Les Cahiers du GERAD ISSN: 0711–2440

Factorization-free methods for computed tomography

Y. Goussard, M. McLaughlin,
D. Orban

G–2017–65

August 2017

Factorization-free methods for computed tomography

Yves Goussard [a]

Maxime McLaughlin [b]

Dominique Orban [b]

[a] *Department of Biomedical Engineering, Polytechnique Montréal (Québec) Canada, H3C 3A7*

[b] *GERAD & Department of Mathematics and Industrial Engineering, Polytechnique Montréal (Québec) Canada, H3C 3A7 other example*

yves.goussard@polymtl.ca
maxime.mclaughlin@polymtl.ca
dominique.orban@gerad.ca

August 2017

Les Cahiers du GERAD
G–2017–65

Abstract: We study X-ray tomograqphic reconstruction using statistical methods. The problem is expressed in cylindrical coordinates, which yield significant computational and memory savings, with nonnegativity bounds. A change of variables involving a Fourier matrix attempts to improve the conditioning of the Hessian but introduces linear inequality constraints. The scale and density of the problem call for factorization-free methods. We argue that projections into the feasible set can be computed efficiently by solving a bound-constrained linear least-squares problem with a fast operator. This motivates our interest towards projection-based active-set methods for the reconstruction problem, namely a spectral projected gradient method and a trust-region projected Newton method that we generalize to our specific scenario. For the projection subproblem, we consider several projection-based methods for bound-constrained problems. We assess the performance of several algorithm combinations on the reconstruction problem using synthetic data. Our results show that the projected Newton method combined with efficient projection strategies applied to the problem in cylindrical coordinates with linear inequality constraints is competitive in terms of run time with a limited-memory BFGS applied to the problem in cartesian coordinates with simple bounds, but reduces the memory footprint by a factor of about 233 on a 2D problem with 512×512 pixels.

Keywords: Factorization-free, convex constrained optimization, tomographic reconstruction, medical imaging

1 Introduction

Generally speaking, tomographic reconstruction methods fall within two categories: *analytical* techniques, which rely on strong approximations to the data formation model, and *statistical* or *algebraic* techniques, which make use of an estimation methodology. Since the inception of X-ray tomography, analytical methods have been prevalent in clinical settings, mainly because of their limited computational requirements. However, their limitations in terms of accuracy has been recognized early (Herman and Rowland, 1973), and statistical methods have been repeatedly shown to produce superior results in many respects (Pan et al., 2009; Beister et al., 2012). Nevertheless, practical use of statistical methods has remained limited, mostly because of high computation times and large memory requirements. Recently, statistical approaches have been the subject of renewed interest due to the increase in computer performance, pressure toward X-ray dose reduction and development of new types of X-ray scanners, even though computation time and memory footprint remain major difficulties.

An avenue to tackle these difficulties is to make use of nonstandard representations that can take advantage of redundancies in the data collection process. Among them, formulation of the problem in cylindrical coordinates (Thibaudeau et al., 2013; Goussard et al., 2013) has been shown to produce considerable reduction of memory footprint without on the fly computation of the projection matrix, and significant acceleration of the computation through straightforward parallelization. However, expressing the reconstruction problem in cylindrical coordinates induces substantial ill conditioning, but the latter can be alleviated by appropriate scaling. The main focus of our work is to investigate the various algorithms that can solve the scaled problem and account for the nonnegativity constraints that the solution must satisfy.

Our paper is organized as follows. In Section 2, we describe how the reconstruction problem is obtained from maximum a posteriori estimation. Its key features are that it is a large-scale regularized linear least-squares problem with linear inequality constraints. To tackle this optimization problem, we consider projection-based factorization-free active-set methods. We outline generic active-set method as well as the various projection operations that are required by our algorithms in Section 3 and detail the methods that we use in Section 4. Because some of the projection operations are cast as other optimization problems, we address their solution in Section 5. We report the results of our reconstruction algorithm on synthetic data in Section 6, and we study its behavior according to the choice of reconstruction solver, projection solver and various parameters. Conclusions, as well as further improvements, appear in Section 7.

Implementations of our solvers are available in object-oriented MATLAB as part of the NLPLab optimization framework available at https://bitbucket.org/maxmcl/nlplab.

2 Iterative reconstruction algorithm

X-ray tomography is an imaging modality based on the measurement of X-ray attenuation trough an unknown object under several incidences. The goal of reconstruction is to recover the spatial distribution of linear X-ray attenuation coefficients in the object from the measurements and a model of the X-ray attenuation process—the data formation model. Typically, such models are based upon the Beer-Lambert law. Here, we consider the *stochastic* version of the Beer-Lambert law, and we show how probability estimation can be used to perform the reconstruction.

2.1 Stochastic Beer-Lambert law and discretization

In tomographic reconstruction, the Beer-Lambert law relates the attenuation of an energy beam, that travels through an object, with a distribution of attenuation coefficients $\mu(x) : \mathbb{R}^{n_{\text{dim}}} \to \mathbb{R}$, where n_{dim} is the number of spatial dimensions (typically 1, 2 or 3). We assume that the uncertainty on the transmitted intensities is dominated by the quantum effects related to the attenuation of the X-ray beams by matter. Under that hypothesis, the photon counts collected by the detectors can be modeled as a Poisson distribution $\mathcal{P}(l)$, where l is the parameter of the distribution. We introduce the random variable N representing the photon counts of realization vector $n \in \mathbb{R}^{n_{\text{meas}}}$, where n_{meas} is the number of intensity measurements. Furthermore, we assume both the attenuation coefficients and the source to be energy *independent*. Consequently, the *monochromatic* stochastic Beer-Lambert law that will be considered hereafter is

$$N \sim \mathcal{P}\left(n_0 \, \mathrm{e}^{-\int_{L_i} \mu(x) \, \mathrm{d}x}\right), \tag{1}$$

where $n_0 \in \mathbb{R}$ is the peak energy of the source, L_i represents a linear path through the patient $\mu(x)$ and n_i an intensity measurement collected by a detector.

The *sinogram*, often called *projections* (not to be confused with mathematical projections) or *projection data*, is

$$y := \ln\left(\frac{n_0}{n}\right), \tag{2}$$

where the logarithm and division occur componentwise.

In practice, n_{meas} is determined by the angular discretization of the rotation of the source and the detectors, as well as the total number of detectors. Assuming three-dimensional (3D) reconstruction, the subscript $i = 1, \ldots, n_{\text{meas}}$ denotes an i-th measurement, obtained at an i-th scan angle, detector and axial position.

In order to obtain an expression of Equation (1) that is suited to numerical methods, the domain of μ must be discretized in n_{vox} voxels, such that μ_j, the j-th component of μ, is assigned to the j-th voxel. This is done by the means of a discretization function $\xi(x)$ that can be interpreted as an n_{dim}-dimensional mesh. Hence, μ becomes independent of x and can be excluded from the integrand of Equation (1), which becomes a collection of ray-voxel intersection lengths along L_i,

$$\int_{L_i} \mu(x) \, \mathrm{d}x \int_{L_i} \sum_j \mu_j \xi_j(x) \, \mathrm{d}x = \sum_j \mu_j \int_{L_i} \xi_j(x) \, \mathrm{d}x = \sum_j p_{ij}\mu_j,$$

where we define

$$p_{ij} := \int_{L_i} \xi_j(x) \, \mathrm{d}x.$$

We call $\mu \in \mathbb{R}^{n_{\text{vox}}}$ the vector of attenuation coefficients, and $P \in \mathbb{R}^{n_{\text{meas}} \times n_{\text{vox}}}$ the *projection matrix* that contains the intersection lengths. We may now rewrite Equation (1) as

$$N \sim \mathcal{P}\left(n_0 \, \mathrm{e}^{-P\mu}\right). \tag{3}$$

Assuming independent and identically distributed (i.i.d.) photon counts, Equation (3) has the conditional probability density function

$$P(N = n \mid \mu) = \prod_i \left[\frac{\exp\left(-n_0 \, \mathrm{e}^{-[P\mu]_i}\right)\left(n_0 \, \mathrm{e}^{-[P\mu]_i}\right)^{n_i}}{n_i!}\right], \tag{4}$$

where $[P\mu]_i$ designates the i-th component of $P\mu$. Hence, using this expression, we can evaluate the probability of measuring the intensities n given μ. Naturally, we will seek the distribution that is the *most likely* according to our data, which corresponds to *maximum likelihood* estimation.

2.2 Maximum likelihood

The *maximum likelihood* (ML) estimator of the conditional probability distribution Equation (4) is

$$\hat{\mu}_{\text{ML}} = \operatorname{argmax} P(N = n \mid \mu) \text{ subject to } \mu \geq 0, \tag{5}$$

where we impose the physical constraint $\mu \geq 0$. Instead of maximizing Equation (4), we may equivalently *minimize* the negative log-likelihood,

$$L(n \mid \mu) = \sum_i \left[n_0 \, \mathrm{e}^{-[P\mu]_i} + n_i[P\mu]_i + \log(n_i!)\right]. \tag{6}$$

Minimizing Equation (6) under nonnegativity constraints remains difficult given its nonlinear nature and compels us to consider different approaches. Sauer and Bouman (1993) circumvent this issue by applying a second-order Taylor expansion to Equation (6), where each term is expanded about y, so that, after dropping terms that do not depend on μ, the objective function reduces to a least-squares residual

$$L(n \mid \mu) \approx \tfrac{1}{2} \|P\mu - y\|^2_{\Delta_N} \tag{7}$$

where y is defined in (2) and $\Delta_N = \operatorname{diag}(n_i)_{i=1,\, \dots,\, n_{\text{meas}}}$ acts as a weighing matrix, where greater penalties are assigned to higher values of n_i. Indeed, since they correspond to less attenuated beams, they have higher signal-to-noise ratio and thus less uncertainty. To further simplify our model, we set $\Delta_N = I$. Finally, by substituting Equation (7) into Equation (5), the maximum likelihood estimator becomes:

$$\hat{\mu}_{\text{ML}} = \operatorname{argmin} \tfrac{1}{2} \|P\mu - y\|^2 \ \text{ subject to } \mu \geq 0. \tag{8}$$

One might note that Equation (8) can also be obtained by assuming that the measured photon counts follow a normal distribution. Moreover, since we know that μ describes biological tissues, we might expect the reconstructed μ to follow certain behaviors. Exploiting this *prior knowledge* is possible through *maximum a posteriori* estimation.

2.3 Maximum a posteriori and penalty function

Bayes's theorem lets us introduce a prior distribution $P(\mu)$ that reflects our knowledge on μ. The *maximum a posteriori* (MAP) estimate is the most probable value of $P(\mu \mid n)$ taking the physical constraint $\mu \geq 0$ into account, i.e.,

$$\hat{\mu}_{\text{MAP}} \in \operatorname{argmax} \frac{P(n \mid \mu)\, P(\mu)}{P(n)} \ \text{ subject to } \mu \geq 0. \tag{9}$$

Instead of maximizing Equation (9), we once again minimize the negative of its logarithm, so that assuming an exponential prior

$$P(\mu) \propto \mathrm{e}^{-\lambda \phi(\mu)} \quad (\lambda > 0),$$

amounts to replacing Equation (8) with the penalized problem

$$\underset{\mu}{\text{minimize}} \ f(\mu) \ \text{ subject to } \mu \geq 0, \qquad f(\mu) := \tfrac{1}{2} \|P\mu - y\|^2 + \lambda \phi(\mu). \tag{10}$$

Hence, under our assumptions and approximations, the reconstruction of an image can be cast as the solution of the bound-constrained regularized linear least-squares problem (10).

We favor penalty functions that preserve convexity of the objective of Equation (10) and that *smoothen* the attenuation coefficients, i.e. that penalize strong local variations. To this end, Goussard et al. (2013) employ $\mathscr{L}_2$ or $\mathscr{L}_2\mathscr{L}_1$ penalty functions, either directly on μ or on the difference between neighboring voxels. If the $\mathscr{L}_2$ norm is used, we refer to the first case as a *penalty on the object*, and to the second as a *penalty on the gradient of the object*.

The $\mathscr{L}_2$-penalty can be written

$$\phi_{\mathscr{L}_2}(\mu) = \tfrac{1}{2} \sum_{n=1}^{n_{\text{dim}}} \mu^{\mathsf{T}} {D^{(n)}}^{\mathsf{T}} \Gamma^{(n)} D^{(n)} \mu,$$

where $\Gamma^{(n)}$, $n = 1, \dots, n_{\text{dim}}$, are diagonal weight matrices corresponding to volume elements for each voxel, i.e. they take into account the variable size of each voxel, and $D^{(n)}$, $n = 1, \dots, n_{\text{dim}}$, are the identity if the penalty applies to the object, or first-derivative matrices if the penalty applies to the gradient of the object.

Note that the superscript (n) indicates the n-th matrix and not powers. The $\mathscr{L}_2\mathscr{L}_1$-penalty can be written

$$\phi_{\mathscr{L}_2\mathscr{L}_1}(\mu) = \sum_{n=1}^{n_{\text{dim}}} e^{\mathsf{T}} \Gamma^{(n)} \left(\delta^2 e + (D^{(n)} \mu)^2 \right)^{1/2}$$

where δ is a nonzero real parameter, e is the vector of ones, and the square and square root are applied componentwise to vectors.

2.4 Scaled problem in cylindrical coordinates

As indicated in Section 1, Thibaudeau et al. (2013) and Goussard et al. (2013) achieved large gains in memory requirements and reconstruction time by discretizing μ in cylindrical coordinates. Cylindrical coordinates allow us to benefit from geometric redondances in the data acquisition process, which translate into a *block-circulant* structure for the projection matrix. Hence, only a single row of blocks of P needs to be stored, which greatly reduces the memory footprint (generally by a factor of several hundreds with respect to storage of the full projection matrix in Cartesian coordinates).

Unfortunately, Goussard et al. (2013) and Golkar (2013) show that state-of-the-art methods, such as L-BFGS-B (Byrd et al., 1995; Zhu et al., 1997), converge slowly near the origin when applied to Equation (10) due to the poor conditioning of P. Observe that $P^\mathsf{T} P$ is block circulant and that ϕ can be chosen so the Hessian of f in (10) remains block circulant. Using the fact that block-circulant matrices may be block-diagonalized by Fourier transforms—see, e.g., (Petersen and Pedersen, 2007), there exists a Hermitian, block-diagonal and positive-definite matrix Π such that

$$\nabla^2 f(\mu) = P^\mathsf{T} P + \lambda \nabla^2 \phi(\mu) = \tfrac{1}{n} \mathscr{F}_n^\star \, \Pi \, \mathscr{F}_n,$$

where $\mathscr{F}_n$ is a discrete Fourier transform (DFT) and $n = n_{\text{vox}}$. Golkar (2013) proposes to seek a diagonal and positive-definite approximation $\Delta \approx \Pi$ and to perform the diagonal scaling in Fourier space

$$C = \tfrac{1}{n} \mathscr{F}_n^\star \, \Delta^{-1/2} \, \mathscr{F}_n, \tag{11}$$

in hopes to improve the conditioning of (10). The simple choice $\Delta = \operatorname{diag}(\Pi)$ has proved effective in practice and is what we use in our implementation.

The change of variables $\mu = Cx$ leads to the *scaled problem*

$$\underset{x}{\text{minimize}} \ \tfrac{1}{2} \|PCx - y\|^2 + \lambda \phi(Cx) \ \text{ subject to } Cx \ge 0. \tag{12}$$

The disadvantage of (12) is that the scaling converts simple bounds into linear inequality constraints.

To illustrate the importance of scaling Equation (10), we report approximate condition numbers κ in cylindrical and Cartesian coordinates on a 2D problem of 128×128 pixels in Figure 1 for various values of the penalty parameter λ. Our approximations are lower bounds on the actual condition number because computing eigenvalues is time intensive, even for such a small problem. The example of Section 6 uses a finer resolution, which results in smaller voxels near the origin, and is likely to have worse conditioning. Figure 1 shows that both regularization and scaling have the potential to improve the condition number of the Hessian drastically.

In a practical implementation, $\mathscr{F}_n$ and $\mathscr{F}_n^\star$ can be applied to a vector in $\mathcal{O}(n \log n)$ time by way of the FFT (Cooley and Tukey, 1965), even though they would materialize as dense matrices. Thus, C can also be applied to a vector in $\mathcal{O}(n \log n)$ time. The presence of such *fast operators* are one of the reasons why it is necessary to devise factorization-free methods for Equation (12). Leveraging the nature of C also allows us to design efficient projection operations, which is why we focus our attention on projection-based active-set methods.

3 Primal active-set methods

Consider a generic optimization problem with linear inequality constraints

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \ f(x) \quad \text{subject to } \ a_i^\mathsf{T} x \ge b_i, \ i = 1, \ \ldots, \ m, \tag{13}$$

and assume that x^* is a local solution. Primal active-set methods are designed from the principle that were the optimal active-set at x^* $\mathcal{A}(x^*)$ known ahead of time, it would suffice to solve the equality-constrained problem

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \ f(x) \quad \text{subject to } \ a_i^\mathsf{T} x = b_i, \ i \in \mathcal{A}(x^*).$$

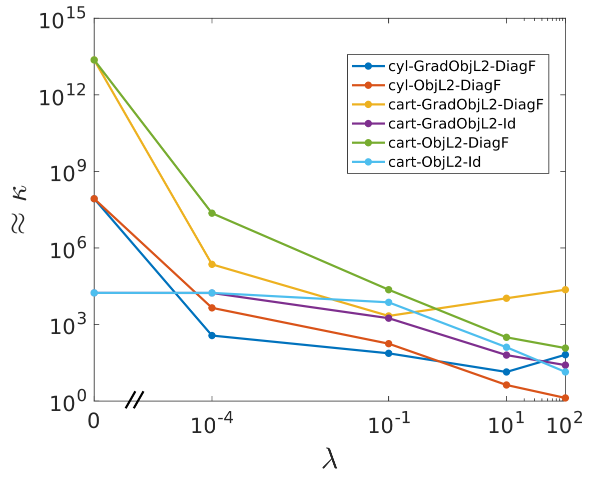


Figure 1: Condition number estimate (κ) of the Hessian as a function of λ. We compare the Hessian in Cartesian ("cart") and cylindrical ("cyl") coordinates for $\mathscr{L}_2$ penalty functions on the object ("ObjL2") and on the gradient of the object ("GradObjL2") using the scaling matrix ("DiagF") or not ("Id"). Note that the scaling matrix (11) only applies in cylindrical coordinates.

The k-th iteration of an active-set method consists in approximately solving the equality-constrained subproblem

$$\underset{x\in\mathbb{R}^n}{\text{minimize}}\ \ f(x) \quad \text{subject to}\ \ a_i^{\mathsf{T}} x = b_i,\ \ i \in \mathcal{A}(x_k), \tag{14}$$

where $\mathcal{A}(x_k) \approx \mathcal{A}(x^*)$. The estimate $\mathcal{A}(x_k)$ is then updated based on local information at the current iterate, including the inactive constraints that are violated, the gradient of f, and possibly Lagrange multiplier estimates.

When the constraints of Equation (13) are simple bounds, those of Equation (14) merely fix a subset of variables. For more general linear inequalities, Equation (14) is a problem with linear equality constraints. The convergence of active-set methods relies on the fact that there are only finitely many possible active-set estimates and that, once the correct active-set has been identified, so has a local solution to Equation (13). For more information on active-set methods, we refer the interested reader to, e.g., (Luenberger and Ye, 2008, Chapter 12).

The active-set methods that we consider below belong to the family of *projected direction* methods. In such methods, certain projection operations are repeatedly applied along the iterations and it is crucial that they be performed efficiently. Recall that if $\mathcal{V} \subseteq \mathbb{R}^n$ is a closed convex set, and if $\bar{x} \in \mathbb{R}^n$, the projection $\mathcal{P}_{\mathcal{V}}[\bar{x}]$ of $\bar{x}$ into $\mathcal{V}$ is well defined, unique and solves

$$\underset{x\in\mathbb{R}^n}{\text{minimize}}\ \ \tfrac{1}{2}\left\|x - \bar{x}\right\|^2 \quad \text{subject to}\ \ x \in \mathcal{V}. \tag{15}$$

Our algorithms require that we design three different projection operations. The first one consists of projecting a vector into the feasible set of Equation (13), whereas the second one consists of projecting a vector into a face of the feasible set of Equation (13), i.e. into the feasible set of Equation (14). The last one can be interpreted as a mixture of the previous projections and consists of projecting into a face of the feasible set while maintaining the feasibility of the *inactive* constraints. We now describe how we perform each type of projection in the context of the problem Equation (12).

3.1 Projection into the polyhedral feasible set

We first describe how a projection into the feasible set of Equation (12),

$$\mathcal{F} = \{x \mid Cx \geq 0\}, \tag{16}$$

may be computed efficiently. The projection $\mathcal{P}_{\mathcal{F}}[\bar{x}]$, with $\bar{x} \in \mathbb{R}^n$, solves

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \ \tfrac{1}{2} \|x - \bar{x}\|^2 \quad \text{subject to} \ Cx \geq 0, \tag{17}$$

where C is given by Equation (11). Unfortunately, Equation (17) appears nearly as difficult as Equation (12). However, as a convex problem, Equation (17) has a Lagrange dual—see, e.g., (Boyd and Vandenberghe, 2010)—whose objective is

$$g(z) = \inf_x \ \tfrac{1}{2} \|x - \bar{x}\|^2 - z^{\mathsf{T}} Cx,$$

where $z \geq 0$ are Lagrange multipliers associated to the constraints of Equation (17) and the argument of the infimum is the Lagrangian of Equation (17). The infimum is attained for $x = \bar{x} + C^{\mathsf{T}} z$, which, when injected into the definition of g, yields

$$g(z) = -\tfrac{1}{2} \|C^{\mathsf{T}} z + \bar{x}\|^2 + \tfrac{1}{2} \|\bar{x}\|^2.$$

If we neglect constant terms in the objective function, the Lagrange dual of Equation (17), which consists in maximizing $g(z)$ subject to $z \geq 0$, may be written

$$\underset{z \in \mathbb{R}^n}{\text{minimize}} \ \tfrac{1}{2} \|C^{\mathsf{T}} z + \bar{x}\|^2 \quad \text{subject to} \ z \geq 0. \tag{18}$$

The dual problem Equation (18) is a bound-constrained linear least-squares problem with a fast operator. If we identify a solution $z^\star$ of Equation (18), we may recover a solution of Equation (17), i.e. $\mathcal{P}_{\mathcal{F}}[\bar{x}]$, as $x^\star = \bar{x} + C^{\mathsf{T}} z^\star$. We employ primal active-set methods, such as those presented in Section 5, to solve Equation (18) and note that the projection of any $\bar{z} \in \mathbb{R}^n$ into the nonnegative orthant can be computed easily as the componentwise $\max(\bar{z}, 0)$.

3.2 Projection into the active face of the polyhedral feasible set

We now describe how a projection into the *active* face of Equation (16),

$$\mathcal{A} = \{x \mid BCx = 0\}, \tag{19}$$

may be computed efficiently, where the *restriction operator* B contains the rows of the identity corresponding to the active-set $\{i \mid c_i^{\mathsf{T}} x = 0\}$. The projection $\mathcal{P}_{\mathcal{A}}[\bar{x}]$ solves

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \ \tfrac{1}{2} \|x - \bar{x}\|^2 \quad \text{subject to} \ BCx = 0. \tag{20}$$

The optimality conditions of Equation (20) may be written as the symmetric indefinite system

$$\begin{bmatrix} I & C^{\mathsf{T}} B^{\mathsf{T}} \\ BC & \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \bar{x} \\ 0 \end{bmatrix}, \tag{21}$$

where y is the vector of Lagrange multipliers associated to the constraints of Equation (20), or, equivalently, as the smaller symmetric positive-definite system

$$BCC^{\mathsf{T}} B^{\mathsf{T}} y = BC\bar{x}. \tag{22}$$

Alternatively, we also recognize Equation (21) and Equation (22) as the optimality conditions of the Lagrange dual of Equation (20), which is the unconstrained linear least-squares problem

$$\underset{y}{\text{minimize}} \ \tfrac{1}{2} \|C^{\mathsf{T}} B^{\mathsf{T}} y - \bar{x}\|^2, \tag{23}$$

and from which we recover $x = \bar{x} - C^{\mathsf{T}} B^{\mathsf{T}} y$.

In our implementation, we consider several possible ways of solving Equation (20). The first is to solve Equation (21) using MINRES (Paige and Saunders, 1975), an iterative method for symmetric, not necessarily definite, linear systems that ensures that the system residual decreases monotonically. The second is to solve Equation (22) with PCG (Hestenes and Stiefel, 1952) or MINRES, which is reasonable in this case because the system is defined by a fast operator that we expect to have a moderate condition number. The third is to solve Equation (23) using an iterative method for linear least-squares problems such as LSQR (Paige and Saunders, 1982) or LSMR (Fong and Saunders, 2011). By design, LSQR and LSMR applied to (23) are equivalent to PCG and MINRES applied to (22), respectively, in exact arithmetic, but should be more accurate and less sensitive to ill-conditioning in finite precision. Early experiments suggest that the first approach does not show any advantage compared to the other two and typically requires more storage. Thus we do not consider it further in the sequel.

3.3 Projection into the "mixed" set

The last type of projection we encounter consists of projecting into the active face Equation (19), while ensuring that the remaining *inactive* constraints remain satisfied. This "mixed" set

$$\mathcal{M} = \{x \mid BCx = 0,\ ACx \geq 0\} \tag{24}$$

is a mixture of the two previous sets, where B is the restriction matrix defined in Section 3.2 and A is the restriction matrix that contains the rows of the identity corresponding to the inactive constraints $\{i \mid c_i^\top x > 0\}$. The projection $\mathcal{P}_\mathcal{M}[\bar{x}]$ solves

$$\underset{x\in\mathbb{R}^n}{\text{minimize}}\ \tfrac{1}{2}\|x-\bar{x}\|^2 \quad \text{subject to}\ BCx = 0,\ ACx \geq 0, \tag{25}$$

which is a special case of Equation (17) where part of the Lagrange multipliers correspond to equality constraints and are free. The dual of Equation (25) is

$$\underset{z\in\mathbb{R}^n}{\text{minimize}}\ \tfrac{1}{2}\|C^\top z + \bar{x}\|^2 \quad \text{subject to}\ Az \geq 0. \tag{26}$$

Thus, solving Equation (26) is at most as difficult as solving Equation (18) and can be done using the same techniques.

4 Solving the reconstruction problem

In this section, we present the methods that we consider to solve problem Equation (12). We first consider a spectral projected gradient (SPG) method (Birgin et al., 2001) First-order methods are attractive due to their low computational cost per iteration, i.e. they don't require Hessian products and only require projections on the feasible set Equation (16). Their main drawback is slow convergence near the minimum, which often materializes as short "zig-zagging" steps. Even though those methods seem beneficial from a computational point of view, they might require considerably more work than second-order methods to reach a solution. Thus we also develop a novel adaptation of the Newton trust-region solver TRON (Lin and Moré, 1999) to the case of linear inequality constraints defined by a fast operator. Our adaptation of TRON requires the three different types of projection described in Section 3.1, Section 3.2 and Section 3.3 respectively.

4.1 Non-monotone spectral projected-gradient method

The non-monotone spectral projected gradient (SPG) algorithm described in Algorithm 1 is based on the refinements to the original algorithm of Barzilai and Borwein (1988) proposed by Birgin et al. (2001) and Birgin and Martínez (2002). Projected-gradient methods are suited to problems with a closed convex feasible set such that projections into this set are inexpensive. In our case, we consider Equation (12) and compute projections into the feasible set $\mathcal{F}$ using the procedure outlined in Section 3.1.

Some of the details of Algorithm 1 are stated in Section A. Algorithm 1 uses Barzilai-Borwein step lengths and a non-monotone Armijo linesearch in conjunction with the projected gradient step.

Barzilai and Borwein (1988) proposed two step lengths. Birgin et al. (2001) recommend using only one of the two step lengths, but Dai and Fletcher (2005), Frassoldati et al. (2008), Bonettini et al. (2009) and di Serafino et al. (2017) employ alternating step length strategies in order to further improve convergence. We use the *adaptive* step length schemes that provide rules for choosing between the two step lengths with dynamic thresholds, such as Algorithm 6 and Algorithm 7.

Algorithm 1 Non-monotone spectral projected gradient

```
1: Initialize $x_0 = \mathcal{P}_{\mathcal{F}}[x_0]$, $k = 0$, $\alpha_0 = 1$ and $g_0 = \nabla f(x_0)$
2: while stopping criteria not met do
3:     $d_k = \mathcal{P}_{\mathcal{F}}[x_k - \alpha_k g_k] - x_k$
4:     compute $\lambda_k$ using Algorithm 4
5:     update $x_{k+1} = x_k + \lambda_k d_k$ and $g_{k+1} = \nabla f(x_{k+1})$
6:     set $s_k = x_{k+1} - x_k$ and $y_k = g_{k+1} - g_k$
7:     update $\alpha_k$ using either Algorithm 5, Algorithm 6 or Algorithm 7
8:     $k = k + 1$
9: end while
```

4.2 TRON for linear inequalities

Lin and Moré (1999) argue that the convergence theory of TRON for bound constraints continues to apply to the case of a linear inequality constraints. In this section, we describe our adaptation of TRON to linear inequalities and emphasize the most costly subproblems. We refer the reader to (Lin and Moré, 1999) for a complete description of the algorithm. Our contention is that when the linear inequality constraints are defined by a fast operator, an efficient implementation remains possible.

TRON is a trust-region active-set method for problems of the form Equation (13). At each iteration, we construct a quadratic approximation of the objective,

$$\psi(w) = g_k^{\mathsf{T}} w + \tfrac{1}{2} w^{\mathsf{T}} H_k w, \tag{27}$$

where $g_k = \nabla f(x_k)$ and $H_k = \nabla^2 f(x_k)$. A central subproblem consists in minimizing $\psi(w)$ on the *active face* of the feasible set, while respecting a trust-region bound. Lin and Moré (1999) establish convergence by requiring that the decrease in the quadratic approximation achieved by a trial step be at least a fraction of the decrease achieved by an approximate *Cauchy step*.

An approximate Cauchy step, denoted s^C, is an approximate minimizer of ψ along the projected-gradient direction, i.e.,

$$s^C = \mathcal{P}_{\mathcal{F}}[x_k - \alpha g_k] - x_k, \tag{28}$$

where the feasible set $\mathcal{F}$ is defined in (16), such that α achieves sufficient decrease, and such that the trust-region bound is satisfied. We employ the same strategy as Lin and Moré (1999) to obtain s^C, except that we limit the number of extrapolation steps, which play no role in guaranteeing convergence, but help take larger steps. Both interpolation and extrapolation steps require one projection and one Hessian product, both of which are costly in our case.

Before tackling the trust-region subproblem, which is the core of the algorithm, we define the *active-set*, representing the constraints that are at their bound at x_k, and the *breakpoints*, which are the step lengths along a direction w from x_k such that previously *inactive* constraints become *active*.

4.2.1 Computing the active-set and breakpoints for linear inequalities

Given the feasible set Equation (16) and $x_k \in \mathcal{F}$, the active-set at x_k is

$$\mathcal{A}(x_k) = \{i \mid c_i^{\mathsf{T}} x_k = 0\}, \tag{29}$$

where c_i^{T} is the i-th row of C.

To compute the breakpoints, we first find the set of indices of inactive constraints that could become violated—or active—if we take a step w from x_k:

$$\mathscr{L} = \{i \mid c_i^{\mathsf{T}} x_k > 0 \text{ and } c_i^{\mathsf{T}} w < 0\}. \tag{30}$$

We then compute the positive step lengths $\alpha_{\mathscr{L}}$ along w such that at least one additional constraint becomes active for the indices Equation (30):

$$\alpha_{\mathscr{L}i} = -\frac{c_i^{\mathsf{T}} x_k}{c_i^{\mathsf{T}} w} \quad (i \in \mathscr{L}), \tag{31}$$

where $\alpha_{\mathscr{L}i} > 0$ denotes the i-th component of $\alpha_{\mathscr{L}}$. Because our interest in the breakpoints is to put boundaries on interpolation and extrapolation steps, we pick the minimal and maximal values of Equation (31).

An additional difficulty is that, in general, we cannot project into $\mathcal{F}$ *exactly*; we must solve Equation (18) approximately. We employ a relative measure ϵ defined by a *projection tolerance*, noted ϵ_{proj}, the norm of C and the norm of x_k:

$$\epsilon = \epsilon_{\text{proj}} \cdot \|C\| \cdot \|x_k\| . \tag{32}$$

In other words, we use a *relaxed* definition of Equation (29) and Equation (30). Given Equation (11), we have $\|C\| = \|\Delta\|^{-1/2}$ (we do not include the factor $1/n$ in the constraint definition), which is readily available. Nearly-active constraints at x_k are those for which $|c_i^\mathsf{T} x_k| \le \epsilon$. We stress that the correct identification of the active-set proves to be critical in practice and the implementation is highly sensitive to the value of ϵ_{proj}.

4.2.2 Solving the Trust-Region problem

Lin and Moré (1999) find an approximate minimizer of (27) on the active face and inside the trust region by constructing a sequence of *minor iterates*. Generating the latter not only allows to update the active-set at each iteration, but also to take *projected steps* in order to *add* more constraints to the active-set. The purpose of this strategy is to quickly determine whether the minimizer of ψ lies in the current active face. Minor iterates are defined so as to ensure both the convergence of the method and feasibility with respect to the feasible set and the trust region. More precisely, a minor iterate $x_{k,j}$ is defined as

$$x_{k,j+1} = \mathcal{P}_{\mathcal{M}}[x_{k,j} + \alpha w^\star] \tag{33}$$

where $\mathcal{M}$ is the set Equation (24), $x_{k,1} := x_k + s^C$ and $w^\star$ is chosen such that

$$\left\|w^\star + x_{k,j} - x_k\right\| \le \Delta.$$

We refer the reader to (Lin and Moré, 1999) for more details. In order to compute Equation (33), we use the procedure outlined in Section 3.3. Finally, $x_{k,j+1}$ must also satisfy a sufficient decrease condition and a suitable α is obtained by way of a linesearch. We now describe how $w^\star$ is obtained as an approximate minimizer of the trust-region subproblem.

For bound-constrained problems, reducing a problem to the active face simply implies *fixing* a subset of variables. Hence, $w^\star$ can be obtained by minimizing $\psi(w)$ subject to the trust region on the free variables. Lin and Moré (1999) employ the truncated conjugate gradient method (TRCG) proposed by Steihaug (1983). In our case, such simplification is not possible. We obtain $w^\star$ as an approximate solution of the quadratic optimization problem with linear equality and trust-region constraints

$$\begin{aligned} \underset{w}{\text{minimize}} \quad & \psi(w+s) \\ \text{subject to} \quad & BCw = 0 \\ & \|w+s\| \le \Delta, \end{aligned} \tag{34}$$

where $s = x_{k,j} - x_k$ represents the current step from x_k, and B is the restriction matrix composed of the rows of the identity corresponding to indices in $\mathcal{A}(x_{k,j})$.

We solve Equation (34) using the projected truncated conjugate-gradient method (Gould et al., 2001, 2013), which essentially amounts to applying TRCG and projecting each conjugate-gradient search direction into $\mathcal{A}$.

Once $w^\star$ is obtained, the next minor iterate is computed using Equation (33). Lin and Moré (1999) propose to terminate the procedure if either $\left\|x_{k,j} - x_k\right\| > \Delta$ or if the decrease condition

$$\|Z^\mathsf{T} \nabla \psi(s)\| \le \epsilon_{\text{grad}} \, \|Z^\mathsf{T} g_k\|$$

is satisfied, where $\epsilon_{\text{grad}} > 0$ is a tolerance, typically 1e$-$2, and the columns of Z form an orthonormal basis for the nullspace of BC—note that the implementation never needs to compute Z explicitly. The former simply implies that we have crossed the trust-region boundary, and the latter that s satisfies a sufficient decrease condition for the inactive variables.

4.2.3 TRON algorithm

In summary, we extend the algorithm of Lin and Moré (1999) to the case of linear inequality constraints. The rules governing the update of the trust-region radius and to determine whether a step is valid are the same as those of Lin and Moré (1999). If a step is rejected, we added the option of an Armijo backtracking linesearch. Algorithm 2 summarizes the projected Newton method.

Algorithm 2 TRON for linear inequalities

```
1: Initialize $x_0 = \mathcal{P}[x_0]$, $f_0 = f(x_0)$ and $g_0 = \nabla f(x_0)$
2: while stopping criteria on Equation (12) not satisfied do
3:     Compute $\psi(w)$ Equation (27)
4:     Compute the Cauchy step Equation (28)
5:     while $\|x_{k,j} - x_k\| \le \Delta$ and $\|Z^\top \nabla\psi(s)\| > \epsilon_{\text{grad}} \|Z^\top g_k\|$ do
6:         Update the active-set Equation (29)
7:         Solve Equation (34) using the projected TRCG method
8:         Compute the next minor iterate Equation (33)
9:     end while
10:    Update the trust-region radius and either accept or reject the step
11: end while
```

5 Solving the projection subproblem

Our current implementation requires accurate solutions of Equation (18) and Equation (26), which we anticipate first-order methods may not be able to achieve. In order to validate this claim, we compare SPG, presented in Section 4.1, with second-order methods for the solution of Equation (18) and Equation (26). The other solvers that we consider are the *two-metric projection* (TMP) algorithm of Gafni and Bertsekas (1984), the original trust-region Newton solver (TRON) of Lin and Moré (1999), and the L-BFGS-B solver of Zhu et al. (1997). Since TRON and SPG were detailed previously, we now detail our implementation of TMP. To a reasonable extent, L-BFGS-B may be understood as a special case of TMP.

5.1 Two-metric projection algorithm

The two-metric projection algorithm (TMP) of Gafni and Bertsekas (1984) is similar to the projected Newton method of Bertsekas (1982). Broadly speaking, two-metric projection methods are active-set methods that use second-order information to *rescale* the steepest descent direction. However, when the feasible set is defined by simple bounds $l \le x \le u$, simplifications occur and stronger convergence results can be obtained using the *binding set* instead of the *active-set*. The binding set comprises the indices of the variables that are at their bound and are likely to remain there were a gradient-descent step taken:

$$\mathcal{B}(x_k) = \{i \mid (x_{ki} = l_i \text{ and } g_{ki} > 0) \text{ or } (x_{ki} = u_i \text{ and } g_{ki} < 0)\}, \tag{35}$$

where x_{ki} and g_{ki} are the i-th components of x_k and g_k. A *rescaled* gradient step d_k is then computed for the *free* variables, i.e., those not in $\mathcal{B}(x_k)$, as the solution of

$$\underset{d_k}{\text{minimize}} \ \tfrac{1}{2} d_k^\top Q_k d_k + g_k^\top d_k \quad \text{subject to } d_{ki} = 0, \ i \in \mathcal{B}(x_k), \tag{36}$$

where Q_k is a symmetric approximation of the Hessian of the objective—possibly a quasi-Newton approximation—such that Q_k is positive definite when restricted to the subspace of free variables. Such methods regained interest recently in the context of large-scale bound-constrained problems, including those arising from machine learning (Schmidt et al., 2011). In our experience, whereas quasi-Newton approximations of the Hessian H_k may yield faster computations, attaining *tighter* optimality tolerances often proves to be impossible. Thus, we allow $Q_k = H_k$—the Hessian of the objective function—and provide Krylov methods to solve Equation (36). The optimality conditions of Equation (36) may be written as the linear system

$$(BH_kB^\top)Bd_k = -Bg_k, \tag{37}$$

where B is the restriction matrix formed with the rows of the identity corresponding to indices $i \notin \mathcal{B}(x_k)$ and the remaining components of d_k are set to zero. As in Section 3.2, candidate Krylov methods for Equation (37) include PCG (Hestenes and Stiefel, 1952) and MINRES (Paige and Saunders, 1975), as well as their counterparts LSQR (Paige and Saunders, 1982) and LSMR (Fong and Saunders, 2011). However, given that LSQR and LSMR handle Equation (37) as a linear least-squares problem, they cannot benefit from the fact that the Hessian H_k of Equation (18) or Equation (26) is

$$H_k = CC^{\mathsf{T}} = \left(\tfrac{1}{n}\,\mathscr{F}_n^{\star}\,\Delta^{-1/2}\,\mathscr{F}_n\right)\left(\tfrac{1}{n}\,\mathscr{F}_n^{\star}\,\Delta^{-1/2}\,\mathscr{F}_n\right)^{\mathsf{T}} = \tfrac{1}{n}\,\mathscr{F}_n^{\star}\,\Delta^{-1}\,\mathscr{F}_n,$$

thus saving two FFT operations. Moreover, whereas Equation (22) had to be solved once to project into the active face, Equation (37) must be solved at each iteration of TMP. For that reason, we recommend PCG and MINRES.

Algorithm 3 summarizes the two-metric projection algorithm that we consider, where $\mathcal{P}_+$ is the projection into the nonnegative orthant.

Algorithm 3 Two-metric projection algorithm for bounded problem

```
1: Initialize $x_0 = \mathcal{P}_+[x_0]$, $k = 0$ and $g_0 = \nabla f(x_0)$
2: while stopping criteria not met do
3:     compute the binding set Equation (35)
4:     compute the descent direction Equation (36)
5:     compute the new iterate $x_{k+1} = \mathcal{P}_+[x_k + \alpha_k d_k]$ where $\alpha_k > 0$ is obtained via a projected Armijo linesearch.
6: end while
```

6 Numerical results

In this section, we compare SPG and TRON in order to identify which performs best on Equation (12). Given that projections play a crucial role in our implementations, we are also interested in determining which solvers lead to both more accurate projections and shorter run times. To that effect, we compare various combinations of reconstruction solvers and projection solvers for the solution of Equation (12). Synthetic projection data, i.e. intensity measurements, as well a realistic human phantom, are provided by the software XCAT, developed by Segars et al. (2008). The geometric parameters and X-ray source characteristics were chosen to emulate a standard clinical scanner, with a measurement quality comparable to that of standard acquisition protocols. Using different techniques for data generation and reconstruction allows us to avoid the *inverse crime* Wirgin (2004) to a large extent.

We consider a 2D example consisting of a slice of abdomen, shown in Figure 2, and aim to reconstruct it from the corresponding intensity measurements. Our example contains 512×512 square pixels of length 0.7 mm each, and the sinogram was obtained from 672 detectors and 1,160 scan angles. In Cartesian coordinates, $n_{\text{meas}} = 672 \cdot 1,160 = 779,520$ and $n_{\text{vox}} = 512^2 = 262,144$. In cylindrical coordinates, recall that we assume that the number of angular elements must be an integer multiple of the scan angles. Say we consider 1,160 angular elements, the number of radial elements is $512^2/1,160 \approx 225.9$, which we round up to 226. The effective number of voxels is then $n_{\text{vox}} = 226 \times 1160 = 262,160$. Thus, we have slightly more voxels in cylindrical coordinates due to the size difference between polar and square pixels.

We study the impact of the penalty function by comparing $\mathscr{L}_2$ penalty functions on the gradient and on the object respectively. For each solver, we use the norm of the projected gradient as an optimality measure. For Equation (10) and Equation (12), we use the stopping condition

$$\|x_k - \mathcal{P}[x_k - g_k]\| \le \epsilon_a + \epsilon_r\,\|x_0 - \mathcal{P}[x_0 - g_0]\|,$$

with $\epsilon_a = \epsilon_r = 1\text{e}{-8}$ and where $\mathcal{P}$ is either $\mathcal{P}_+$ or $\mathcal{P}_{\mathcal{F}}$. We set $\epsilon_{\text{proj}} = 1\text{e}{-11}$ for the projection operations Equation (18), Equation (20) and Equation (26). We set a tolerance of 1e−15 on the progress to avoid stagnation in case the norm of the search direction or the improvement in the objective becomes too small. Finally, we limit the total run time to 15 minutes and allow a maximum of 5 minutes for any one projection operation.

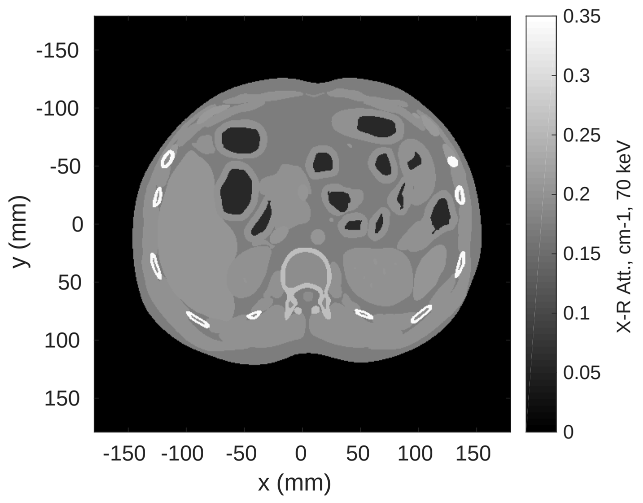


Figure 2: Original phantom: slice of abdomen, 512×512 pixels of size 0.7 mm × 0.7 mm. Sinogram obtained from 672 detectors and $1,160$ projection angles.

To further validate our results, we compare our algorithms with L-BFGS-B applied to Equation (10) in *Cartesian coordinates*, which is our benchmark and offers the best results according to Hamelin (2009). Our goal is to obtain similar performance as L-BFGS-B, while reducing drastically the memory footprint. In this specific example, the projection matrices in Cartesian coordinates and cylindrical coordinates require about 1.0Gb and 4.4Mb, respectively, i.e., cylindrical coordinates allow savings of a factor of about 233. However, keep in mind that Equation (10) in Cartesian coordinates and Equation (12) in cylindrical coordinates are different problems altogether. All solvers are implemented in MATLAB except L-BFGS-B, which is in C++.

The tables of results use the following nomenclature. The columns $\mathcal{P}_{\mathcal{A}}$ and $\mathcal{P}_{\mathcal{F}}$ indicate the choice of method to compute the corresponding projection. The column labeled KKT gives the final optimality residual $\|x_k - \mathcal{P}[x_k - g_k]\|$, "time" gives the run time in seconds, $f(x^*)$ is the final objective value, $\|\mu < 0\|$ is the final infeasibility, "$\#P$" and "$\#C$" are the number of operator-vector products with P and C, respectively, and their adjoints, "#iter" is the number of iterations, "$\#\mathcal{P}_{\mathcal{F}}$" and "$\#\mathcal{P}_{\mathcal{A}}$" are the number of projections into the feasible set and into an active face. The numbers of projections $\mathcal{P}_{\mathcal{F}}$ and $\mathcal{P}_{\mathcal{M}}$ are grouped together given the similarity of the projection problems. The failure codes reported are as follows: "cpu" indicates that the maximum run time was exceeded, "prog" indicates that the improvement in the objective function was too small and "xfail" indicates an unknown failure.

Table 1, Table 2, and Table 3 report the results of L-BFGS-B on (10) and TRON and SPG on (12) using a penalty on the gradient of the object.

Table 1: L-BFGS-B applied to Equation (10) using a $\mathscr{L}_2$ penalty function on the gradient of the object with $\lambda = 15$.

Proj. Solver	KKT	time	$f(x^\star)$	$\|\mu < 0\|$	#P	#C	#iter	$\#\mathcal{P}_{\mathcal{F}}$	$\#\mathcal{P}_{\mathcal{A}}$	fail
N/A	3.4e−6	2.0e2	6.8e3	0	136	N/A	61	N/A	N/A	

Similarly, Table 4, Table 5 and Table 6 compare L-BFGS-B, TRON and SPG using a penalty on the object.

For both penalty functions, TRON is effective and fast, and is accurate in terms of final infeasibility, especially when combined with either LSQR or LSMR to apply $\mathcal{P}_{\mathcal{A}}$ and TMP-PCG or TMP-MINRES to apply $\mathcal{P}_{\mathcal{F}}$. Most combinations perform reasonably, with a few exceptions; mostly combinations involving SPG that are less accurate, stagnate or fail to converge.

Furthermore, the results highlight the effectiveness of the scaling Equation (11), as the run times for TRON are similar to those for L-BFGS-B. Such results are encouraging and show the potential of specifically-tailored projection methods for large-scale problems.

Table 2: TRON reconstruction results on Equation (12) using a $\mathscr{L}_2$ penalty function on the gradient of the object with $\lambda = 0.1$.

$\mathcal{P}_\mathcal{A}$	$\mathcal{P}_\mathcal{F}$	KKT	time	$f(x^\star)$	$\|\mu < 0\|$	$\#P$	$\#C$	#iter	$\#\mathcal{P}_\mathcal{F}$	$\#\mathcal{P}_\mathcal{A}$	fail
LSQR	TMP-PCG	3.6e−5	1.5e2	6.9e3	2.2e−14	240	5914	9	46	86	
	TMP-MINRES	3.6e−5	1.4e2	6.9e3	2.1e−14	240	5840	9	46	86	
	SPG	2.0e−5	1.4e2	6.9e3	1.1e−8	240	5527	9	46	86	
	TRON	3.6e−5	4.6e2	6.9e3	3.2e−14	240	33337	9	46	86	
	L-BFGS-B	3.5e−5	1.4e2	6.9e3	2.4e−7	246	4619	9	47	87	
LSMR	TMP-PCG	3.6e−5	1.5e2	6.9e3	2.1e−14	240	5908	9	46	86	
	TMP-MINRES	3.6e−5	1.5e2	6.9e3	2.1e−14	240	5834	9	46	86	
	SPG	2.4e−4	1.4e2	6.9e3	1.7e−9	250	5405	9	46	86	prog.
	TRON	3.6e−5	4.6e2	6.9e3	3.1e−14	240	29182	9	46	86	
	L-BFGS-B	3.5e−5	1.4e2	6.9e3	2.4e−7	246	4645	9	47	87	
MINRES	TMP-PCG	3.6e−5	1.4e2	6.9e3	3.6e−9	240	4471	9	46	86	
	TMP-MINRES	3.6e−5	1.4e2	6.9e3	3.6e−9	240	4397	9	46	86	
	SPG	3.0e−5	1.3e2	6.9e3	3.8e−9	242	3997	9	45	87	
	TRON	3.6e−5	4.5e2	6.9e3	3.6e−9	240	27164	9	46	86	
	L-BFGS-B	3.5e−5	1.2e2	6.9e3	2.4e−7	246	3109	9	47	87	
PCG	TMP-PCG	3.6e−5	1.5e2	6.9e3	1.7e−11	240	4855	9	46	86	
	TMP-MINRES	3.6e−5	1.4e2	6.9e3	1.7e−11	240	4781	9	46	86	
	SPG	2.0e−5	1.3e2	6.9e3	2.8e−8	238	4240	9	46	84	
	TRON	3.6e−5	4.6e2	6.9e3	1.7e−11	240	27524	9	46	86	
	L-BFGS-B	3.5e−5	1.3e2	6.9e3	2.4e−7	246	3523	9	47	87	

Table 3: SPG reconstruction results on Equation (12) using a $\mathscr{L}_2$ penalty function on the gradient of the object with $\lambda = 0.1$.

Step	$\mathcal{P}_\mathcal{F}$	KKT	time	$f(x^\star)$	$\|\mu < 0\|$	#P	#C	#iter	$\#\mathcal{P}_\mathcal{F}$	$\#\mathcal{P}_\mathcal{A}$	fail
BB1	TMP-PCG	3.5e−5	8.1e2	6.9e3	2.6e−15	610	41867	176	353	0	
	TMP-MINRES	2.8e−4	8.6e2	6.9e3	2.6e−15	729	41190	209	419	0	prog.
	SPG	7.9e−4	3.7e2	6.9e3	3.0e−7	499	18143	139	279	0	prog.
	TRON	1.4e0	9.0e2	6.9e3	3.0e−14	166	54611	49	99	0	cpu
	L-BFGS-B	6.6e−5	6.0e2	6.9e3	1.8e−7	535	16948	156	313	0	
ABB	TMP-PCG	2.5e−4	4.8e2	6.9e3	4.3e−15	218	26889	70	141	0	prog.
	TMP-MINRES	2.6e−4	4.2e2	6.9e3	2.5e−15	218	22370	70	141	0	prog.
	SPG	4.5e−3	1.9e2	6.9e3	1.0e−6	244	9889	66	133	0	prog.
	TRON	2.8e−2	9.0e2	6.9e3	2.3e−14	156	56879	50	101	0	cpu
	L-BFGS-B	3.8e−4	4.1e2	6.9e3	1.2e−7	262	11797	77	155	0	prog.
ABB_{min1}	TMP-PCG	4.2e−4	5.0e2	6.9e3	6.8e−15	240	27727	75	151	0	prog.
	TMP-MINRES	4.2e−4	4.3e2	6.9e3	2.6e−15	240	23065	75	151	0	prog.
	SPG	1.8e−4	2.4e2	6.9e3	2.4e−7	327	11937	84	171	0	prog.
	TRON	7.1e−2	9.0e2	6.9e3	4.0e−7	158	56808	50	101	0	cpu
	L-BFGS-B	5.0e−4	4.1e2	6.9e3	1.3e−7	259	11604	74	149	0	prog.
ABB_{SS}	TMP-PCG	2.2e−4	5.2e2	6.9e3	2.0e−12	257	29508	83	167	0	prog.
	TMP-MINRES	5.6e−4	4.5e2	6.9e3	3.3e−15	257	24537	83	167	0	prog.
	SPG	9.6e−5	2.2e2	6.9e3	2.7e−9	271	11693	86	173	0	
	TRON	7.8e−4	6.9e2	6.9e3	4.8e−15	227	41694	73	147	0	prog.
	L-BFGS-B	7.0e−4	3.7e2	6.9e3	8.0e−8	249	10812	79	159	0	prog.

Table 4: L-BFGS-B applied to Equation (10) using a $\mathscr{L}_2$ penalty function on the object with $\lambda = 25$.

Proj. Solver	KKT	time	$f(x^\star)$	$\|\mu < 0\|$	#P	#C	#iter	$\#\mathcal{P}_\mathcal{F}$	$\#\mathcal{P}_\mathcal{A}$	fail
N/A	2.3e−6	1.0e2	1.2e5	0	114	N/A	49	N/A	N/A	

The discrepancy between the run times for TRON and SPG confirms that projections into the feasible set Equation (17) dominate the cost of the projections on the active face Equation (20). It is thus favorable to avoid taking too many projected gradient steps.

Our results show that, despite appearing prohibitive for large-scale problems, second-order methods can be competitive after all, provided that subproblems can be solved efficiently.

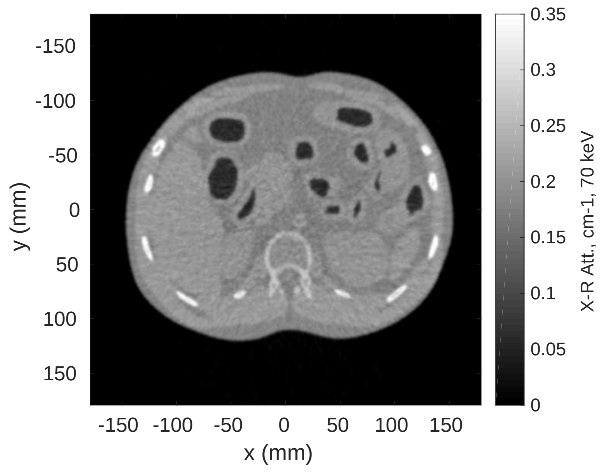
Table 5: TRON reconstruction results on Equation (12) using a $\mathscr{L}_2$ penalty function on the object with $\lambda = 0.1$.

$\mathcal{P}_\mathcal{A}$	$\mathcal{P}_\mathcal{F}$	KKT	time	$f(x^\star)$	$\|\mu < 0\|$	$\#P$	$\#C$	#iter	$\#\mathcal{P}_\mathcal{F}$	$\#\mathcal{P}_\mathcal{A}$	fail
	TMP-PCG	1.9e−4	1.2e2	7.1e4	2.8e−14	193	5740	8	38	63	
	TMP-MINRES	1.9e−4	1.2e2	7.1e4	2.6e−14	193	5297	8	38	63	
LSQR	SPG	1.9e−4	1.1e2	7.1e4	2.9e−10	193	5367	8	38	63	
	TRON	1.9e−4	4.3e2	7.1e4	6.2e−14	193	33407	8	38	63	
	L-BFGS-B	1.9e−4	1.2e2	7.1e4	1.1e−7	193	4303	8	38	63	
	TMP-PCG	1.9e−4	1.2e2	7.1e4	2.5e−14	193	5738	8	38	63	
	TMP-MINRES	1.9e−4	1.2e2	7.1e4	2.4e−14	193	5295	8	38	63	
LSMR	SPG	1.9e−4	1.1e2	7.1e4	3.2e−10	193	5207	8	38	63	
	TRON	1.9e−4	4.2e2	7.1e4	6.1e−14	193	33536	8	38	63	
	L-BFGS-B	1.9e−4	1.3e2	7.1e4	1.1e−7	193	4323	8	38	63	
	TMP-PCG	1.9e−4	1.2e2	7.1e4	1.2e−15	193	4885	8	38	63	
	TMP-MINRES	1.9e−4	1.1e2	7.1e4	1.2e−15	193	4441	8	38	63	
MINRES	SPG	7.3e−5	1.1e2	7.1e4	4.8e−10	222	4909	9	43	73	
	TRON	2.5e−3	4.0e2	7.1e4	4.6e−11	160	32301	7	33	51	cpu.
	L-BFGS-B	1.9e−4	1.2e2	7.1e4	1.2e−7	193	3416	8	38	63	
	TMP-PCG	1.9e−4	1.2e2	7.1e4	1.0e−11	193	5139	8	38	63	
	TMP-MINRES	1.9e−4	1.2e2	7.1e4	1.0e−11	193	4696	8	38	63	
PCG	SPG	7.3e−5	1.1e2	7.1e4	4.3e−10	222	5142	9	43	73	
	TRON	1.9e−4	4.2e2	7.1e4	1.0e−11	193	33556	8	38	63	
	L-BFGS-B	1.9e−4	1.2e2	7.1e4	1.1e−7	193	3657	8	38	63	

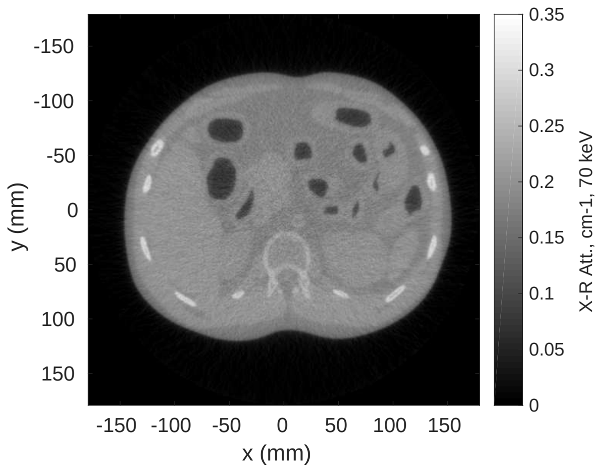
Table 6: SPG reconstruction results on Equation (12) using a $\mathscr{L}_2$ penalty function on the object with $\lambda = 0.1$.

Step	$\mathcal{P}_\mathcal{F}$	KKT	time	$f(x^\star)$	$\|\mu < 0\|$	$\#P$	$\#C$	#iter	$\#\mathcal{P}_\mathcal{F}$	$\#\mathcal{P}_\mathcal{A}$	fail
	TMP-PCG	1.5e−4	3.2e2	7.1e4	2.0e−15	406	17031	118	237	0	
	TMP-MINRES	2.8e−4	3.2e2	7.1e4	8.8e−16	410	16161	120	241	0	prog.
BB1	SPG	2.1e−4	2.6e2	7.1e4	4.9e−8	384	15080	107	215	0	prog.
	TRON	3.3e−5	6.1e2	7.1e4	1.1e−14	397	43760	116	233	0	
	L-BFGS-B	1.3e−4	3.0e2	7.1e4	5.4e−7	376	9505	110	221	0	
	TMP-PCG	2.8e−3	1.9e2	7.1e4	6.5e−15	211	11351	68	137	0	prog.
	TMP-MINRES	2.8e−3	1.9e2	7.1e4	9.3e−16	211	10253	68	137	0	prog.
ABB	SPG	4.9e−3	1.6e2	7.1e4	4.5e−10	202	11187	65	131	0	prog.
	TRON	1.0e−4	8.0e2	7.1e4	1.2e−6	235	58716	75	151	0	
	L-BFGS-B	1.9e−4	2.2e2	7.1e4	2.7e−7	227	7548	73	147	0	
	TMP-PCG	3.5e−4	2.1e2	7.1e4	2.5e−15	228	11776	70	141	0	prog.
	TMP-MINRES	3.7e−4	2.0e2	7.1e4	9.4e−16	220	10335	68	137	0	prog.
ABB_{min1}	SPG	1.7e−4	1.7e2	7.1e4	1.2e−8	219	11308	68	137	0	
	TRON	1.4e−3	4.9e2	7.1e4	3.6e−14	213	37906	66	133	0	prog.
	L-BFGS-B	3.7e−4	2.2e2	7.1e4	1.3e−7	220	7333	68	137	0	prog.
	TMP-PCG	1.9e−4	2.0e2	7.1e4	7.7e−14	221	11486	71	143	0	
	TMP-MINRES	7.9e−4	1.7e2	7.1e4	7.8e−16	196	9608	63	127	0	prog.
ABB_{SS}	SPG	1.3e−4	1.8e2	7.1e4	2.0e−10	233	11579	75	151	0	
	TRON	1.9e−4	4.9e2	7.1e4	1.4e−14	219	37979	71	143	0	
	L-BFGS-B	1.9e−4	2.3e2	7.1e4	1.3e−7	233	7598	75	151	0	

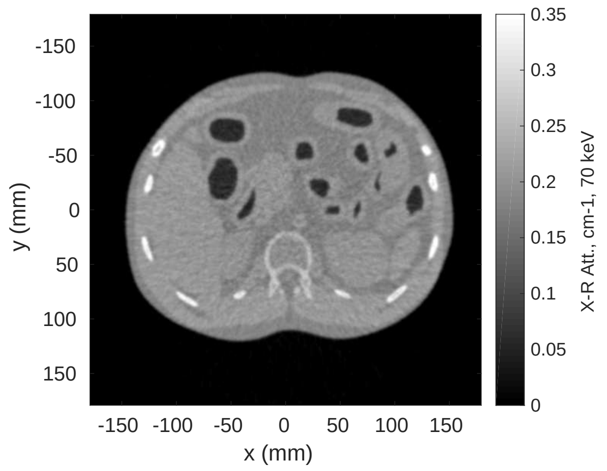
Figure 3 illustrates reconstructed images for the "best" candidate with each penalty function and each solver. Given that the results are fairly similar and that no combination stands out from the others, we pick those with the best overall performance: the combinations of SPG/ABB/TMP-MINRES and TRON/LSQR/TMP-MINRES. For the projection on the active face, we tend to favor LSQR and LSMR over their PCG and MINRES counterparts because of their improved numerical stability.



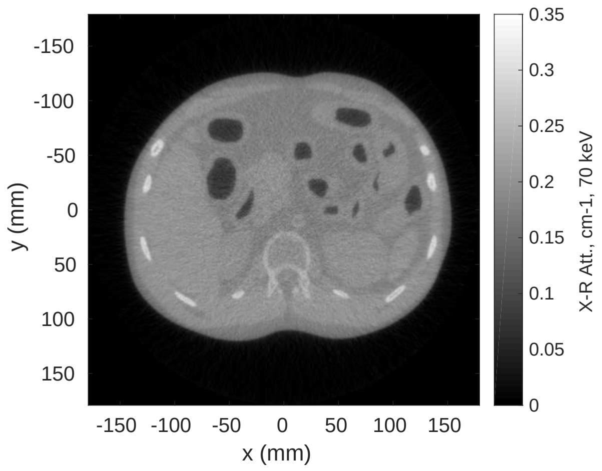
(a) TRON on Equation (12), $\lambda = 0.1$



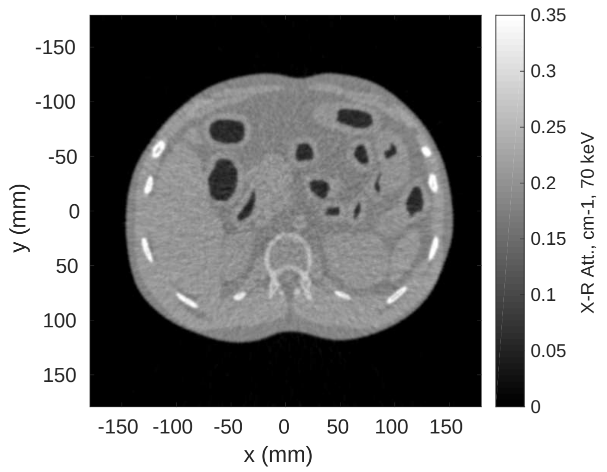
(b) TRON on Equation (12), $\lambda = 0.1$



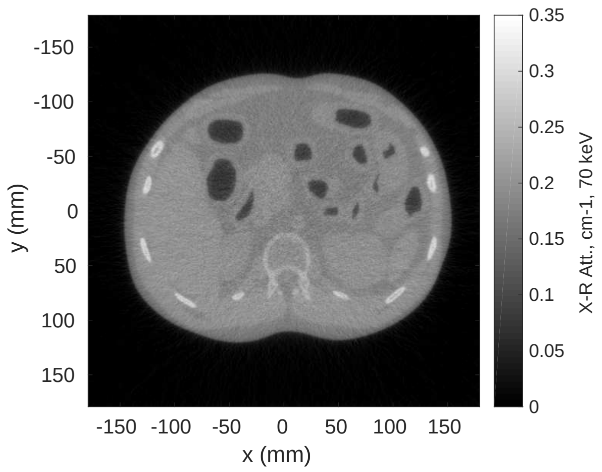
(c) SPG on Equation (12), $\lambda = 0.1$



(d) SPG on Equation (12), $\lambda = 0.1$



(e) L-BFGS-B on Equation (10), $\lambda = 15$



(f) L-BFGS-B on Equation (10), $\lambda = 25$

Figure 3: Reconstruction results using the $\mathscr{L}_2$ norm on the gradient of the object (left) and the $\mathscr{L}_2$ norm on the object (right). Problem Equation (12) is posed in cylindrical coordinates while Equation (10) is posed in Cartesian coordinates.

7 Discussion

We studied and designed factorization-free implementations for Equation (12) by capitalizing on its fast Jacobian operator. This motivated us to develop efficient projection operations, and efficient implementations of projection-based active-set methods.

Our results not only show the effectiveness of specifically tailored projected methods but also that second-order methods on large-scale problems arising from image reconstruction can be viable.

Moreover, the drastic reduction in memory requirements obtained using cylindrical coordinates might allow the application of iterative methods to 3D tomographic reconstruction on common computers. For that reason, we believe that the present work is a step towards applying iterative methods in clinical settings in the near future.

The impact of *inexact* projection must be assessed and it would be interesting to have theoretical results indicating how inexact projections are allowed to be. For the moment, we are content with using tight tolerances on the projection solvers, which ensure that infeasibility is insignificant in comparison to the accuracy of the iterates. Finally, we are considering factorization-free implementations of other approaches, including interior-point methods and proximal algorithms.

Implementations of our solvers are available in object-oriented MATLAB as part of the NLPLab optimization framework available at `https://bitbucket.org/maxmcl/nlplab`.

Appendix A Step length updates for Barzilai-Borwein methods

In this section, we state the non-monotone Armijo linesearch proposed by Birgin et al. (2001) and Birgin and Martínez (2002) as Algorithm 4. This line search uses quadratic interpolation to initialize λ_{temp}. Typical values are $\gamma = 1\text{e}{-4}$, $\sigma_1 = 0.1$, $\sigma_2 = 0.9$ and $m = 10$.

Algorithm 4 Safeguarded non-monotone Armijo line search

```
1: Given $0 < \sigma_1 < \sigma_2 < 1$, $\gamma \in (0, 1)$, $m \in \mathbb{N}_0^+$, $x_k \in \mathbb{R}^n$, $\nabla f(x_k)$ and a direction $d_k$,
2: set $f_{\max} = \max\{f(x_{k-j}) \mid 0 \le j \le \min(k, m-1)\}$
3: set $\lambda_k = 1$, $x_{k+1} = x_k + \lambda_k d_k$, $\delta = \nabla f(x_k)^\top d_k$
4: while $f(x_{k+1}) > f_{\max} + \gamma \lambda_k \delta$ do
5:     $\lambda_{\text{temp}} = -\lambda_k^2 \delta / (2(f(x_{k+1}) - f(x_k) - \lambda_k \delta))$
6:     if $\sigma_1 \le \lambda_{\text{temp}} \le \sigma_2$ then
7:         set $\lambda_k = \lambda_{\text{temp}}$
8:     else
9:         set $\lambda_k = \frac{\lambda_k}{2}$
10:    end if
11:    $x_{k+1} = x_k + \lambda_k d_k$
12: end while
13: return $\lambda_k$, $x_{k+1}$
```

We now state the various step length update schemes that we consider for Algorithm 1. Recall that at iteration k, $s_k = x_{k+1} - x_k$ and $y_k = g_{k+1} - g_k$.

Algorithm 5, known as the *first* Barzilai-Borwein step length (BB1), gives the step used by Birgin et al. (2001). Typical values for $\alpha_{\max}$ and $\alpha_{\min}$ are 1.0e3 and 1.0e−3 respectively.

Algorithm 5 Safeguarded first Barzilai-Borwein step (BB1)

```
1: Given the step length safeguards $\alpha_{\max} > \alpha_{\min} > 0$ and vectors $s_k$ and $y_k$
2: if $s_k^\top y_k < 0$ then
3:     return $\alpha_{\max}$
4: else
5:     return $\min\{\alpha_{\max}, \max\{\alpha_{\min}, s_k^\top s_k / s_k^\top y_k\}\}$
6: end if
```

We now state the various adaptative update rules that we consider. For a survey, we refer the reader to the works of di Serafino et al. (2017). Algorithm 6 ($\mathrm{ABB}_{\mathrm{min1}}$), suggested by Frassoldati et al. (2008), determines whether the first or the smallest of the last m_α second Barzilai-Borwein step length should be used, given a threshold value τ. The original ABB algorithm presented by Zhou et al. (2006) can be derived from Algorithm 6 if we set $m_\alpha = 1$. Typical values of the parameters are $\tau = 0.8$ and $m_\alpha = 9$.

Algorithm 6 Adaptive minimal Barzilai-Borwein step ($\mathrm{ABB}_{\mathrm{min1}}$)

```
1: Given $\tau$, the memory $m_\alpha$ and the inputs $s_k$ and $y_k$,
2: compute $\alpha^{(1)} = s_k^\top s_k / s_k^\top y_k$ and $\alpha^{(2)} = s_k^\top y_k / y_k^\top y_k$
3: if $\alpha^{(2)} < \tau\,\alpha^{(1)}$ then
4:     return $\min\{\alpha_j^{(2)} \mid j = \max(1, k - m_\alpha), \dots, k\}$
5: else
6:     return $\alpha^{(1)}$
7: end if
```

Bonettini et al. (2009) propose Algorithm 7 ($\mathrm{ABB}_{\mathrm{SS}}$), that is similar to Algorithm 6, but uses a dynamic threshold τ and safeguards on α, as in Algorithm 5. Because Equation (12) is already scaled, we set the scaling matrix as the identity in our implementation of Algorithm 7. Typical values of the parameters are $\tau = 0.5$ and $m_\alpha = 2$.

Algorithm 7 Safeguarded adaptive minimal B.-B. step with dynamic threshold ($\mathrm{ABB}_{\mathrm{SS}}$)

```
1: Given $\tau$, $\alpha_{\max} > \alpha_{\min} > 0$, the memory $m_\alpha$ and vectors $s_k$ and $y_k$,
2: if $s_k^\top y_k \le 0$ then
3:     $\alpha^{(1)} = \alpha_{\max}$
4:     $\alpha^{(2)} = \alpha_{\max}$
5: else
6:     $\alpha^{(1)} = \max\{\alpha_{\min}, \min\{s_k^\top s_k / s_k^\top y_k, \alpha_{\max}\}\}$
7:     $\alpha^{(2)} = \max\{\alpha_{\min}, \min\{s_k^\top y_k / y_k^\top y_k, \alpha_{\max}\}\}$
8: end if
9: if $\alpha^{(2)} \le \tau\,\alpha^{(1)}$ then
10:     $\alpha = \min\{\alpha_j^{(2)} \mid j = \max(1, k - m_\alpha), \dots, k\}$
11:     $\tau = 0.9 \cdot \tau$
12: else
13:     $\alpha = \alpha^{(1)}$
14:     $\tau = 1.1 \cdot \tau$
15: end if
16: return $\alpha$
```

Appendix B TRON for bounded problems compared to IPOPT

In this section, we validate our MATLAB implementation of TRON solver against IPOPT (Wächter and Biegler, 2006) on a selection of bound-constrained problems from the CUTE library in AMPL format.[1] Our implementation of TRON accomodates (13) and we specialize it to bound constraints by supplying appropriate projection functions—see Section 3.1, Section 3.2 and Section 3.3. IPOPT is run with default parameters. Table 7 uses the same failure codes as in Section 6. Given the smaller sizes of the problems, we reduced the maximal allowed run time to 120 seconds and added a limit of 1e5 iterations, corresponding to the the error code “iter”.

Table 7: TRON vs. IPOPT on bound-constrained problems from CUTE.

Problem	Solver	KKT	time	$f(x^\star)$	#iter	$\#f$	$\#g$	$\#H$	fail
3pk	IPOPT	3.10e−12	2e−2	1.72e0	11	12	12	11	
	TRON	6.68e−7	2e−1	1.72e0	111	111	111	220	

Continued on next page

[1] https://github.com/mpf/Optimization-Test-Problems

Table 7 – continued from previous page

Problem	Solver	KKT	time	$f(x^\star)$	#iter	$\#f$	$\#g$	$\#H$	fail
allinit	IPOPT	3.87e−15	4e−3	1.67e1	12	20	13	12	
	TRON	4.97e−9	1e2	1.67e1	18	44	9	44	prog.
bdexp	IPOPT	8.82e−11	1e−1	2.13e−8	21	22	22	21	
	TRON	4.45e−9	2e−1	2.13e−8	22	22	22	42	
biggsb1	IPOPT	3.92e−16	2e−2	1.50e−2	20	21	21	20	
	TRON	2.73e−10	7e−1	1.50e−2	530	530	530	1058	
bqpgabim	IPOPT	1.39e−17	8e−3	−3.79e−5	18	24	19	18	
	TRON	8.26e−11	3e−2	−3.79e−5	8	8	8	14	
bqpgasim	IPOPT	5.02e−17	8e−3	−5.52e−5	19	25	20	19	
	TRON	3.29e−10	2e−2	−5.52e−5	26	84	13	66	prog.
camel6	IPOPT	9.92e−16	8e−3	−1.03e0	11	12	12	11	
	TRON	2.46e−14	1e−2	−1.03e0	9	11	9	17	
chenhark	IPOPT	1.67e−15	3e−2	−2.00e0	21	22	22	21	
	TRON	5.47e−8	5e1	−2.00e0	3376	3434	3360	6767	prog.
cvxbqp1	IPOPT	2.74e−12	6e−1	2.25e6	11	12	12	11	
	TRON	2.78e−15	1e−2	2.25e6	2	2	2	2	
deconvb	IPOPT	2.44e−9	1e1	3.04e−13	10000	47482	10001	10000	iter.
	TRON	3.36e−9	5e−2	1.33e−12	46	49	46	91	
eg1	IPOPT	4.24e−16	4e−3	−1.43e0	8	9	9	8	
	TRON	2.38e−8	1e2	−1.13e0	19	51	7	48	cpu.
explin	IPOPT	4.34e−14	1e−2	−7.24e5	19	20	20	19	
	TRON	2.33e−6	1e2	−7.24e5	34	51	19	82	cpu.
explin2	IPOPT	2.77e−14	2e−2	−7.24e5	19	20	20	19	
	TRON	7.16e−5	1e2	−7.24e5	31	48	14	77	prog.
hadamals	IPOPT	1.73e−14	2e−1	2.53e1	109	110	110	109	
	TRON	2.00e−9	2e−2	8.13e2	10	10	10	18	
harkerp2	IPOPT	2.55e−15	2e−2	−5.00e−1	17	18	18	17	
	TRON	0.00e0	1e−2	−5.00e−1	3	3	3	4	
hart6	IPOPT	1.90e−15	8e−3	−3.32e0	9	15	10	9	
	TRON	4.59e−8	1e2	−3.32e0	24	48	11	63	prog.
hatflda	IPOPT	2.23e−14	8e−3	9.52e−21	11	12	12	11	
	TRON	6.21e−13	2e−2	3.47e−25	30	30	30	58	
hatfldb	IPOPT	7.39e−15	1e−2	5.57e−3	11	12	12	11	
	TRON	2.00e−15	1e−2	5.57e−3	27	27	27	52	
hatfldc	IPOPT	1.16e−16	4e−3	6.44e−24	6	7	7	6	
	TRON	0.00e0	5e−3	0.00e0	6	6	6	10	
himmelp1	IPOPT	5.79e−15	1e−2	−6.21e1	13	21	14	13	
	TRON	1.09e−8	1e2	−6.21e1	23	91	14	56	prog.
hs110	IPOPT	3.68e−15	4e−3	−4.58e1	7	8	8	7	
	TRON	8.44e−12	1e−2	−4.58e1	18	18	18	34	
hs3mod	IPOPT	0.00e0	8e−3	−9.99e−9	6	7	7	6	
	TRON	0.00e0	5e−3	0.00e0	5	5	5	8	
logros	IPOPT	4.18e−13	8e−2	0.00e0	76	679	77	76	
	TRON	2.56e−9	1e2	0.00e0	47	81	33	115	cpu.

Continued on next page

Table 7 – continued from previous page

Problem	Solver	KKT	time	$f(x^\star)$	#iter	$\#f$	$\#g$	$\#H$	fail
maxlika	IPOPT	1.51e−12	2e−2	1.14e3	20	26	21	20	
	TRON	5.71e−11	4e−2	1.15e3	14	18	14	28	
mccormck	IPOPT	9.17e−16	4e0	−4.57e4	18	237	19	18	
	TRON	4.69e−5	1e2	−4.57e4	30	98	16	77	cpu.
mdhole	IPOPT	0.00e0	4e−2	−9.99e−9	48	119	49	48	
	TRON	8.64e−18	1e−2	1.87e−37	15	21	15	30	
ncvxbqp1	IPOPT	9.18e−13	3e1	−1.99e10	261	262	262	261	
	TRON	5.67e−16	2e−2	−1.99e10	2	2	2	2	
ncvxbqp2	IPOPT	2.82e4	1e2	−1.33e10	1161	1162	1162	1161	xfail.
	TRON	1.27e−2	1e2	−1.33e10	32	59	13	81	cpu.
ncvxbqp3	IPOPT	1.02e5	1e2	−6.32e9	975	976	976	975	xfail.
	TRON	1.76e−3	1e2	−6.56e9	29	117	14	72	prog.
nonscomp	IPOPT	2.51e−11	3e−1	1.96e−8	27	88	28	27	
	TRON	7.88e−7	1e−1	9.79e−15	12	12	12	22	
obstclal	IPOPT	2.73e−16	1e−2	1.40e0	15	16	16	15	
	TRON	2.23e−11	8e−3	1.40e0	11	13	11	21	
obstclbl	IPOPT	2.35e−16	4e−3	2.88e0	12	13	13	12	
	TRON	3.36e−9	1e2	2.88e0	20	91	8	50	cpu.
obstclbu	IPOPT	2.49e−16	8e−3	2.88e0	13	14	14	13	
	TRON	2.16e−11	6e−3	2.88e0	7	7	7	12	
oslbqp	IPOPT	9.09e−17	1e−2	6.25e0	17	18	18	17	
	TRON	0.00e0	4e−3	6.25e0	2	2	2	2	
palmer1	IPOPT	1.77e−9	4e−1	1.18e4	766	1983	767	767	prog.
	TRON	1.87e−8	3e−2	2.82e4	61	61	61	120	
palmer5a	IPOPT	1.91e1	7e0	2.82e−2	10000	50184	10001	10000	iter.
	TRON	8.91e−4	5e0	5.86e−2	9802	10000	9802	19695	prog.
palmer5b	IPOPT	2.15e−11	4e−2	9.75e−3	80	203	81	80	
	TRON	1.63e−5	2e−1	1.50e−2	422	437	422	849	
palmer5d	IPOPT	1.55e−12	0e0	8.73e1	1	2	2	1	
	TRON	3.09e−8	4e−3	8.73e1	4	4	4	6	
palmer5e	IPOPT	6.24e−3	8e0	2.07e−2	10000	69390	10001	10000	iter.
	TRON	5.39e−4	6e0	3.87e−2	9967	10000	9967	19948	prog.
palmer6a	IPOPT	7.28e−11	1e−1	5.59e−2	288	701	289	288	
	TRON	1.78e−7	2e−1	5.59e−2	372	384	372	747	
palmer6e	IPOPT	7.06e−13	6e−2	2.24e−4	28	64	29	28	
	TRON	7.00e−7	1e−1	2.24e−4	177	207	177	367	
palmer7e	IPOPT	2.12e3	7e0	6.46e0	10000	40833	10001	10000	iter.
	TRON	3.06e−6	5e−1	1.02e1	944	991	944	1908	
palmer8a	IPOPT	4.32e−13	4e−2	7.40e−2	86	172	87	86	
	TRON	4.15e−7	4e−2	7.40e−2	75	75	75	148	
palmer8e	IPOPT	1.56e−11	1e−2	6.34e−3	28	44	29	28	
	TRON	9.79e−7	4e−2	6.34e−3	74	82	74	149	
pentdi	IPOPT	4.44e−16	2e−2	−7.50e−1	18	19	19	18	
	TRON	0.00e0	4e−3	−7.50e−1	2	2	2	2	

Continued on next page

Table 7 – continued from previous page

Problem	Solver	KKT	time	$f(x^\star)$	#iter	$\#f$	$\#g$	$\#H$	fail
probpenl	IPOPT	2.62e−5	1e2	−9.32e4	2548	2989	2549	2548	xfail.
	TRON	3.24e−6	1e−2	3.99e−7	2	2	2	2	
pspdoc	IPOPT	1.11e−16	8e−3	2.41e0	8	16	9	8	
	TRON	4.33e−14	5e−3	2.41e0	8	8	8	14	
qr3dls	IPOPT	3.80e−14	1e−1	1.34e−21	50	115	51	50	
	TRON	2.61e−10	4e−1	4.66e−18	92	109	92	190	
qrtquad	IPOPT	3.18e−11	2e−2	−3.65e6	30	48	31	30	
	TRON	3.16e−6	1e2	−3.65e6	62	178	50	151	prog.
qudlin	IPOPT	3.55e−15	2e−2	−7.20e3	53	101	54	53	
	TRON	0.00e0	2e−3	−7.20e3	2	2	2	2	
s368	IPOPT	1.55e−16	1e−1	3.01e−20	7	8	8	7	
	TRON	0.00e0	9e−3	0.00e0	1	1	1	0	
scon1dls	IPOPT	4.83e−12	1e0	1.65e−16	458	2379	459	458	
	TRON	2.91e−4	1e2	8.06e−4	8830	9757	8830	18061	cpu.
sim2bqp	IPOPT	2.95e−18	4e−3	−9.99e−9	8	9	9	8	
	TRON	0.00e0	3e−3	0.00e0	2	2	2	2	
simbqp	IPOPT	2.97e−18	8e−3	−9.99e−9	8	9	9	8	
	TRON	0.00e0	3e−3	0.00e0	2	2	2	2	
sineali	IPOPT	1.70e−2	6e0	−1.90e3	10000	26766	10001	10000	iter.
	TRON	4.44e−6	1e2	−1.90e3	20	33	9	50	cpu.
torsion-1	IPOPT	1.59e−16	1e−1	−4.18e−1	17	18	18	17	
	TRON	8.52e−10	1e2	−4.18e−1	35	123	24	81	prog.
torsion-2	IPOPT	1.90e−16	2e−1	−4.18e−1	18	19	19	18	
	TRON	9.13e−11	4e−1	−4.18e−1	27	34	27	55	
torsion-3	IPOPT	2.33e−16	2e−1	−4.18e−1	18	19	19	18	
	TRON	2.75e−11	5e−1	−4.18e−1	32	43	32	66	
yfit	IPOPT	4.20e−12	2e−2	6.67e−13	49	97	50	49	
	TRON	1.35e−8	3e−2	1.63e−12	63	81	63	132	

References

Barzilai, J., and J. M. Borwein. 1988. Two-point step size gradient methods. IMA J. Numer. Anal. 8:141–148.

Beister, M., D. Kolditz, and W. Kalender. 2012. Iterative Reconstruction Methods in X-ray CT. Physica Medica 28:94–108.

Bertsekas, D. P. 1982. Projected Newton Methods for Optimization Problems with Simple Constraints. SIAM J. Control Optim. 20:221–246.

Birgin, E. G., and J. M. Martínez. 2002. Large-Scale Active-Set Box-Constrained Optimization Method with Spectral Projected Gradients. Sci. (80-.). 23:101–125.

Birgin, E. G., J. M. Martínez, and M. Raydan. 2001. Algorithm 813: SPGsoftware for convex-constrained optimization. ACM Trans. Math. Softw. 27:340–349.

Bonettini, S., R. Zanella, and L. Zanni. 2009. A scaled gradient projection method for constrained image deblurring. Inverse Probl. 25:015002.

Boyd, S., and L. Vandenberghe. 2010. Convex Optimization, vol. 25. New York, NY, USA: Cambridge University Press.

Byrd, R. H., P. Lu, J. Nocedal, and C. Zhu. 1995. A Limited Memory Algorithm for Bound Constrained Optimization. SIAM J. Sci. Comput. 16:1190–1208.

Cooley, J. W., and J. W. Tukey. 1965. An Algorithm for the Machine Calculation of Comples Fourier Series. Math. Comput. 19:297–301.

Dai, Y. H., and R. Fletcher. 2005. Projected Barzilai-Borwein methods for large-scale box-constrained quadratic programming. Numer. Math. 100:21–47.

di Serafino, D., V. Ruggiero, G. Toraldo, and L. Zanni. 2017. On the steplength selection in gradient methods for unconstrained optimization. Tech. rep., Optimization Online. URL http://www.optimization-online.org/DB_HTML/2017/01/5832.html.

Fong, D. C.-L., and M. A. Saunders. 2011. LSMR: An Iterative Algorithm for Sparse Least-Squares Problems. SIAM J. Sci. Comput. 33:2950–2971.

Frassoldati, G., L. Zanni, and G. Zanghirati. 2008. New adaptive stepsize selections in gradient methods. J. Ind. Manag. Optim. 4:299–312.

Gafni, E. M., and D. P. Bertsekas. 1984. Two-Metric Projection Methods for Constrained Optimization. SIAM J. Control Optim. 22:936–964.

Golkar, M. A. 2013. Fast Iterative Reconstruction in X-ray Tomography Using Polar Coordinates. Master's thesis, École Polytechnique de Montréal.

Gould, N. I. M., M. E. Hribar, and J. Nocedal. 2001. On the Solution of Equality Constrained Quadratic Programming Problems Arising in Optimization. SIAM J. Sci. Comput. 23:1376–1395.

Gould, N. I. M., D. Orban, and T. Rees. 2013. Projected Krylov methods for saddle-point systems. Cah. du GERAD G-2013-23, GERAD, 1–26.

Goussard, Y., M. A. Golkar, A. Wagner, and M. Voorons. 2013. Cylindrical coordinate representation for statistical 3D CT reconstruction. 12th Int. Meet. Fully Three-Dimensional Image Reconstr. Radiol. Nucl. Med. pp. 138–141.

Hamelin, B. 2009. Accélération d'une approche régularisée de reconstruction en tomographie à rayons X avec réduction des artéfacts métalliques. Ph.D. thesis, École Polytechnique de Montréal.

Herman, G. T., and S. W. Rowland. 1973. Three Methods for Reconstructing Objects From X-Rays: A Comparative Study. Comput. Graph. Image Process. 2:151–178.

Hestenes, M. R., and E. Stiefel. 1952. Methods of conjugate gradients for solving linear systems. J. Res. Natl. Bur. Stand. (1934). 49:409–436.

Lin, C.-J., and J. J. Moré. 1999. Newton's method for large bound-constrained optimization problems. SIAM J. Optim. 9:1100–1127.

Luenberger, D. G., and Y. Ye. 2008. Linear and Nonlinear Programming. Springer.

Paige, C. C., and M. A. Saunders. 1975. Solution of Sparse Indefinite Systems of Linear Equations. SIAM J. Numer. Anal. 12:617–629.

Paige, C. C., and M. A. Saunders. 1982. LSQR: An Algorithm for Sparse Linear Equations and Sparse Least Squares. ACM Trans. Math. Softw. 8:43–71.

Pan, X., E. Y. Sidky, and M. Vannier. 2009. Why Do Commercial CT Scanners still Employ Traditional, Filtered Back-projection for Image Reconstruction? Inverse problems 25. URL http://stacks.iop.org/0266-5611/25/i=12/a=123009. Ref.: 123009.

Petersen, K. B., and M. S. Pedersen. 2007. The Matrix Cookbook. Citeseer 16:1–66.

Sauer, K., and C. Bouman. 1993. A Local Update Strategy for Iterative Reconstruction from Projections. IEEE Trans. Signal Process. 41:534–548.

Schmidt, M., D. Kim, and S. Sra. 2011. Projected Newton-type Methods in Machine Learning. In Optim. Mach. Learn., 305–330. Cambridge, MA, USA: MIT Press.

Segars, W. P., M. Mahesh, T. J. Beck, E. C. Frey, and B. M. W. Tsui. 2008. Realistic CT simulation using the 4D XCAT phantom. Med. Phys. 35:3800–3808.

Steihaug, T. 1983. The Conjugate Gradient Method and Trust Regions in Large Scale Optimization. SIAM J. Numer. Anal. 20:626–637.

Thibaudeau, C., J.-D. Leroux, R. Fontaine, and R. Lecomte. 2013. Fully 3D iterative CT reconstruction using polar coordinates. Med. Phys. 40:111904.

Wächter, A., and L. T. Biegler. 2006. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Math. Program. 106:25–57.

Wirgin, A. 2004. The inverse crime. arXiv.org 1–10.

Zhou, B., L. Gao, and Y.-H. Dai. 2006. Gradient Methods with Adaptive Step-Sizes. Comput. Optim. Appl. 35:69–86.

Zhu, C., R. H. Byrd, P. Lu, and J. Nocedal. 1997. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. ACM Trans. Math. Softw. 23:550–560.