routines (section 2.5) that correspond to how images are digitised.

3. Representing the discretised forward model $\mathcal{T} \colon \mathscr{X} \to Y \subset \mathbb{R}^m$ together with the adjoint of its derivative while retaining the couplings to the correct forward model software routines (section 2.3) that correspond to how images and data are digitised.

4. Representing various functionals and their derivatives, such as the regularisation functional $\mathcal{S}$ and the data discrepancy functional $\mathcal{D}$.

5. Enable consistency checks of operations involving vector space elements, functionals and operators, *e.g.*, have a system to check whether an operator composition is possible or to check whether two operators are adjoint to each other.

6. Have a system based on the chain rule for automatically assembling the first derivative of a composition of functionals given software components that provide derivatives of the latter.

7. Delegate evaluation of operations involving vector space elements, functionals and operators to numerical routines contained in external libraries that carry their own specific data structures.

8. Act as a middle-ware connecting the above functional analysis structures to routines and specific data structures provided by internal/external software for optimisation (section 2.6).

2.7.2 Regularisation functional software

Types and data structures for representing the regularisation functionals $f \mapsto \mathcal{S}(f)$ and their sensitivity (derivative) for images $f \in \mathscr{X}$. Numerical software routines for evaluating the corresponding discretised regularisation functional $\boldsymbol{S}\colon X \to \mathbb{R}_+$ given an image digitisation operator.

2.7.3 Data discrepancy software

Types and data structures for representing the data discrepancy functional $f \mapsto \mathcal{D}\big(\boldsymbol{\mathcal{T}}(f), \boldsymbol{g}\big)$ and its sensitivity (derivative) for images $f \in \mathscr{X}$ and digitised data $\boldsymbol{g} \in Y$. Numerical software routines for evaluating the corresponding discretised data discrepancy functional $\boldsymbol{D}\colon X \times Y \to \mathbb{R}_+$ given image and data digitisation operators.

2.7.4 Regularisation parameter selection software

Software components for determining regularisation parameter(s), like $\lambda > 0$ in (2), given a statistical model for the noise in pre-processed data $\boldsymbol{g} \in Y$ and the type of data discrepancy functional that is used.

2.7.5 Nuisance parameters

Structures and types for representing methods that involve solving a sequence of variational regularisation problems (2).

3 Forward model software

Forward model software is essentially a collection of software components that evaluate $\boldsymbol{\mathcal{T}}(f)$ given an image $f \in \mathscr{X}$. Ideally, the same library can also evaluate the adjoint of the derivative of $f \mapsto \boldsymbol{\mathcal{T}}(f)$. Such software needs to follow certain design principles if it is to be used as an external software library for a general software framework for regularisation. The most important ones are listed below:

Licensing: A licensing scheme that provides access to source code for research purposes.

Modularisation: The software framework should be appropriately modularised in the sense that software components are separated and organised into sub-libraries with associated application programming interfaces (APIs) that are callable from C/C++ and high-level languages like MATLAB and Python. There should be such sub-libraries for (i) file I/O routines with software primitives that encode data acquisition protocols, (ii) data pre-processing, (iii) evaluating the forward model and the adjoint of its derivative, and (iv) evaluating functionals.

Forward models: Provide computationally efficient, yet sufficiently accurate, implementations of the forward model and the adjoint of its derivative relevant for the imaging modality at hand.

I/O routines and data acquisition protocols: Provide software primitives for reading and writing files in appropriate formats. This also includes encode data acquisition protocols, such as source–detector positioning, relevant for the imaging modality at hand.

Data pre-processing: Data often undergo various pre-processing steps in order to better match (i) the forward model and (ii) the data acquisition protocol that are implicitly, or explicitly, assumed when reconstructing. These are data-to-data transformations and examples are calibration steps and correction for beam hardening and scatter. Another example is re-binning, which is used to map an actual data acquisition protocol to an ideal one that is more suitable for reconstruction.

Reconstruction methods: Provide software primitives for selected reconstruction methods.

Computational feasibility: Software components should be efficiently implemented in the sense that one can perform basic iterative reconstructions, *e.g.*, a conjugate gradient type of iterative scheme, on realistic data sets within the time-frame required by the clinical setting.

Below we list some forward model software that seem to live up to the above requirements. Note that these are actually software suites for reconstruction, however our interest is only in the forward model software part.

ASTRA

ASTRA is a C++ framework with Python and MATLAB extensions for 2D and 3D transmission tomography [31]. It is developed and maintained by the Vision Lab research group at the University of Antwerp.

Licensing: GNU GPL v3

Modularisation: Separate components for the forward problem and for file I/O. No APIs or software primitives for variational regularisation. No APIs or software primitives for representing operators or functionals. The software is however integrated with Spot, a linear-operator toolbox for MATLAB. Here, one can perform MATLAB operations with linear operators in the same way as with regular matrices, but without requiring explicit matrix representation. Hence, columns/rows of the final matrix representation of the linear operator are generated as the operator is applied, so one does not have to store the entire matrix.

Forward models: Software components for evaluating the ray transform and its associated backprojection relevant for transmission tomography applications.

I/O routines and data acquisition protocols: Software primitives to encode highly flexible source/detector positioning in 2D parallel & fan beam geometries, and in 3D parallel & cone beam geometries. Need to determine what data formats that are supported.

Data pre-processing: Need to determine what pre-processing steps that are implemented.

Reconstruction methods: Filtered backprojection (FBP), simultaneous iterative reconstruction technique (SIRT), simultaneous algebraic reconstruction technique (SART), conjugate gradient least squares (CGLS), and discrete algebraic reconstruction technique (DART).

Computational feasibility: Computationally intensive functions, like the evaluation of the forward model and the adjoint of its derivative, are GPU accelerated using the CUDA library. Furthermore, a distributed computing framework is being implemented to handle very large tomographic problems.

NiftyRec

NiftyRec is a C library with Python and MATLAB extensions for tomography [33]. It was initially developed at the Centre for Medical Image Computing, University College London during the period 2009–2012 and is currently being developed at the Martinos Center for Biomedical Imaging, Massachusetts General Hospital, Harvard University.

Licensing: BSD License

Modularisation: Separate components for the forward problem and for file I/O. No APIs or software primitives for variational regularisation. No APIs or software primitives for representing operators or functionals.

Forward models: Forward models for transmission tomography, emission tomography (with depth-dependent resolution modelling), synchrotron X-ray tomography, neutron tomography, and optical projection tomography.

I/O routines and data acquisition protocols: 3D parallel beam, cone-beam, fan-beam, and helical cone-beam geometries. Need to determine what data formats that are supported.

Data pre-processing: Need to determine what pre-processing steps that are implemented.

Reconstruction methods: maximum-likelihood expectation maximisation (ML-EM), ordered subsets expectation maximisation (OSEM), and one step late maximum a posteriori expectation maximisation (OSL-MAPEM).

Computational feasibility: Computationally intensive functions, like the evaluation of the forward model and the adjoint of its derivative, are GPU accelerated using the CUDA library.

ASPIRE

ASPIRE is a C library for tomographic image reconstruction developed at the University of Michigan by Jeff Fessler and his students [9, 8]. There is also a MATLAB image reconstruction toolbox developed by the same author that contains many of the algorithms from ASPIRE.

Licensing: The compiled executables are available for free access to both academic and industrial users for research purposes. To access the source code one must agree to the terms of the license agreement, which prohibits selling or further distributing the software.

Modularisation: Unclear since source code is not available. It is however likely that components for the forward problem and for file I/O are properly modularised. The software may also contain APIs and/or software primitives for variational regularisation. No APIs or software primitives for representing operators or functionals.

Forward models: Forward models for transmission and emission tomography, the latter with depth-dependent resolution modelling. TERSE is an extension for combined SPECT/CT imaging.

I/O routines and data acquisition protocols: 3D parallel beam, cone-beam, fan-beam, and helical cone-beam geometries. Need to determine what data formats that are supported.

Data pre-processing: Need to determine what pre-processing steps that are implemented.

Reconstruction methods: ML-EM and penalised-likelihood (variational regularisation) methods. Currently, all the penalty functions implemented on the gradient (1st-order pixel differences) and they include 2-norm of the gradient magnitude as well as the Huber function on the gradient magnitude [9].

Computational feasibility: To be determined.

Other

TomoPy is a Python toolbox to perform tomographic data processing and image reconstruction [12]. The aim is to provide a single software framework for reconstruction for tomographic datasets at synchrotron light sources (including X-ray transmission tomography, X-ray fluorescence microscopy and X-ray diffraction tomography). It has several advanced data pre-processing routines that aim to prepare data for reconstruction.

Reconstruction Toolkit is an open-source and cross-platform software for fast circular cone-beam CT reconstruction used primarily for treatment planning in radiation therapy [37]. The software is based on Insight Toolkit and it provides a variety if algebraic iterative reconstruction methods. It is also modularised in the sense that basic operators for reconstruction, *e.g.*, filtering, forward, projection and backprojection are separately available and implemented in multi-threaded CPU and GPU versions. There are furthermore tools for respiratory motion correction, routines for file I/O for several scanners, and routines for pre-processing of raw data for scatter correction. In this context one may also mention SNARK09 that provides a variety of algebraic iterative reconstruction methods for 2D tomography with fan beam and parallel beam geometry [20]. A more recent MATLAB package is AIR Tools that has several 2D algebraic iterative reconstruction methods for discretisations of inverse problems [13].

Finally, TVreg is a software package for TV image reconstruction written in C with MEX interface to MATLAB [15].

4 Optimisation software

This section considers primarily software libraries for general purpose large scale optimisation that are suitable for solving the variational problem in (2) in the context of medical imaging. Since (2) is a very-large scale non-linear optimisation problem, optimisation software that require storage of matrices with size $n \times m$ (n is the number of voxels, m is the number of data points), such as the matrix representing the Jacobian of the objective functional, are computationally unfeasible. On the other hand, storing one or two vectors of size n and m

is feasible, so in the context of optimisation, it is enough to require the action of the Jacobian to evaluate and store a gradient direction.

A number of software libraries for optimisation are listed in the Wikipedia entry for optimisation software, see also the listing at the NEOS Server, a free internet-based service for solving numerical optimisation problems. Below is an selection that may be useful for solving the variational problem in (2).

Galahad: The Galahad library [11] is a freely available thread-safe library for optimisation. The areas covered by the library are unconstrained and bound-constrained optimisation, quadratic programming, non-linear programming, systems of non-linear equations and inequalities, and non-linear least squares problems. The library is mostly written in the Fortran 90 programming language.

It is based on augmented Lagrangian methods which have recently turned out to be useful for TV-denoising and compressed sensing, see, *e.g.*, the SALSA and C-SALSA [1, 2] packages. The Galahad library also includes a filter-based method for systems of linear and non-linear equations and inequalities, an active-set method for non-convex quadratic programming, a primal-dual interior-point method for non-convex quadratic programming, a pre-solver for quadratic programs, a Lanczos method for trust-region subproblems, and an interior-point method to solve linear programs or separable convex programs or alternatively, to compute the analytic centre of a set defined by such constraints, if it exists.

SNOPT: Sparse non-linear OPTimizer (SNOPT) [10] is a commercial software written in Fortran, but interfaces to C and C++, as well as MATLAB are available.

SNOPT employs a sparse sequential quadratic programming algorithm with limited-memory quasi-Newton approximations to the Hessian of Lagrangian. It is especially effective for non-linear problems whose functions and gradients are expensive to evaluate. The functions should be smooth but need not be convex. SNOPT is part of a large collection of optimisation software libraries developed by UC San Diego and the Stanford Systems Optimization Laboratory at Stanford University.

KNitro: KNitro [6] is a commercial modern C-package for non-linear optimisation that implements interior (barrier) type method with trust regions. It provides a wide range of interfaces, including C, C++, Fortran, Java, Python, and MATLAB. KNitro is designed for large-scale problems, *e.g.*, it does not require a user to provide a matrix representation of the Jacobian of the objective functional.

OPT++: OPT++ [27] is a free library of non-linear optimisation algorithms written in C++. The aim is to provide a software framework for rapid prototyping and development of new optimisation algorithms. Specific focus is on robust and efficient algorithms for problems where function and constraint evaluations require the execution of an expensive computer simulation.

OPT++ distinguishes between an algorithm-independent class hierarchy for non-linear optimisation problems and a class hierarchy for non-linear optimisation methods that is based on common algorithmic traits. Currently, OPT++ includes the classic Newton methods, a nonlinear interior-point method, parallel direct search, generating set search, a trust region-parallel direct search hybrid, and a wrapper to NPSOL. Between these methods, a wide range of problems can be solved, *e.g.*, with or without constraints, with or without analytic gradients, simulation based, *e.t.c.*

YALMIP: YALMIP [21] is an free modelling language for advanced modeling and solution of convex and non-convex optimisation problems. It is implemented as a toolbox for MATLAB. The main motivation for using it is rapid algorithm development. The language is consistent with standard MATLAB syntax, thus making it extremely simple to use for anyone familiar with MATLAB. Another benefit of YALMIP is that it implements a large amount of modeling tricks, allowing the user to concentrate on the high-level model, while YALMIP takes care of the low-level modeling to obtain as efficient and numerically sound models as possible.

The modelling language supports a large number of optimisation classes, such as linear, quadratic, second order cone, semidefinite, mixed integer conic, geometric, local and global polynomial, multiparametric, bilevel and robust programming. YALMIP relies on external solvers for the actual computations. It does however have a low-level scripting language that enables one to solve sub-problems using the external solvers.

pyOpt: pyOpt [35] is an free object-oriented Python-based framework for formulating and solving nonlinear constrained optimisation problems in an efficient, reusable and portable manner. The framework uses object-oriented concepts, such as class inheritance and operator overloading, to maintain a distinct separation between the problem formulation and the optimisation approach used to solve the problem. This creates a common interface in a flexible environment where both practitioners and developers alike can solve their optimisation problems or develop and benchmark their own optimisation algorithms. The framework is developed in the Python programming language, which allows for easy integration of optimisation software programmed in Fortran, C/C++, and other languages. A variety of optimisation algorithms are integrated in pyOpt and are accessible through the common interface.

TOMLAB: TOMLAB [14] is a commercial optimisation platform and modelling language for solving optimisation problems in MATLAB. The overall objective is to provide the best modelling and optimisation tools for the MATLAB user and thereby enable a wide range of opportunities for large-scale robust optimisation for practically every area of optimisation.

The base module of TOMLAB supplies more than 70 different algorithms for linear, discrete, global and non-linear optimisation. All routines are call-compatible with the MATLAB Optimisation Toolbox and there are additional add-on modules that provide connection to a wide range of external solvers. These are distributed as compiled binary MEX DLLs on Windows-systems, and compiled MEX library files on Unix and other systems.

TFOCS: TFOCS [4] is a set of MATLAB templates, or building blocks, that can be used to construct efficient, customised solvers for a variety of convex models, including in particular those employed in sparse recovery applications.

Other: A number of software packages for reconstruction/regularisation also contain sophisticated optimisation schemes. The Department of Computational and Applied Mathematics at Rice University lists a number of optimisation software related to sparsity promoting regularisation, see also the Rice Compressive Sensing Resources maintained by the Digital Signal Processing group at Rice University. A similar list of various solvers for sparsity promoting regularisation is provided in the Big Picture in Compressive Sensing page and the implementation entries in the Nuit Blanche blog.

Most of the software packages mentioned in the lists above are however not suitable for large-scale problems and/or have too uncertain developer support. A couple are however worth taking a close look at. One is SPGL1 [40], a MATLAB solver for large-scale ℓ_1-norm regularised least squares. Another is NESTA [3], a fast and robust first-order method than solves basis-pursuit problems and a large number of extensions (including TV-regularisation). The SPArse modelling Software [26, 25] and SparseLab also have several optimisation routines for solving various sparsity promoting variational regularisation problems, even though they are mostly for smaller problems than 3D tomography reconstruction. A nice survey of GPU accelerated implementation of several greedy algorithms for ℓ_1 regularisation is provided in [5].

Finally, optimisation for minimising entropy type of functionals is implemented in the MomEnt+ software package and in the MemSys5 and MitSys software packages developed by Maximum Entropy Data Consultants Ltd. A modified and more up-to-date approach making use of entropy to penalise complexity in image reconstruction is provided by the Pixon Software [36].

Various optimisation methods have also been compared against each other to asses their performance in the context of regularisation. One such systematic comparison is the L1TestPack [22, 23] that consists of several MATLAB files to generate test instances for the so-called Basis Pursuit Denoising problem and to check several conditions related to these problems. Another comparison is provided by L1General, which is a set of MATLAB routines implementing several of the available strategies for solving ℓ_1-regularisation. The software only assumes that the user can provide a 'black box' function that returns values of the functional and its gradient for a given parameter setting. The purpose of the software is to compare the performance of different optimisation strategies in this black box setting. Although treating the functionals as a black box means that the codes cannot take full take advantage of the structure of the objective functional, this perspective makes it very easy to use the codes to solve a variety of different ℓ_1-regularisation problems.

5 Regularisation library

5.1 Overall design goals

The platform for variational regularisation we seek should consist of software components that are self-contained in that they hide implementation details from the user (encapsulation). The platform should be modular and flexible in that it should be relatively easy to add new; algorithms, data acquisition protocols, pre-processing schemes, forward models, and image discretisation schemes. Ideally, it should be possible to select at run-time which version of these components you want to use. The advantages of such a platform are (a) modularity and flexibility of the reconstruction building blocks to implement new reconstruction algorithms, (b) possibility to compare analytic and iterative methods within a common framework, (c) the possibility to use the same software implementation of the building blocks to perform image reconstruction on different scanner geometries and (d) independence of the computer platform on which the software runs. Another aspect is that the platform should contain software components to run parts of the reconstruction in parallel on distributed memory architectures, although these are not distributed yet. This will enable the software to be run not only on single processors, but also on massively parallel computers, or on clusters of workstations. Finally, it should be portable on all systems supporting the GNU C/C++ compiler or MS Visual C/C++ (or hopefully any ANSI C/C++ compliant compiler).

Now, a key part is to appropriately capture the abstraction layers provided by mathematics. More precisely, images are represented by real valued functions, so the reconstruction space $\mathscr{X}$ in (1) is typically some infinite dimensional vector space and the variational problem in (2) is an optimisation over infinite dimensional vector spaces. A natural *design goal* for a software platform for variational regularisation is therefore to provide computational types realising the principal concepts of calculus in (infinite dimensional) vector spaces. In particular it should contain the following:

1. Components that encode core concepts of calculus in Hilbert space (vector, function, functional, ...) with minimal implementation dependence.

2. Components that encode the notion of discretisation, *i.e.*, how elements in an infinite dimensional vector space are repented in a finite dimensional setting.

3. Standardised interfaces behind which to hide application-dependent implementation details (data containers, function objects).

Given the above, a variety of coordinate-free algorithms from linear algebra and optimisation may then be expressed purely in terms of this system of components, resulting in a code that can be used without alteration in a wide range of imaging applications, and in serial and parallel computing environments.

5.2 Relation to external software libraries

Evaluation of the discretised forward model corresponds to simulating the imaging process. This involves a variety of computational types and data structures specific for the physical modelling and numerical implementation. Next, it is often desirable to use optimisation methods that make use of gradient information in solving (2). Hence, one also needs to evaluate the adjoint of its (Fréchet) derivative of the forward model.

Due to the problem size in tomography, the evaluation of the discretised forward model $\boldsymbol{\mathcal{T}}\colon \mathscr{X} \to Y$ and the adjoint of its derivative will involve large scale calculations, so a variety of high performance computing techniques are commonly employed. Next, such software libraries also presuppose a specific discretisation of both the image (*i.e.*, element in $\mathscr{X}$) and data (*i.e.*, element in $\mathscr{Y}$), which are given by the operators $P_{\text{image}}\colon \mathscr{X} \to X \subset \mathbb{R}^n$ and $P_{\text{data}}\colon \mathscr{Y} \to Y \subset \mathbb{R}^m$, respectively. These can be data structures for representing arrangement of lines, geometric meshes or grids and rules for their construction and refinement, functions on these grids representing physical fields, equations relating grid functions and embodying (discretised versions of) physical laws, and iterative or recursive algorithms which produce solutions of these equations. Hence, we can expect forward model softwares to evolve over time. Next, modelling choices considered reasonable today may likely give way in the future to other systems incorporating more complete physics, so one may expect an evolution of forward model software. Likewise, new source-detector arrangements gives rise to new data acquisition protocols, so the representation of the digitisation of data, *i.e.*, the implementation of P_{data}, will have to be evolvable.

Optimisation and linear algebra algorithms, on the other hand, generally have no intrinsic interaction with physics and its numerical realisation, involving instead a more abstract layer of mathematical constructs: vectors, functions, gradients, ... Many of these algorithms, including some of the most effective ones for large-scale problems, may be expressed without explicit reference to coordinates. These coordinate-free (sometimes called matrix-free) algorithms use only the intrinsic operations of linear algebra and calculus in Hilbert space. Examples include Krylov subspace methods for the solution of linear systems and eigenvalue problems, Newton and quasi-Newton methods for unconstrained optimisation, and many constrained optimisation methods. These algorithms also gain complexity as their effectiveness increases, especially with regard to the incorporation of model constraints. Thus, it is desirable to protect the investment in effective optimisation algorithms.

In summary, both the forward model software and optimisation software may be quite complex and involve a considerable investment of time and software development effort. These software components may also be developed by different groups of experts. Consequently, incorporating them into a variational regularisation scheme (2) for solving a particular inverse problem may require extensive modification of both types of software components. Therefore, a good design for a software platform for variational regularisation needs to considerably ease this task. Ideally, such a platform would links to virtually unmodified forward model and optimisation software packages, via a generic middleware package and a minimal amount of additional code.

5.3 Programming paradigms

The aforementioned discrepancy between levels of abstraction that may arise when using forward model and optimisation software is the source of a software engineering problem: in procedural programs for solving (2), the details of structure for evaluating the forward model invariably intrude on the optimisation code, and vice versa. Time-honoured software 'tricks' used to hide these details within procedural code (common blocks, parameter arrays, '`void*`' parameters, reverse communication, . . .) lead to software that is difficult to debug and maintain and nearly impossible to modify, extend, or reuse outside of the originating context. One way out of this dilemma is to employ data abstraction and polymorphism.

Data abstraction defines data objects in terms of the operations used to manipulate them, rather than in terms of their composition out of simpler objects. This device allows implementation details in one part of a program to be hidden completely from other parts which do not intrinsically involve them. Without data abstraction, the algorithm must be written to reference the entire data structure; the implementation becomes dependent on data which is foreign to the algorithm, and cannot be used without modification in a different context. An abstract data array type can expose only those details needed in the context of a specific algorithm or class of algorithms, via a set of operations on these details, and hide other details entirely. Thus data abstraction frees the algorithm writer from the necessity to refer explicitly to all aspects of a data structure.

Polymorphism complements data abstraction to enable reuse of numerical algorithms across many applications. Polymorphic functions accept a variety of argument types, and perform operations dictated by the type of their arguments. This concept is familiar: for example, the '+' operator in Fortran 77 accepts arguments of all arithmetic types and performs the appropriate form of addition for each. Abstract polymorphic functions take this concept a step further: they accept arguments of abstract data types, and rely for their definitions only on the attributes of these types. A polymorphic linear operator type defines the matrix-vector product as an input-output operation on an abstract vector type, and can be realised concretely using either type of definition, accessing the data of the vector arguments as is appropriate. An abstract polymorphic linear operator type does not specify the means by which the matrix-vector product is carried out: it merely provides a promise that it is done. An algorithm written in terms of an abstract polymorphic linear operator type can thus be used in contexts involving either 'implicit' or ordinary, explicit matrices, without recoding: the specific instance of the type need supply only the computations mandated by the type, without any reference to the explicit computations by which they are carried out.

Using an object oriented programming paradigm allows one to ensure software components are properly encapsulated. Specialisation of concepts is implemented with hierarchies of classes (inheritance) and conceptually identical operations are implemented using functions with identical names (polymorphism). In the context of tomographic reconstruction, natural building block classes for an object oriented library are; information about the data (scanner characteristics, study type, algorithm type, *e.t.c.*); multi-dimensional arrays (any dimension) with various operations, including numeric manipulations; reading and writing (I/O) data in relevant formats; classes of projection data (complete data set, segments, sinograms, viewgrams) and images (2D/3D); various filter transfer functions (1D, 2D and 3D); operators representing the forward model and the adjoint of its derivative; classes for sparse projection matrices, both for on-the-fly computation and pre-stored; trimming/zooming utilities on projection and image data; classes for data pre-processing (scatter estimation, normalisation and attenuation correction); classes for iterative and variational reconstruction algorithms; classes for temporal modelling; and stream-based classes for message passing between different processes, built on top of MPI.

5.4 Data structures representing data digitisation

One part is to read the relevant data formats (section 2.2), another is to correctly represent the data acquisition protocol to properly encode the data.

In tomographic inverse problems, data is mathematically represented as a discretised function defined on a set of lines in $\mathbb{R}^2$ or $\mathbb{R}^3$ for the stationary case, temporal variation adds another dimension. Thus, one must first provide a mathematical description of the sets (manifolds) of lines in $\mathbb{R}^3$ that correspond to actual data acquisition protocols arising in the tomographic imaging applications. For CT, one may consult [7, Chapter 7], [38, Chapter 2 and Appendix A], [43, chapter 5], [28, section 3.1], and [30, chapter 4] to get an idea of how to mathematically describe the set on lines that represent the data acquisition protocol. Similar descriptions are also provided in papers dealing with the development of FBP type of inversion formulas, *e.g.*, [17, 32, 44, 18, 42, 41, 19, 24]. As a final remark, Siemens has a specific acquisition scheme, flying focal spot (FFS) data acquisition, for its CT machines that is described in [16]. In emission tomography, the data acquisition geometries depend on the actual arrangement of detectors in PET and SPECT machines. This information can also be difficult to obtain without support from the vendor.

Another issue concerns the file format for the data, which is often proprietary and vendor specific. Hence, here we must have access to a format specification and/or vendor supplied software for reading the data. In emission tomography there are some formats that have been made public, *e.g.*, the Interfile format (for which a 3D PET extension is proposed), the GE Advance sinogram data format, and the ECAT6 and ECAT7 formats.

5.5 Existing general software platforms for reconstruction

There are no software platforms for variational regularisation that fulfil the design criteria listed in the previous sections. There are however some software libraries that can be used to design such a software platform.

5.5.1 The Rice vector library

The Rice vector library (RVL) [29, 39] is a C++ framework for functional analysis in the context of solving large-scale simulation-driven optimisation problems arising in science and engineering. It defines computational types realising the principal concepts of calculus in vector spaces, in the form of C++ classes, and links the specific data structures of a forward model to an abstract numerics library, thereby acting as a 'middle-ware' translation layer. RVL consists of several sub-packages briefly described below, see the class documentation and the above cited publications for a more complete description.

RVL (base): Fundamental interfaces defining calculus in Hilbert space: vector spaces, vectors, linear and non-linear scalar- and vector-valued functions, and closely related concepts such as product (block data, operator) decompositions and abstract quality control tests for derivatives, adjoints, *e.t.c.*

Also data management types (data containers, function objects) – uniform abstract interfaces to concrete data structures and implementations.

LocalRVL: A simple realisation of the data management hierarchy for containers that expose a pointer to a contiguous block of memory (C or Fortran arrays). Useful both in itself and as building block for more complex in-core, out-of-core, and distributed data management schemes. Includes various utilities.

MPIRVL: MPI wrappers for the LocalRVL classes – makes them available in various ways in distributed applications.

Algorithm: An algorithm is a von Neumann machine, that is, you start it and it runs until it stops. This sub-package contains simple abstractions for Algorithms and Terminators (objects equipped to give an answer, 'yes' or 'no' on various grounds, to the question, 'should this algorithm continue?'). Various standard combinations build loops, lists, decision trees *e.t.c.* and give an elegant mechanism for expressing virtually any iterative algorithm.

Umin: Unconstrained minimisation and iterative linear algebra algorithms, realised as `RVL::Algorithms`. Quasi-Newton, trust region, Krylov subspace methods for optimisation, linear systems, and eigenvalue problems.

Sequence: Example showing how infinite dimensional vector spaces can actually be implemented in RVL, implying that dimension cannot be a general attribute of an RVL vector space. Defines space of finite sequences, a linear operator on them (polynomial multiplication), and approximates the solution of a least squares problem.

Time stepping for optimisation: Time stepping methods encapsulated as `RVL::Operator` interfaces, in terms of natural component interfaces. Groups forward, linearised, and adjoint iterations together as coherent object. Several schemes for random access to simulation history to support adjoint state method, including implementations of of Griewank's optimal checkpointing algorithm for both fixed and adaptive time steps.

Tests: A small set of unit tests to verify correct installation of RVL and LocalRVL.

5.5.2 Spot – A Linear-Operator Toolbox

SPOT is a MATLAB toolbox that seeks to bring the expressiveness of MATLAB's built-in matrix notation to problems where explicit matrices are not practical, *e.g.*, due to problem size. The software introduces the notion of a Spot operator that represents a matrix, and can be treated in a similar way, but it doesn't rely on the matrix itself to implement most of the methods.

5.5.3 iLang

iLang [34] is a Python module for Bayesian inference designed for imaging applications. It is essentially an imaging inference language that enables probabilistic reasoning in volumetric imaging, simplifying the definition of complex imaging models. The probabilistic models are represented by graphs, which enables the automated synthesis of efficient inference algorithms. The inference engine of iLang enables the definition of non-smooth constraints such as non-negativity and sparsity. The iLang software constitutes a unified framework for multi-modal imaging that enables the integration of image formation, registration, de-noising and other image processing tasks. It has been used for 4D PET with motion correction in which the motion is considered a nuisance variable and estimated from the PET emission data under the assumption of sparsity.

5.5.4 Other

In [38, Chapter 3] there is a description of a software architecture for reconstruction in medical X-ray imaging that is modular in a sense that different accelerator hardware platforms are supported. Here, one can implement different parts of the algorithm using different acceleration architectures and techniques. This enables us to take advantage of the parallelism in off-the-shelf accelerator hardware such as multi-core systems, the Cell processor, and graphics accelerators in a very flexible and reusable way.

References

[1] M. Afonso, J. Bioucas-Dias, and M. Figueiredo. Fast image recovery using variable splitting and constrained optimization. *IEEE Transactions on Image Processing*, 10(9):2345–2356, 2010.

[2] M. Afonso, J. Bioucas-Dias, and M. Figueiredo. An augmented Lagrangian approach to the constrained optimization formulation of imaging inverse problems. *IEEE Transactions on Image Processing*, 20(3):681–695, 2011.

[3] S. Becker, J. Bobin, and E. J. Candès. NESTA: A fast and accurate first-order method for sparse recovery. *SIAM Journal on Imaging Sciences*, 4(1):1–39, 2011.

[4] S. Becker, E. J. Candès, and M. Grant. Templates for convex cone problems with applications to sparse signal recovery. *Mathematical Programming Computation*, 3(3):165–218, 2011.

[5] J. Blanchard and J. Tanner. GPU accelerated greedy algorithms for compressed sensing. *Mathematical Programming Computation*, 5:267–304, 2013.

[6] R. H. Byrd, J. Nocedal, and R. A. Waltz. KNITRO: An integrated package for nonlinear optimization. In G. di Pillo and M. Roma, editors, *Large-Scale Nonlinear Optimization*, pages 35–59. Springer-Verlag, 2006.

[7] R. Cierniak. *X-Ray Computed Tomography in Biomedical Engineering*. Springer-Verlag, 2011.

[8] J. A. Fessler. ASPIRE 3.0 user's guide: A sparse iterative reconstruction library. Technical Report 293, Department of Electrical Engineering and Computer Science, The University of Michigan, 2009.

[9] J. A. Fessler. Users guide for ASPIRE 3D image reconstruction software. Technical Report 310, Department of Electrical Engineering and Computer Science, The University of Michigan, 2013.

[10] P. E. Gill, W. Murray, and M. A. Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Rview*, 47(1):99–131, 2005.

[11] N. I. M. Gould, D. Orban, and P. L. Toint. GALAHAD, a library of thread-safe Fortran 90 packages for large-scale nonlinear optimization. *ACM Transactions on Mathematical Software*, 29(4):353–372, 2004.

[12] 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):1188–1193, 2014.

[13] P. C. Hansen and M. Saxild-Hansen. AIR tools - A MATLAB package of algebraic iterative reconstruction methods. *Journal of Computational and Applied Mathematics*, 236(8):2167–2178, 2012.

[14] K. Holmström, M. M. Edvall, and A. O. Göran. TOMLAB – for large-scale robust optimization. In *Proceedings, Nordic MATLAB Conference 2003, October 21, 2003*, 2003.

[15] 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*, 52(2):329–356, 2011.

[16] M. Kachelriess, M. Knaup, C. Penssel, and W. A. Kalender. Flying focal spot (FFS) in cone-beam CT. *IEEE Transactions on Nuclear Science*, 53(3):1238–1247, 2006.

[17] A. Katsevich. A general scheme for constructing inversion algorithms for cone beam CT. *International Journal of Mathematics and Mathematical Sciences*, 2003(21):1305–1321, 2003.

[18] A. Katsevich. 3pi algorithms for helical computer tomography. *Advances in Applied Mathematics*, 36(3):213–250, 2006.

[19] A. Katsevich, A. A. Zamyatin, and M. D. Silver. Optimized reconstruction algorithm for helical CT with fractional pitch between 1PI and 3PI. *IEEE Transactions on Medical Imaging*, 28(7):982–990, 2009.

[20] J. Klukowska, R. Davidi, and G. T. Herman. SNARK09 – a software package for reconstruction of 2D images from 1D projections. *Computer Methods and Programs in Biomedicine*, 110:424–440, 2013.

[21] J. Löfberg. YALMIP: A toolbox for modeling and optimization in MATLAB. In *IEEE International Symposium on Computer Aided Control Systems Design, 2004*, pages 284–289, 2004.

[22] D. A. Lorenz. Constructing test instances for basis pursuit denoising. *IEEE Transactions in Signal Processing*, 61(5):1210–1214, 2012.

[23] D. A. Lorenz, M. E. Pfetsch, and A. M. Tillmann. Solving basis pursuit: Heuristic optimality check and solver comparison. Technical Report Preprint, Optimization Online, 2013.

[24] Y. Lu, A. Katsevich, J. Zhao, H. Yu, and G. Wang. Fast exact/quasi-exact FBP algorithms for triple-source helical cone-beam CT. *IEEE Transactions on Medical Imaging*, 29(3):756–770, 2010.

[25] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 689–696, 2009.

[26] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research*, 11:19–60, 2010.

[27] J. C. Meza, R. A. Oliva, P. D. Hough, and P. J. Williams. OPT++: An object-oriented toolkit for nonlinear optimization. *ACM Transactions on Mathematical Software*, 33(2), 2007.

[28] F. Natterer and F. Wübbeling. *Mathematical methods in image reconstruction*, volume 5 of *SIAM monographs on mathematical modeling and computation*. SIAM, 2001.

[29] A. D. Padula, D. S. Shannon, and W. W. Symes. A software framework for abstract expression of coordinate-free linear algebra and optimization algorithms. *ACM Transactions on Mathematical Software*, 36(2):8:1–8:36, 2009.

[30] V. Palamodov. *Reconstructive Integral Geometry*, volume 98 of *Monographs in Mathematics*. Springer-Verlag, 2004.

[31] W. J. Palenstijn, K. J. Batenburg, and J. Sijbers. Performance improvements for iterative electron tomography reconstruction using graphics processing units (GPUs). *Journal of Structural Biology*, 176(2):250–253, 2011.

[32] X. Pan, D. Xia, Y. Zou, and L. Yu. A unified analysis of FBP-based algorithms in helical cone-beam and circular cone- and fan-beam scans. *Physics in Medicine and Biology*, 49(18):4349–4369, 2004.

[33] S. Pedemonte, A. Bousse, K. Erlandsson, M. Modat, S. Arridge, B. F. Hutton, and S. Ourselin. GPU accelerated rotation-based emission tomography reconstruction. In *2010 IEEE Nuclear Science Symposium Conference Record (NSS/MIC)*, pages 2657–2661, 2010.

[34] S. Pedemonte, C. Catana, and K. Van Leemput. An inference language for imaging. In M. J. Cardoso, I. Simpson, T. Arbel, D. Precup, and A. Ribbens, editors, *Bayesian and grAphical Models for Biomedical Imaging*, volume 8677 of *Lecture Notes in Computer Science*, pages 61–72. Springer-Verlag, 2014.

[35] R. E. Perez, P. W. Jansen, and J. R. R. A. Martins. pyOpt: a Python-based object-oriented framework for nonlinear constrained optimization. *Structural and Multidisciplinary Optimization*, 45(1):101–118, 2012.

[36] R. C. Puetter, T. R. Gosnell, and A. Yahil. Digital image reconstruction: Deblurring and denoising. *Annual Review of Astronomy and Astrophysics*, 43:139–194, 2005.

[37] S. Rit, M. Vila Oliva, S. Brousmiche, R. Labarbe, D. Sarrut, and G. C. Sharp. The Reconstruction Toolkit (RTK), an open-source cone-beam CT reconstruction toolkit based on the Insight Toolkit (ITK). *Journal of Physics: Conference Series*, 489(1):012079, 2014.

[38] H. Scherl. *Evaluation of State-of-the-Art Hardware Architectures for Fast Cone-Beam CT Reconstruction.* Vieweg+Teubner Verlag, 2011.

[39] W. W. Symes, D. Sun, and M. Enriquez. From modelling to inversion: designing a well-adapted simulator. *Geophysical Prospecting*, 59:814—833, 2011.

[40] E. van den Berg and M. P. Friedlander. Sparse optimization with least-squares constraints. *SIAM Journal on Optimization*, 21(4):1201—1229, 2011.

[41] D. Wang, Y. Ye, and H. Yu. Approximate and exact cone-beam reconstruction with standard and non-standard spiral scanning. *Physics in Medicine and Biology*, 52(6):R1–R13, 2007.

[42] H. Yu, Y. Ye, S. Zhao, and G. Wang. Studies on Palamodov's algorithm for cone-beam CT along a general curve. *Inverse Problems*, 22(2):447–460, 2006.

[43] L. Zeng. *Medical Image Reconstruction. A Conceptual Tutorial.* Springer-Verlag, 2010.

[44] J. Zhu, S. Zhao, H. Yu, Y. Ye, S. W. Lee, and G. Wang. Numerical studies on Feldkamp-type and Katsevich-type algorithms for cone-beam scanning along nonstandard spirals. In U. Bonse, editor, *Developments in X-Ray Tomography IV, (26 October 2004)*, volume 5535 of *Proceedings of SPIE*, page 8 p., 2004.