We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

6,900
Open access books available

185,000
International authors and editors

200M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities



WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com



Chapter 3

Survey of Computational Methods for Inverse Problems

Sergey Voronin and Christophe Zaroli

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.73332

Abstract

Inverse problems occur in a wide range of scientific applications, such as in the fields of signal processing, medical imaging, or geophysics. This work aims to present to the field practitioners, in an accessible and concise way, several established and newer cutting-edge computational methods used in the field of inverse problems—and when and how these techniques should be employed.

Keywords: inverse problems, matrix factorizations, regularization, parameter estimation, model appraisal, seismic tomography

1. Introduction

In this work, we aim to survey several techniques useful to a practitioner in the field of inverse problems, where the solution to a vector of interest is given through a linear system $Ax = b$ or through a set of nonlinear equations $F(x) = 0$. In our presentation below, we review both classical results and newer approaches, which the reader may not be familiar with. In particular, this chapter offers entries on the following material:

- Matrix factorizations and sparse matrices
- Direct solves and pivoted factorizations
- Least squares problems and regularization
- Nonlinear least squares problems
- Low-rank matrix factorizations and randomized algorithms
- An introduction to Backus-Gilbert inversion

IntechOpen

© 2018 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited. CC BY

2. Notation

In this chapter, we use the norm $\|\cdot\|$ to refer to the spectral or operator norm, and $\|\cdot\|_p$ to refer to the ℓ_p norm. We make frequent use of the QR decomposition and the SVD (singular value decomposition). For any $M \times N$ matrix A and (ordered) subindex sets J_r and J_c, $A(J_r, J_c)$ denotes the submatrix of A obtained by extracting the rows and columns of A indexed by J_r and J_c, respectively; and $A(:, J_c)$ denotes the submatrix of A obtained by extracting the columns of A indexed by J_c. We will make use of the covariance and the variance matrix, which we define as follows in terms of the expected value:

$$\mathrm{cov}(x, y) = E[x - E[x])(y - E[y])^T] \text{ and } \mathrm{var}(x) = \mathrm{cov}(x, x).$$

By "Diag," we refer to a diagonal matrix with nonzeros only on the diagonal. We make use of the so-called GIID matrices. These are matrices with independent and identically distributed draws from the Gaussian distribution. In the Octave environment (which we frequently reference), these can be obtained with the "randn" command. We assume the use of real matrices, although most techniques we describe extend to the complex case.

3. Matrix factorizations and sparse matrices

Let A be an $M \times N$ matrix with real or complex entries, and set $r = \min(M, N)$. We will make use of the singular value decomposition (SVD) of a real matrix $A \in \mathbb{R}^{M \times N}$: if A is of rank r, then there exist $U \in \mathbb{R}^{m \times r}$, $V \in \mathbb{R}^{N \times r}$ and $\sum \in \mathbb{R}^{r \times r}$ such that

1. $U^T U = I$, $V^T V = I$,
2. $\sum = \mathrm{Diag}(\sigma_1, \sigma_2, \ldots, \sigma_r) \in \mathbb{R}^{r \times r}$ is a diagonal matrix with $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$, and
3. $A = U \Sigma V^T$.

This is known as the economic form of the SVD [1]. For $1 \leq i \leq \min\{M, N\}$, the i-th largest singular value of A is defined to be σ_i, with $\sigma_j = 0$ for $j = r + 1, \ldots, \min\{M, N\}$ whenever $r < \min\{M, N\}$. The generalized inverse of $A \in \mathbb{R}^{m \times N}$ with $SVD A = U \Sigma V^T$, is defined as $A^+ = V \Sigma^{-1} U^T$ (and $\sum^{-1} = \mathrm{Diag}\left(\sigma_1^{-1}, \sigma_2^{-1}, \ldots, \sigma_r^{-1}\right) \in \mathbb{R}^{r \times r}$). By a rank deficient matrix, we imply a nonlinear decay of singular values $\{\sigma_i\}$. In this case, the numerical rank of A may be smaller than the actual rank due to the use of finite precision arithmetic.

The (compact) QR-factorization of A takes the form

$$\underset{m \times n}{A} \quad \underset{n \times n}{P} \quad = \quad \underset{m \times r}{Q} \quad \underset{r \times n}{R}, \tag{1}$$

where P is a permutation matrix, Q has orthonormal columns, and R is upper triangular. The permutation matrix P can more efficiently be represented via a vector $J_c \in \mathbb{Z}_+^n$ of indices such that $P = I(:, J_c)$ where I is the $n \times n$ identity matrix. The factorization (1) can then be written as:

http://dx.doi.org/10.5772/intechopen.73332

$$\begin{array}{ccccc} A(:,J_c) & = & Q & & R \\ m \times n & & m \times r & & r \times n \end{array} \tag{2}$$

Another commonly used decomposition is the pivoted LU:

$$\begin{array}{ccccc} A(:,J_c) & = & L & & U. \\ m \times n & & m \times r & & r \times n \end{array} \tag{3}$$

with L a lower triangular and U an upper triangular matrix. In the Octave environment, these decompositions can be constructed, respectively, via the commands $([Q,R,I] = qr(A,0);$ $[L,U,I] = lu(A)$. The matrix P does not need to be explicitly formed. Instead, vector I gives the permutation information. The relation between the two in Octave is given by the command $\mathrm{P}(:,\mathrm{I}) = \mathrm{eye}\,(\mathrm{length}(\mathrm{I}))$.

Many matrices in applications turn out to be sparse. They can be stored more efficiently, without the need to store all $m \times n$ elements. The simplest sparse format is the so called coordinate sparse format, common to, e.g., the Octave environment. In this format, we store the integers row, column, and floating point value for each nonzero of the sparse matrix A: a set of triplets of the form (i,j,v). However, we do not need to store all the row and column indices of the nonzero elements. Below, we summarize the two commonly used sparse formats for an example matrix.

$$A = \begin{bmatrix} 0 & 6 & 3 & 0 \\ 1 & 0 & 8 & 0 \\ 7 & 0 & 0 & 2 \end{bmatrix}$$

The compressed column and row formats for this matrix are given by the vectors:

$$i_c = [1,2,0,0,1,2], \quad p_c = [0,2,3,5,6], \quad d_c = [1,7,6,3,8,2],$$

and

$$i_r = [2,1,0,2,3,0], \quad p_r = [0,2,4,6], \quad d_r = [3,6,1,8,2,7].$$

In the compressed column format, the d_c array stores the nonzero elements, scanned row by row. The array i_c stores the row index of the corresponding data element, and the array p_c stores

function y = mat_mult (A, x)	**function y = mat_trans_mult (A, x)**
y = **zeros** (m, 1);	y = **zeros** (n, 1);
for i = 1:m	**for** i = 1:m
for j = pr (i): pr (i + 1)	**for** j = pr (i): pr (i + 1)
y (i) = y (i) + dr (j) * × (i r (j));	y (i r (j)) = y (i r (j)) + dr (j) * × (i);
end	**end**
end	**end**
end	**end**

the index of the start of each column in the data array d_c. Similarly, for the compressed row format, all the column indices of nonzeros are given, but the row information is compressed by giving in p_r, the index of the start of each row in d_r. Moreover, if needed, the three vectors for the sparse representation above can be further compressed, with, e.g., lossless compression techniques, such as arithmetic coding [2]. BLAS operations on sparse matrices can be performed directly using these storage formats. Below, we list the pseudocode for the operations $y_1 = Ax_1$ and $y_2 = A^T x_2$ for a $m \times n$ sparse matrix A stored with compressed row format.

4. Direct solves

Given a linear system $Ax = b$ with a square matrix A which is invertible $(\det(A) \neq 0)$, the solution x can be constructed through the inverse of A, built up using Gaussian elimination. For relatively small systems, the construction of such solutions is often desired over least squares formulations, when a solution is known to exist. Typically elimination is used to construct the factorization of A into a QR or LU decomposition. The construction of factorizations (QR, LU) with column pivoting can be applied to system solves involving rank deficient matrices. As an example, consider the pivoted QR factorization $AP = QR$. Here $AP = A(:; I)$, a rearrangement of the columns of A upon multiplication with permutation matrix P. Plugging into $Ax = b$ yields $QRP^T x = b \Rightarrow QRy = b \Rightarrow Ry = Q^T b$, which is an upper triangular system, and can be solved by back substitution. A simple permutation $P^T x = y \Rightarrow x = Py$ yields the solution x.

Similarly, suppose we have the pivoted LU factorization $AP = LU$. Then, plugging into $Ax = b$ yields $LUP^T x = b$. Next, set $z = UP^T x = Uy$ with $y = P^T x$. Then $Lz = b$ (which is a lower triangular system) can be solved by forward substitution for z, while $Uy = z$ can be solved by back substitution for y. Again applying a permutation matrix to $x = Py$ yields the result. Notice that multiplying by P can be done efficiently, simply be re-arranging the elements of y. The implementations of the back substitution and forward substitution algorithms are given below.

% Solve Lz = b

```
function z = fwd_sub (L, b)
  n = length (b);
  z = zeros (n, 1);
  for i = 1:n
    z (i) = (b (i) − L(i,:) * z) /L(i, i);
  end
end
```

% Solve Uy = z

```
function y = back_sub (U, z)
  n = length (z);
  y = zeros (n, 1);
  for i = n: −1:1
    y (i) = (z (i) − U(i,:) y) /U(i, i);
  end
end
```

http://dx.doi.org/10.5772/intechopen.73332

5. Regularization

5.1. Least squares

Prior to discussing two-norm or what is more commonly known as Tikhonov regularization, we mention the least squares problem:

$$\overline{x}_{\text{lsq}} = \arg\min_x \|Ax - b\|_2^2 \tag{4}$$

This formulation arises due to the noise in the right hand side b, in which case it does not make sense to attempt to solve the system $Ax = b$ directly. Instead, if we have an estimate for the noise norm (that is, $b = \overline{b} + e$ with unknown noise vector e, but we can estimate $\|e\|_2$), then we could seek *a* solution x such that $\|Ax - b\|_2 \approx \|e\|_2$. Let us now look at the solution of (4) in more detail. As the minimization problem is convex and the functional quadratic, we obtain the minimum by setting the gradient of the functional to zero:

$$\nabla_x \|A_x - b\|_2^2 = 0 \Rightarrow A^T Ax = A^T b \tag{5}$$

A common choice of solution to the quadratic Eq. (5) would be directly through the generalized inverse:

$$x = \left(A^T A\right)^+ A^T b = \left(V\Sigma^{-2}V^T\right)V\Sigma U^T b = A^+ b, \tag{6}$$

because of all the solutions to $A^T Ax = A^T b$, $A^+ b$ has the smallest ℓ_2-norm: $A^T Ax = A^T b$ if and only if $x = A^+ b + d$ for some $d \in \ker\left(A^T A\right) = \text{range}\left(A^T A\right)^\perp = \text{range}\left(A^+\right)^\perp$, and

$$\left\|A^+ b + d\right\|^2 = \left\|A^+ b\right\|^2 + 2d^T\left(A^+ b\right) + \|d\|^2 = \left\|A^+ b\right\|^2 + \|d\|^2\left\|A^+ b\right\|^2.$$

Typically, the least squares problem is solved by an iterative method such as conjugate gradients (CG) or related techniques such as LSQR. Whichever way the solution to the normal equations in (5) is obtained, it will be close $A^+ b$ and share its properties.

First, if A has small singular values, the norm of the solution $A^+ b = V\Sigma^{-1}U^T b$ will be very large because of the Σ^{-1} matrix. Another disadvantage of (4) is that the solution $A^+ b$ will be very sensitive to any noise in b or even in A (if any approximations in the calculations are used). Suppose the noise vector e behaves like white noise. Its different entries are uncorrelated, each having mean 0 and standard deviation v. If, in addition, the elements of $\overline{b}$ and e are uncorrelated, we have:

$$\text{var}(e) = E\left[(e - E[e])(e - E[e])^T\right] = E\left[ee^T\right] = v^2 I,$$

and

$$\mathrm{var}(b) = E\Big[(b - E[b])(b - E[b])^T\Big] = E\big[ee^T\big] = \nu^2 I.$$

We may then estimate the norm of the variance of the solution vector:

$$\mathrm{var}(x) = \mathrm{var}\big(A^+ b\big) = \mathrm{var}\big(\big(V\Sigma^{-1}U^T\big)b\big) = \big(V\Sigma^{-1}U^T\big)\mathrm{var}(b)\Big(U\big(\Sigma^{-1}\big)^T V^T\Big) = \nu^2 V\Sigma^{-2}V^T$$

$$\Rightarrow \|\mathrm{var}(x)\|_2 = \nu^2\big\|V\Sigma^{-2}V^T\big\|_2 = \frac{\nu^2}{\sigma_{\min}^2},$$

where $\sigma_{\min}$ is the smallest magnitude singular value of A. We can clearly see that when A has an appreciable decay of singular values (such that $\sigma_{\min}$ is small relative to σ_1), the solution $x = A^+ b$ will be sensitive to data errors. For these reasons, adding additional terms (regularization) to the optimization problem is often necessary.

5.2. Tikhonov regularization

Having discussed the least squares approach we turn our attention to the simplest form of Tikhonov Regularization:

$$\overline{x}_{\mathrm{tik}} = \arg\min_x \left\{ \|Ax - b\|_2^2 + \lambda\|x\|_2^2 \right\} \tag{7}$$

where $\lambda > 0$ is a scalar regularization parameter, which controls the tradeoff between the solution norm $\|x\|_2$ and the residual fit $\|Ax - b\|_2$. Since the functional in brackets is convex, we can get the solution by again setting the gradient to zero:

$$\nabla_x \left\{ \|Ax - b\|_2^2 + \lambda\|x\|_2^2 \right\} = 0 \Rightarrow 2A^T(A\overline{x} - b) + 2\lambda\overline{x} = 0 \Rightarrow \big(A^T A + \lambda I\big)\overline{x} = A^T b \tag{8}$$

If we plug in the SVD of $A = U\Sigma V^T$, we get:

$$\begin{aligned}
\overline{x} &= \Big(\big(U\Sigma V^T\big)^T\big(U\Sigma V^T\big) + \lambda I\Big)^{-1} A^T b \\
&= \Big(V\big(\Sigma^T\Sigma + \lambda I\big)^{-1} V^T\Big) V\Sigma^T U^T b \\
&= V\big(\Sigma^T\Sigma + \lambda I\big)^{-1}\Sigma^T U^T b \\
&= V\,\mathrm{Diag}\left(\frac{\sigma_i}{\sigma_i^2 + \lambda}\right) U^T b = VDU^T b
\end{aligned}$$

We see that the effect of the regularization is to filter the small singular values σ_i, by replacing each σ_i by $\frac{\sigma_i}{\sigma_i^2 + \lambda}$ which prevents the singular values smaller than λ from dominating the solution. If we now compute the norm of the solution variance, we obtain:

$$\operatorname{var}(\overline{x}) = \operatorname{var}\left(VDU^T b\right) = \left(VDU^T\right)\operatorname{var}(b)\left(UDV^T\right) = v^2 VD^2 V^T$$
$$\Rightarrow \quad \|\operatorname{var}(\overline{x})\|_2 = v^2\left\|VD^2V^T\right\|_2 = v^2\left\|D^2\right\|_2 \leq \frac{v^2}{4\lambda}.$$

The result follows because the function $h(t) := \frac{t}{t^2+\lambda}$ satisfies $h'(t) = 0$ at $t = \pm\sqrt{\lambda}$ with $h\left(\sqrt{\lambda}\right) = \frac{1}{2\sqrt{\lambda}}$. Thus, the solution to the system from (8) (even if obtained from a CG type scheme after a finite number of iterations) relieves the problems due to small singular values and noise, which affect the solution from (5).

In practice, a slight generalization of (7) is performed by adding a regularization matrix L:

$$\overline{x}_{\text{tikL}} = \arg\min_x \left\{\|Ax - b\|_2^2 + \lambda_1\|x\|_2^2 + \lambda_2\|Lx\|_2^2\right\} \tag{9}$$

Generally, L is some kind of sharpening (difference) operator. The penalization of the term $\|Lx\|$, thus results in smoothing. If the coordinate system x is one-dimensional, it could take the form:

$$L_1 = \begin{bmatrix} -1 & 1 & \dots & 0 & 0 \\ 0 & -1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & -1 & 1 \end{bmatrix}$$

When the coordinate system corresponding to the model vector x is higher dimensional, L will be correspondingly more complicated and will generally have block structure. Note that the solution to (9) can be obtained through the linear equations:

$$\left(A^T A + \lambda_1 I + \lambda_2 L^T L\right)\overline{x} = A^T b \tag{10}$$

and can also be cast as an augmented least squares system and solved through its corresponding augmented normal equations:

$$\overline{x} = \arg\min_x \left\| \begin{bmatrix} A \\ \sqrt{\lambda_1} I \\ \sqrt{\lambda_2} L \end{bmatrix} x - \begin{bmatrix} b \\ 0 \\ 0 \end{bmatrix} \right\|_2^2 \Rightarrow \begin{bmatrix} A \\ \sqrt{\lambda_1} I \\ \sqrt{\lambda_2} L \end{bmatrix}^T \begin{bmatrix} A \\ \sqrt{\lambda_1} I \\ \sqrt{\lambda_2} L \end{bmatrix} \overline{x} = \begin{bmatrix} A \\ \sqrt{\lambda_1} I \\ \sqrt{\lambda_2} L \end{bmatrix}^T \begin{bmatrix} b \\ 0 \\ 0 \end{bmatrix}$$

This is an important point with regards to implementation, as it means that codes which solve the normal equations for standard least squares can be readily modified to solve (9). If L is a smoothing operator, the parameters λ_1 and λ_2 effect the degree of norm damping and model smoothing, respectively. Notice that increasing λ_2 from zero also changes the solution norm, so λ_1 may need to be altered to compensate.

5.3. Sparse regularization and generalized functional

The ℓ_2 penalty in (7) has the effect of penalizing the solution norm in a way which encourages all coefficients to be small (as λ is increased). For very large λ, only $x = 0$ would result in the required minimum but for modest values (below, e.g., $\|A^T b\|_\infty$), the effect would be to force all solution coefficients to have smaller magnitudes with increasing λ, but would not make any of them explicitly zero. In many applications, a sparse solution is sought (where, a significant percentage of the coefficients of x are zero). A so-called ℓ_0 measure is an indicator function for the number of nonzeros of x. This measure is not a norm, as it does not satisfy, e.g., the basic triangle inequality. Constraining the ℓ_0 measure (e.g., $\|Ax - b\| < \epsilon$, $\min \|x\|_0$), leads to a combinatorially difficult optimization problem.

A key insight of compressive sensing is that the minimization with respect to the ℓ_0 measure and the convex ℓ_1 norm (the sum of the absolute values of the components of a vector) produce the same result (the sparsest solution) under some strong conditions (i.e., RIP, restricted isometry property) on A [3]. In practice, a nonrandom A from a physical application would not satisfy the RIP conditions. On the other hand, the minimization with respect to the ℓ_1 norm (and the ℓ_p-norms for $0 < p < 2$, convex only for $p \geq 1$) produce sparse solutions (but not-necessarily the sparsest one at a given residual level). Sample illustrations for the 2d case are given in **Figure 2**, where we can observe that in two dimensions, the minimization of the ℓ_p norm for $p \leq 1$ results in one of the two components equal to zero. To account for the possibility of employing a sparse promoting penalty and also for more general treatment of the residual term, which we discuss more below, we will consider the two-parameter functional [4]:

$$F_{l,p}(x) = \|Ax - b\|_l^l + \lambda \|x\|_p^p = \sum_{i=1}^{m} \left| \sum_{j=1}^{m} A_{ij} x_j - b_i \right|^l + \lambda \sum_{i=1}^{n} |x_i|^p, \tag{11}$$

with $\tilde{F}_p = F_{2,p}$ (with $l = 2$ being the most-common residual-based penalty). For $p < 2$, the functional $\tilde{F}_p$ is not differentiable. This means that the minimum value cannot be obtained by setting the gradient equal to zero as for ℓ_2-based minimization. A particularly well-studied example is the ℓ_1 case, which is the closest convex problem to the ℓ_0 penalty. Convexity of $\tilde{F}_1(x)$

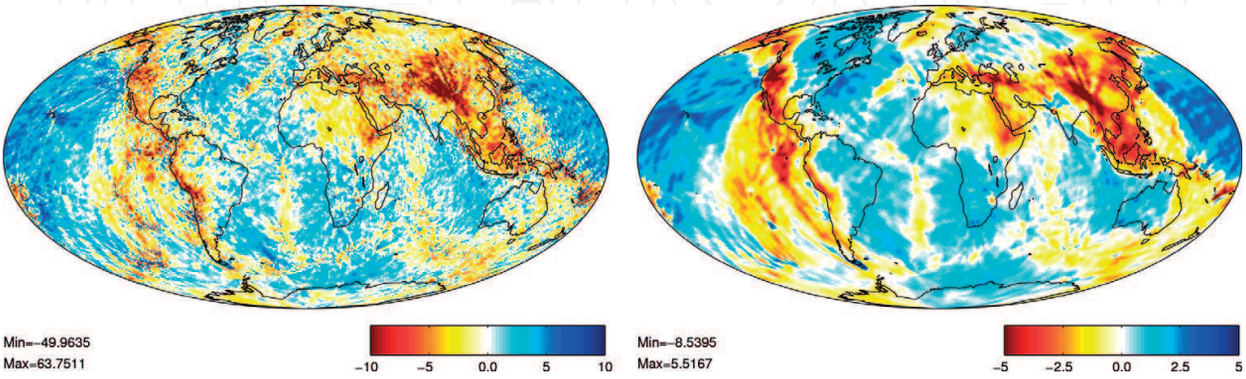


Figure 1. Comparison of geophysical models recovered via (10) with $\lambda_2 = 0$ and $\lambda_2 > 0$.

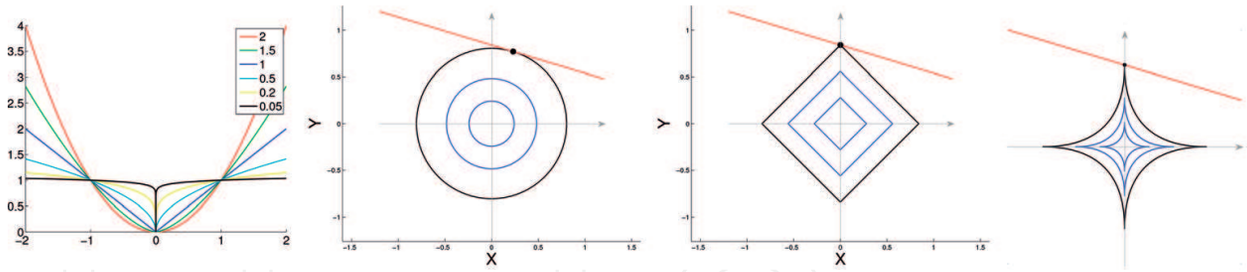


Figure 2. Illustration of family of functions $|y|^p$ for $p \in (0, 2)$ and sample solutions to $ax_1 + bx_2 = c$ subject to min $\|x\|_p$ for $p = 2, 1, 0.5$.

guarantees that any local minimizer is necessarily a global one. In this case, an algorithm can be constructed which decreases the functional value and tends to the (global) minimizer of $\tilde{F}_1(x)$. One such method is called the iterative soft thresholding algorithm (ISTA) and relies on the soft thresholding function $S_\tau(x)$, defined as:

$$(\mathbb{S}_\tau(x))_k = \text{sgn}(x_k)\max\{0, |x_k| - \tau\}, \quad \forall k = 1, \ldots, n, \forall x \in \mathbb{R}^n.$$

The benefit of this function is two-fold: it explicitly sets small components of x to zero (promoting sparsity) and is continuous (unlike the related hard thresholding function which simply zeros out all components smaller than τ in absolute value). The soft thresholding function satisfies a useful identity:

$$\mathbb{S}_\tau(b) = \arg\min_x \left\{ \|x - b\|_2^2 + 2\tau\|x\|_1 \right\},$$

which is utilized with a surrogate functional approach to construct the ISTA scheme:

$$x^{n+1} = \mathbb{S}_\tau\left(x^n + A^T b - A^T A x^n\right)$$

This algorithm converges to the ℓ_2 minimizer for any initial guess and with $\|A\|_2$ (the spectral norm of A) being less than 1 (which can be accomplished simply by rescaling). The spectral norm of a matrix A can easily be estimated using so called power iteration [1]. Let us assume that A is square. If it is not we can take the matrix $A^T A$ in it's place and take the square root of the eigenvalue found to be the estimate for the spectral norm. If we take a vector x^0 and write it as a linear combination of eigenvectors of A, then $x^0 = \alpha_1 v_1 + \ldots + \alpha_n v_n$. It follows that the iterative computation $x^m = Ax^{m-1}$ yields (plugging in $Av_k = \lambda_k v_k$):

$$\alpha_1 \lambda_1^m v_1 + \ldots + \alpha_n \lambda_n^m v_n = \lambda_1^m \left[\alpha_1 v_1 + \alpha_2 \left(\frac{\lambda_2}{\lambda_1}\right)^m v_2 + \ldots + \lambda_n \left(\frac{\lambda_n}{\lambda_1}\right)^m v_n \right] \Rightarrow \lim_{m\to\infty} \frac{x^m}{\lambda_1^m} = \alpha_1 v_1,$$

a scalar multiple of the dominant eigenvector. A simple computation yields the dominant eigenvalue. In practice, a much faster converging scheme called FISTA (Fast ISTA) [5] is utilized, which is a slight reformulation of ISTA applied to a linear combination of two previous iterates:

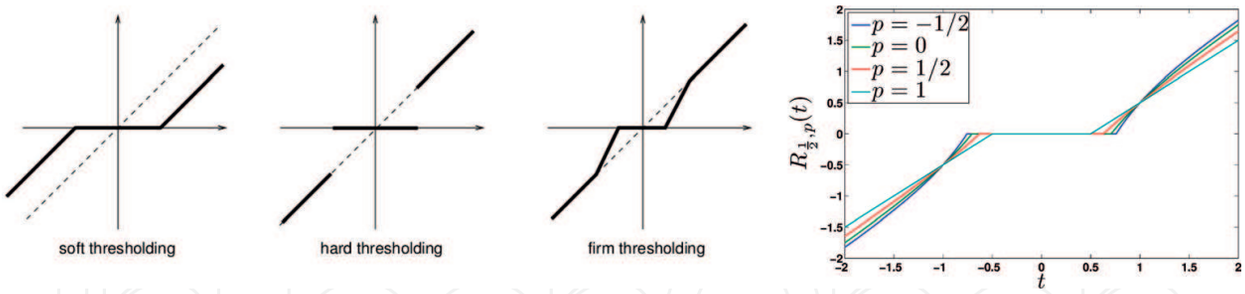


Figure 3. Illustrations of different thresholding functions [6, 7].

$$x^{n+1} = \mathbb{S}_\tau\left(y^n + A^T b - A^T A y^n\right), \quad y^n = x^n + \frac{t_{n-1} - 1}{t_n}\left(x^n - x^{n-1}\right), \quad t_{n+1} = \frac{1 + \sqrt{1 - 4t_n^2}}{2}, \tag{12}$$

where $t_1 = 1$. This algorithm is simple to implement and readily adapts to parallel architectures. For challenging problems (such as when the decay of singular values of A is rapid), the thresholding function $\mathbb{S}_\tau$ can be slightly altered (see **Figure 3**), but the same iterative scheme (12) can be utilized. Two possible approaches are either to vary the thresholding, starting from soft thresholding and slowly approaching the (discontinuous) hard thresholding function, or to use a function which better mimics hard thresholding away from zero. The use of different thresholding functions alters the optimization problem being solved. Thresholding-based techniques are simple to implement but are not effective in all situations, particularly when only a few iterations are feasible (for example, when A is large). In this case, two interesting approaches are iteratively re-weighted least squares (IRLS) and convolution smoothing [4]. Both techniques replace the nonsmooth part of the functional (namely, the absolute value function $|x|$) by a smooth approximation. Moreover, both techniques have the particular advantage of being able to employ gradient-based methods (such as CG) at each iteration, considerably increasing the per-iteration performance. The IRLS approach is based on the approximation:

$$|x_k| = \frac{x_k^2}{|x_k|} = \frac{x_k^2}{\sqrt{x_k^2}} \approx \frac{x_k^2}{\sqrt{x_k^2 + \epsilon^2}}$$

where in the rightmost term, a small $\epsilon \neq 0$ is used, to insure the denominator is finite, regardless of the value of x_k. The resulting algorithm [4] for the minimization of (11) can be written as:

$$\left(A^T R^n A + (D^n)^T (D^n)\right) x^{n+1} = A^T R^n b$$

with two diagonal iteration dependent matrices D_n and R_n. The diagonal matrix D^n has elements $\sqrt{\frac{1}{2}\lambda p w_k^n}$ and R^n has diagonal elements $l\left|r_i^n\right|^{l-2}$ (for i where $\left|r_i^n\right| < \epsilon$, we can set the entry to $l\epsilon^{l-2}$ with the choice of ϵ user controllable, tuned for a given application). Here, the residuals $r_i^n = (Ax^n - b)_i$ and the iteration dependent weights are given by:

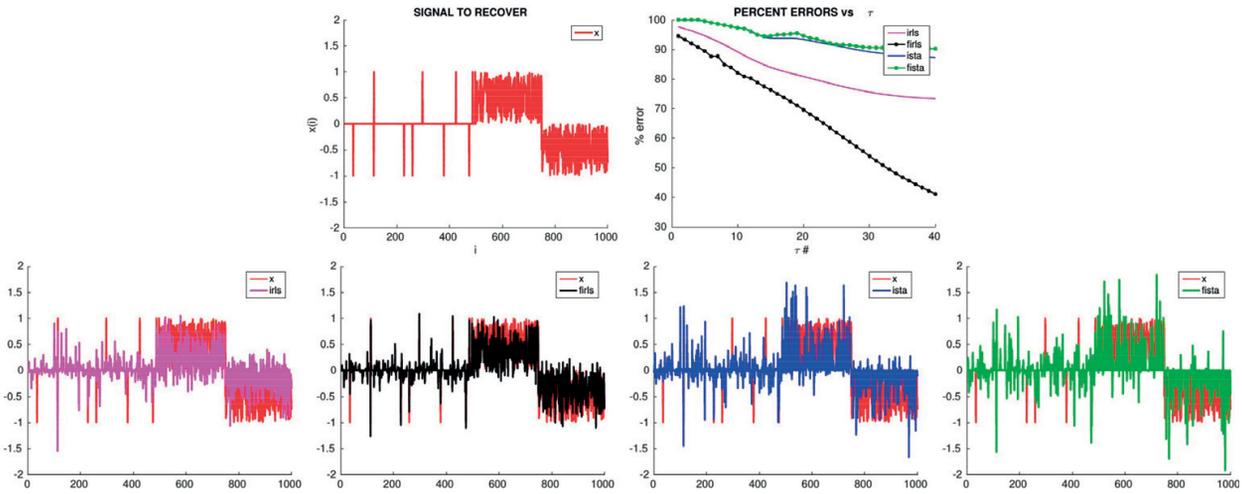


Figure 4. Illustration of half-sparse, half-dense signal recovery with different algorithms.

$$w_k^n = \frac{1}{\left[\left(x_k^n\right)^2 + \epsilon_n^2\right]^{\frac{2-p}{2}}}.$$

The diagonal matrices (or simply the vectors holding the diagonal elements) are updated at each iteration and the system in (6.10) can be solved approximately via a few iterations of CG- or LSQR-based algorithms. Another advantage of the IRLS approach is that the powers p can be made component dependent. This then allows for better inversion of partially sparse signals (if of course, the location of the sparse part can be estimated with some accuracy). An example is illustrated in **Figure 4** and further discussed in [8]. Another approach discussed in [4] is based on a smooth approximation of the absolute value function $f(t) = |t|$ obtained via convolution with a sequence of Gaussian kernels, which have approximately shrinking support. The resulting "conv-CG" method is suitable especially for rapid warm start acceleration.

5.4. Alternate penalty functions and regularization parameter selection

We mention here the classical image deconvolution problem. Given a blurring source g, such as a 2D Gaussian function, we can produce a blurry image s from an unaltered source f via convolution $s = f^* g + n$, where n is some additive noise component. For such situations, a TV (total variation) norm penalty is frequently used, for purposes of noise removal [9]. For a 2-D signal (such as an image), the TV penalty can be written as $V(s) = \sum_{i,j} \sqrt{\left|s_{i+1}, j - s_{i,j}\right|^2 + \left|s_{i,j+1} - s_{i,j}\right|^2}$. Sometimes, the alternate approximation $\sum_{i,j} \left|s_{i+1,j} - s_{i,j}\right| + \left|s_{i,j+1} - s_{i,j}\right|$ is utilized. Various iterative schemes have been developed for such penalty functions [9].

Both the ℓ_2-based approaches and sparsity promoting regularization schemes (as well as TV-norm penalty functionals) utilize one or more regularization parameters. In the case of Tikhonov regularization with smoothing (as in (9)) more than one parameter is present. In this case, the second (smoothing) parameter can generally be set according to the desired smoothing effect, once the first parameter λ_1 is chosen (with a fixed value of λ_2) and then, λ_1 can be

adjusted to achieve a desired solution norm. Thus, we focus here on techniques to adjust λ_1, which we simply refer to as λ.

The standard way to choose the parameter is to use the L-curve technique starting at a large λ (generally a value close to $\left\|A^T b\right\|_\infty$ is a good choice) and decreasing down in logarithmic fashion using the logarithmically spaced sequence:

$$S = \frac{\log(\lambda_{\max}) - \log(\lambda_{\min})}{N-1}; \quad \lambda_i = \exp(\log(\lambda_{\max}) - S(i-1)), \quad i = 1, \ldots, N.$$

The parameter N can vary by application but is typically in the range [5, 10]. Two typical Strategies for parameter selection are employed.

The first is based on a target residual value, typically determined by the estimate of the noise norm. At every λ after the initial value we reuse the previous solution as the initial guess for the CG scheme at the current λ. We can use the solution x_λ for which $\|Ax_\lambda - b\|$ is closest to the desired residual level (or refine further the solution at this λ with more CG iterations).

If however, the target residual norm is not available, other techniques must be used. We discuss a method using the so-called L-curve where for the norm damping problem (7), we plot a curve composed of points $(\log\|Ax_\lambda - b\|, \log\|x_\lambda\|)$ which we can obtain using the same continuation procedure previously discussed. The curve represents the tradeoff between the residual value $\|Ax_\lambda - b\|$ and the solution norm $\|x_\lambda\|$. In practice, neither of these quantities should dominate over the other. Hence, an established strategy is to look for the point of maximum curvature along the L-curve [11]. If we set:

$$\overline{\epsilon} = \log\|x_\lambda\|_p \quad \text{and} \quad \overline{\rho} = \log\|Ax_\lambda - b\|_l, \tag{13}$$

where x_λ is the solution of (11) at the particular value of λ. We can then compute the curvature by the formula:

$$\overline{c}_\lambda = 2\frac{\overline{\rho}'\overline{\epsilon}'' - \overline{\rho}''\overline{\epsilon}'}{\left((\overline{\rho}')^2 + (\overline{\epsilon}')^2\right)^{\frac{3}{2}}}, \tag{14}$$

where the derivative quantities can be approximated via finite differences. We illustrate various plots for a synthetic example in **Figure 5**. In the residual plot, the target residual is taken to be the magnitude of the noise vector norm. We can also see that the lowest percent error

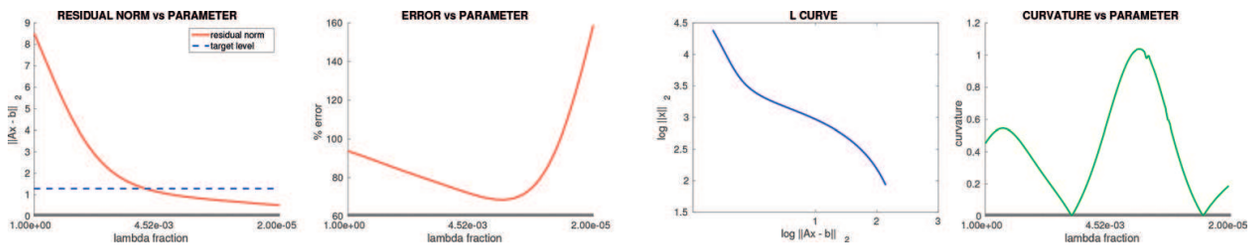


Figure 5. Regularization parameter picking. Set 1: Residuals and percent errors vs. λ fraction (fraction of $\left\|A^T b\right\|_\infty$). Set 2: L-curve and curvature curve as a function of λ fraction.

between $\overline{x}_\lambda$ and the true x occurs at a value of λ roughly corresponding to the highest curvature of the L-curve. In fact, for this example, the curvature method gives a better estimate of good λ than the residual curve technique.

6. Nonlinear least squares (NLS) problems

In many cases, the inverse problem may be posed in terms of a nonlinear function $F(x, t)$ with x a vector of variables, which may be time dependent with parameter t. We first describe, here, the popular Newton-Gauss method for NLS [1]. Let $g(x) = \frac{1}{2}\|r(x)\|^2$ with $r_i(x) = y_i - F(x, t_i)$. Then, the NLS problem takes the form: $\overline{x} = \arg\min_x g(x)$. Setting $\nabla g(x) = 0$, yields with Newton's method:

$$x^{n+1} = x^n - \left[\nabla^2 g(x^n)\right]^{-1} \nabla g(x^n)$$

Expanding the gradient and Hessian of g yields:

$$\nabla g(x) = \sum_{i=1}^{m} r_i(x)\nabla r_i(x) = J^T r(x) \text{ where} J = J[r(x)]$$
$$\nabla^2 g(x) = \sum_{i=1}^{m} \nabla r_i(x)\nabla r_i(x)^T + \sum_{i=1}^{m} r_i(x)\nabla^2 r_i(x) = J^T J + T(x) \approx J^T J.$$

where $T(x) = \sum_{i=1}^{m} r_i(x)\nabla^2 r_i(x)$ and $J[r(x)]_{(i,:)} = \nabla r_i(x)^T = -\nabla F(x, t_i)^T$. The approximation $T(x) \approx 0$ is used in the expression for the Hessian in the Gauss-Newton method, yielding a simple iterative scheme:

$$x^{n+1} = x^n - \left[J_n^T J_n\right]^{-1} J_n^T r_n.$$

Unfortunately, this method is not stable and will typically not converge if initialized far away from a minimum solution [1]. Improvements include the introduction of a step size parameter:

$$x^{n+1} = x^n - \alpha_n \left[J_n^T J_n\right]^{-1} J_n^T r_n$$
$$\alpha_n = \arg\min_\alpha g(x^n - \alpha s^n) \quad \text{with} \quad J_n^T J_n s^n = J_n^T r_n.$$

and of the use of a regularizer (e.g., Levenberg-Marquardt method [1]): where the system ${J_n}^T J_n y = {J_n}^T r_n$ is replaced by an ℓ_2-norm penalty regularized system, $\left({J_n}^T J_n + \lambda I\right)\tilde{y} = {J_n}^T r_n$.

7. Low-rank matrix factorizations

In many applications, there are large matrices with rapidly decaying singular values. In such cases, low-rank matrix approximations like the low-rank SVD are useful for compression,

speed gains, and data analysis purposes. For $A \in \mathbb{R}^{m\times n}$, the low-rank SVD of rank k (with $k < \min(m, n)$) is the optimal matrix approximation of A in the spectral and Frobenius norms. Taking $p = \min(m, n)$, we define the low-rank SVD of rank k by A_k by taking into account only the first $k < p$ singular values and vectors: that is, with $U_k \in \mathbb{R}^{m\times k}$ consisting of the first k columns of U, $\Sigma_k = \mathrm{Diag}(\sigma_1, \ldots, \sigma_k) \in \mathbb{R}^{k\times k}$ consisting of k rows and columns of Σ, and $V_k \in \mathbb{R}^{n\times k}$ consisting of the first k columns of V:

$$A_k = \sum_{j=1}^{k} \sigma_j u_j v_j^T = U_k \sum\nolimits_k V_k^T, \tag{15}$$

$$U_k = [u_1\ u_2 \ldots u_k], \quad V_k = [v_1 v_2 \ldots v_k], \quad \text{and} \quad \sum = \begin{bmatrix} \sigma_1 & 0 & 0 & \cdots & 0 \\ 0 & \sigma_2 & 0 & \cdots & 0 \\ 0 & 0 & \sigma_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & & 0 \\ 0 & 0 & 0 & \cdots & \sigma_k \end{bmatrix}$$

By the Eckart-Young theorem [12]:

$$\|A - A_k\| = \sigma_{k+1},$$

when the error is measured in the spectral norm, and

$$\|A - A_k\|_F = \left(\sum_{j=k+1}^{p} \sigma_j^2 \right)^{1/2}$$

in the Frobenius norm. When $k \ll p$, the matrices U_k, Σ_k and V_k are significantly smaller (cost of storage of all nonzeros is $mk + nk + k$) than the corresponding full SVD matrices U, Σ, and V (cost of storage is $mp + np + p$) and that of A (cost of storage is mn, but only some fraction of this if A is sparse). While the construction of A_k is expensive (requiring in most cases the SVD of A), it can be approximated very accurately via *randomized algorithms*, which requires only the SVD of a smaller matrix. Various randomized algorithms for constructing the low-rank SVD and related factorizations are described in [13]. Techniques for computing the low-rank SVD of a matrix rely on a simple principal. An orthonormal matrix $Q \in \mathbb{R}^{m\times r}$ (with $r = k + l$ where l is a small oversampling parameter, e.g., $l = 5$), is computed such that $QQ^T A \approx A$. If in fact r is large enough so that $QQ^T A = A$, the range of A is a subset of the range of Q. Thus, when $QQ^T A \approx A$, we expect the range of Q to capture a good portion of the range of A, a statement which can be made rigorous with some analysis. In this case, we form the smaller matrix $B = Q^T A$, where $B \in \mathbb{R}^{r\times n}$, possibly much smaller than the $m \times n$ matrix A. Instead of performing the SVD on A, we can obtain the SVD of $B = U\Sigma V^T$. If $A \approx QQ^T A = QB$, then $A \approx (QU)\Sigma V^T$ and the later will form a low-rank SVD approximation for A (if we only take the first k singular vectors and values of the corresponding factorization). Notice that when A is rectangular, the eigen-

decomposition of the BB^{T} or $B^{\mathrm{T}}\,B$ matrices can be used to construct the approximate low-rank approximation of A.

A separate problem is the construction of a suitable matrix Q from A. Again, the idea is to construct as small (in terms of column number) as possible Q with orthonormal columns, such that Q captures a good chunk of the range of A. When A is a matrix of known rank k then (in MATLAB notation), simply setting $Q = qr(A\Omega, 0)$ where $\Omega \in \mathbb{R}^{n\times(k+l)}$ is a GIID matrix, with l a small over-sampling parameter, produces a valid matrix Q for projection. When the tail singular values (those smaller than σ_k) are still expected to be significant, a power sampling scheme turns out to be effective. Instead of setting $Y = A\Omega$ and performing QR of Y, we use the matrix $Y = \left(AA^T\right)^q A\Omega$ with q ≥ 1. Plugging in the SVD of A, we obtain $\left(AA^T\right)^q A = U\Sigma^{2q+1}V^T$, which has the same eigenvectors as A but much faster decaying singular values. Care must be taken when taking powers of matrices, to prevent multiplying matrices whose singular values are greater than one in magnitude. However, when the rank of the matrix A is not known, it is hard to use this approach, since the optimal size of the random matrix Ω to use would not be known. In this situation, a blocked algorithm can be employed [14], where on output with user supplied $\epsilon > 0$ parameter, an orthonormal matrix Q and matrix B are produced such that $\|QB - A\| < \epsilon$ where $B = Q^T A$. Then, any number of standard low-rank matrix factorizations can be computed by operating on the matrix B instead of A. The basic steps of the proposed algorithm are given in **Figure 6**. We note that the resulting Q matrix can be utilized also for purposes of model reduction (e.g., one can use the reduced linear system $Q^T Ax = Q^T b$ as an approximation to the full system $Ax = b$). That is, one can use the reduced linear system $Q^T Ax = Q^T b$ as an approximation to the full system $Ax = b$ (or to replace A and b with the projected values in the least squares formulation); which has applications to e.g. accelerate image deblurring. The construction of Q for large matrices can in practice be done in parallel by employing the algorithm in [13] over row blocks of the matrix, as illustrated in **Figure 6**.

function $[Q, B] = \texttt{randQB_pb}(A, \varepsilon, q, b)$

(1) **for** $i = 1, 2, 3, \ldots$
(2) $\quad \Omega_i = \texttt{randn}(n, b)$.
(3) $\quad Q_i = \texttt{orth}(A\Omega_i)$.
(4) $\quad$ **for** $j = 1 : q$
(5) $\quad\quad Q_i = \texttt{orth}(A^T Q_i)$.
(6) $\quad\quad Q_i = \texttt{orth}(AQ_i)$.
(7) $\quad$ **end for**
(8) $\quad Q_i = \texttt{orth}(Q_i - \sum_{j=1}^{i-1} Q_j Q_j^T Q_i)$
(9) $\quad B_i = Q_i^T A$
(10) $\quad A = A - Q_i B_i$
(11) $\quad$ **if** $\|A\| < \varepsilon$ **then stop**
(12) **end while**
(13) Set $Q = [Q_1 \ \cdots \ Q_i]$ and $B = [B_1^T \ \cdots \ B_i^T]^T$.

Figure 6. A *blocked* and *adaptive* version of the accuracy enhanced QB algorithm proposed in [14].

The rank-k SVD $\left(A_k = U_k \sum_k V_k^T\right)$ of a general $m \times n$ matrix A yields an optimal approximation of rank k to A, both in the operator (spectral) and Frobenius norms. On the other hand, even if A is a sparse matrix, the $m \times k$ and $n \times k$ factors U_k and V_k are typically dense. This means that if the matrix is approximately p percent filled, the matrix will have approximately $N = \left[\frac{p}{100} m \times n\right]$ nonzeros. On the other hand, the rank k SVD will consist of approximately $mk + k + nk$ nonzeros. For growing rank k, this quantity will quickly approach and even exceed N. Thus, even though the low-rank SVD is optimal for a given rank k, the choice of rank may be limited to relatively low values with respect to min(m, n) for sparse matrices, in order to achieve any useful compression ratios. (Of course, the usefulness of low-rank SVD representation is not simply limited to compression; indeed they are useful, e.g., for low-dimensional data projections; but the utility of a low-rank approximation is greatly reduced once the storage size of the factors exceeds that of the original matrix). Yet another aspect of the SVD which may be problematic is the difficulty in interpreting the eigenvectors present in U_k and V_k. While in many applications these have distinct meanings, they are not often easy to interpret for a particular data set.

It is thus plausible, in the above context, to look for factorizations which may not be optimal for rank k, but which may preserve useful properties of A such as sparsity and non-negativity, as well as allow easier interpretation of its components. Such properties may be found in the one- and two-sided interpolative decompositions and the CUR decomposition based on the pivoted QR decomposition. If we stop the QR procedure after the first k iterations, we obtain:

$$A(:, J_c) = \begin{array}{c} m \\ \end{array} \begin{array}{cc} k & r-k \\ \multicolumn{2}{c}{[\,Q_1 \quad Q_2\,]} \end{array} \times \begin{array}{c} k \\ r-k \end{array} \begin{array}{c} n \\ \begin{bmatrix} S_1 \\ S_2 \end{bmatrix} \end{array} = Q_1 S_1 + Q_2 S_2. \tag{16}$$

$$S_1 = \begin{array}{c} k \\ \end{array} \begin{array}{cc} k & n-k \\ \multicolumn{2}{c}{[\,S_{11} \quad S_{12}\,]} \end{array} \text{ and } S_2 = \begin{array}{c} k \\ \end{array} \begin{array}{cc} k & n-k \\ \multicolumn{2}{c}{[\,0 \quad S_{22}\,]} \end{array} \tag{17}$$

$$\text{i.e., } S = \begin{array}{c} \\ k \\ r-k \end{array} \begin{array}{c} \begin{array}{cc} k & n-k \end{array} \\ \begin{bmatrix} S_{11} & S_{12} \\ 0 & S_{22} \end{bmatrix} \end{array}, \tag{18}$$

$$A(:, J_C) = Q_1[\,S_{11} \quad S_{12}\,] + Q_2[\,0 \quad S_{22}\,] = m \begin{bmatrix} \overset{k}{Q_1 S_{11}} & \overset{n-k}{Q_1 S_{12} + Q_2 S_{22}} \end{bmatrix}.$$

From this formulation, we set

$$C \coloneqq A(:, J_C(1:k)) = Q_1 S_{11}.$$

$$Q_1 S_1 = [\,Q_1 S_{11} \quad Q_1 S_{12}\,] = Q_1 S_{11} [\,I_k \quad S_{11}^{-1} S_{12}\,] = C[\,I_k \quad T_l\,],$$

where T_l is the solution to the matrix equation $S_{11} T_l = S_{12}$ (which if solved for T_l a column at a time, is simply a set of linear systems). It follows that we can write:

$$A \approx CV^T, \quad \text{where} \quad V^T = [I_k \quad T_l]P^T. \tag{19}$$

The one-sided ID of (rank k) is the approximate factorization:

$$\underset{m\times n}{A} \approx \underset{m\times k}{A(:, J_c(1:k))} \underset{k\times n}{V^T}, \tag{20}$$

where we use a *partial column skeleton* $C = A(:, J_c(1:k))$ of a subset of the columns of A and V is a well-conditioned matrix. Clearly, C simply represents a subset of the columns of A chosen based on the pivoting strategy used in the QR factorization. The typical pivoting strategy is to choose the permutation matrix P (which simply dictates the re-arrangement of the columns of A) such that if S_{22} above is omitted, yielding:

$$AP \approx Q_1[S_{11} \quad S_{12}] + Q_2[0 \quad 0] = Q_1\tilde{S},$$

then the components of $\tilde{S}$ satisfy $|\tilde{s}_{11}| \geq |\tilde{s}_{22}| \geq \cdots \geq |\tilde{s}_{nm}|$. Several other pivoting strategies can be employed and each will yield a somewhat different re-arrangement of the columns of A.

Once the single-sided ID is defined, the two-sided ID can be constructed simply by obtaining a one-sided ID of A and that of A^T. A set of select columns of A^T obtained by this procedure, will be the same as the set of select rows of A. Thus, we can write the two-sided ID of (rank k) as:

$$\underset{m\times n}{A} \approx \underset{m\times k}{W} \underset{k\times k}{A(J_r(1:k), J_c(1:k))} \underset{k\times n}{V^T}, \tag{21}$$

The procedure for the construction of the interpolative decompositions can be accelerated by means of randomization, just like for the low-rank SVD. This is possible by virtue of the result below [13].

Lemma 1 *Let* $\tilde{\Omega} \in \mathbb{R}^{l\times m}$ *be a matrix with GIID entries. Then, for any* $a \in \mathbb{R}^m$, *we have that* $E\left[\frac{\left\|\tilde{\Omega}a\right\|^2}{\|a\|^2}\right] = l$ *and* $Var\left[\frac{\left\|\tilde{\Omega}a\right\|^2}{\|a\|^2}\right] = 2l$.

Suppose, A is $m \times n$ and we draw a $l \times m$ GIID matrix $\tilde{\Omega}$. Suppose, we then form the $l \times n$ matrix $Z = \tilde{\Omega}A$. Then, $E\left[\frac{\|Z(:,j)\|^2}{\|A(:,j)\|^2}\right] = l$. As the pivoting result depends heavily on the ratio of the individual column norms of A with respect to one another, the above result tells us that the ratio of column norms is roughly preserved in a matrix resulting from the multiplication of the original matrix by a Gaussian random matrix from the left. As the product matrix consists of fewer rows than the original matrix, the pivoted QR factorization is correspondingly cheaper to perform on the product matrix Z than on A, while the resulting permutation matrix (really the re-arrangement vector) will be similar for both cases.

The two-sided ID allows us to construct the popular Column/Row skeleton CUR (rank k) decomposition:

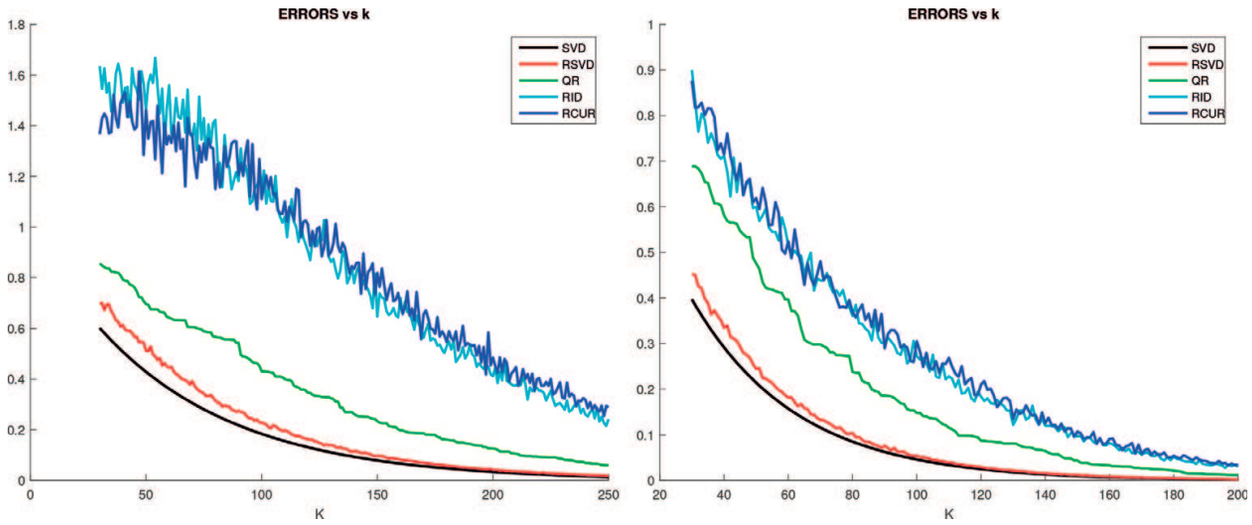


Figure 7. Relative errors for RSVD, ID, and CUR decompositions of rank k for matrices with two different rates of singular value decay (slower, faster).

$$\begin{array}{ccccc} A & \approx & C & U & R \\ m \times n & & m \times k & k \times k & k \times n \end{array} \tag{22}$$

Suppose, we compute a two-sided rank k ID factorization forming the $k \times k$ column/row skeleton $A(J_r(1{:}\,k), J_c(1{:}\,k))$. Set:

$$C = A(:, J_C(1:k)) \quad \text{and} \quad R = A(J_r(1:k), :)$$

We then set this to equal the factors C and R in CUR:

$$CUR = A(:, J_c(1:k))U\,A(J_r(1:k), :) \approx A(:, J_c(1:k))V^T \tag{23}$$

where we take U to satisfy the system $UR = V^T$: In **Figure 7**, we compare the relative errors obtained with different approximations at the same rank. For matrices with mild singular value decay, the low-rank SVD obtained via a randomized scheme (with oversampling) gives significantly closer to optimal performance (to true truncated SVD) than other decompositions.

8. An introduction to Backus-Gilbert inversion

As previously mentioned, damped least-squares (DLS) techniques are commonly exploited to solve linear, discrete inversion problems, such as those encountered in seismic tomography [15, 16]. To break the nonuniqueness of the least-squares solution, DLS inversion schemes often rely on *ad hoc* regularization strategies (e.g., model norm damping or smoothing), designed to subjectively favor the model simplicity. However, in regions of poor seismic data coverage, DLS methods may lead to locally *biased* model solutions—potentially causing model misinterpretations [17]. In other words, DLS models may represent "biased averages" over the true-

model parameters. Most geotomographical studies suffer from uneven data coverages, and thus are concerned by these averaging *bias* effects. For example, teleseismic body-wave raypaths irregularly sample the Earth's interior, because earthquakes typically are concentrated along oceanic ridges or subduction zones and seismometers are located over continental areas or oceanic islands.

A fundamentally different approach is that of linear Backus-Gilbert inversion [18–20], which belongs to the class of optimally localized averages (OLA) methods. In the discrete version of the Backus-Gilbert theory, one aims at evaluating (weighted) averages of the true-model parameters. That is, the Backus-Gilbert method seeks to determine unbiased model estimates. Over the past half century, many authors have considered that, in addition to being computationally very intensive, it could be a clumsy affair in the presence of data errors to practically implement the Backus-Gilbert method to (large-scale) tomographic applications [15, 21–23]. In the following, we aim to describe a recently developed—and (finally!) computationally tractable—*tomographic* approach [10] based on the Backus-Gilbert philosophy.

The SOLA (subtractive optimally localized averages) method [24, 25] is a variant of the original Backus-Gilbert approach, which has been exploited to solve helioseismic inversion problems [26, 27]. As a remark, Pijpers and Thompson [24] termed this alternative the SOLA method, though it may have been rediscovered independently by different authors [28, 29]. SOLA retains all the advantages of the original Backus-Gilbert method, but is much more computationally efficient and versatile in the construction of resolving (averaging) kernels. Recently, SOLA has been introduced and adapted to large-scale, linear and discrete "tomographic" problems by Zaroli [10]. We now briefly review the SOLA inversion scheme, tailored to seismic tomography.

In this section, let us slightly change the notations about linear inverse problems, to keep closer with those preferred in the geosciences community [10, 17]. Let us consider linear, discrete forward problems of the form:

$$\mathbf{d} = \mathbf{G}\mathbf{m} + \mathbf{n}, \tag{24}$$

where $\mathbf{d} = (d_i)_{1 \le i \le N}$ denotes the data, $\mathbf{G} = \left(G_{ij}\right)_{1 \le i,j \le N,M}$ the sensitivity matrix, $\mathbf{m} = \left(m_j\right)_{1 \le j \le M}$ the true-model parameters, and $\mathbf{n} = (n_i)_{1 \le i \le N}$ the noise. The sensitivity matrix elements are the partial derivatives of the data with respect to the model parameters: $G_{ij} = \partial d_i / \partial m_j$. Typically, in "large-scale" tomographic studies, one may have to deal with $M \gtrsim 10^5$ model parameters and $N \gtrsim 10^6$ data. Let us consider, without loss of generality, that the data are time-residuals, the model parameters are velocity anomalies, the model space is parametrized using regular-size cells (local and "orthonormal" parameterization), the noise is randomly drawn from a normal distribution $\mathcal{N}(0, \sigma_n)$, and the data covariance matrix is $\mathbf{C}_d = \sigma_n^2 \mathbf{I}_N$. For local and "irregular" parametrizations, the reader is referred to [10]. It is a common practice to normalize both the data and sensitivity matrix by the data errors; thus $\mathbf{C}_d = \mathbf{I}_N$.

One aims to find a model estimate, $\widehat{\mathbf{m}}$, that can be expressed as a linear combination of the data:

$$\widehat{\mathbf{m}} = \widehat{\mathbf{G}}^{\dagger}\mathbf{d},$$

where the matrix $\widehat{\mathbf{G}}^{\dagger}$ denotes *some* generalized inverse. The model estimate can be decomposed as

$$\underbrace{\widehat{\mathbf{m}}}_{\text{model estimate}} = \underbrace{\widehat{\mathbf{R}}\mathbf{m}}_{\text{filtered true model}} + \underbrace{\widehat{\mathbf{G}}^{\dagger}\mathbf{n}}_{\text{propagated noise}}, \tag{25}$$

where

$$\widehat{\mathbf{R}} = \widehat{\mathbf{G}}^{\dagger}\mathbf{G}, \tag{26}$$

is often referred to as the model resolution matrix. The first term in right member of (25), $\widehat{\mathbf{R}}\mathbf{m}$, represents the filtered true model, and shows our inability, if $\widehat{\mathbf{R}} \neq \mathbf{I}_M$, to perfectly recover the true model. Here, we refer to the k-th row of the resolution matrix, $\widehat{\mathbf{R}}_{k.} = \left(\widehat{R}_{kj}\right)_{1 \le j \le M}$, as the resolving kernel that linearly relates the k-th parameter estimate, $\widehat{m}_k$, to the true-model parameters:

$$\widehat{m}_k = \sum_{j=1}^{M} \widehat{R}_{kj} m_j, \quad \left(\text{ignoring the term of propagated noise}\right). \tag{27}$$

Therefore, we wish that $\widehat{\mathbf{R}}\mathbf{m}$ represents an *unbiased averaging* over the true model parameters, $\mathbf{m}$. This means that, for any parameter index $k \in [1, \ldots, M]$, we wish that $\widehat{\mathbf{R}}_k$. is non-negative and satisfies to

$$\sum_{j=1}^{M} \widehat{R}_{kj} = 1. \tag{28}$$

The second term in right member of (25), $\widehat{\mathbf{G}}^{\dagger}\mathbf{n}$, denotes the propagated noise (i.e. the propagation of data errors) into the model estimate. Robust model interpretations require accurate appraisals of model estimates, that is to compute and carefully analyze both $\widehat{\mathbf{R}}$ and the model covariance matrix

$$\mathbf{C}_{\widehat{m}} = \widehat{\mathbf{G}}^{\dagger}\mathbf{C}_d\left(\widehat{\mathbf{G}}^{\dagger}\right)^{T}. \tag{29}$$

As a remark, for DLS models this would also mean to quantify averaging bias effects (if any)—see [17]. The model estimate $\widehat{\mathbf{m}}$, resolution $\widehat{\mathbf{R}}$, and covariance $\mathbf{C}_{\widehat{m}}$ can be inferred from the generalized inverse $\widehat{\mathbf{G}}^{\dagger}$; efficiently computing the full generalized inverse is then crucial for any linear inverse problem. As we shall see, in the "SOLA Backus-Gilbert" approach the generalized inverse is directly determined.

The original Backus-Gilbert scheme consists in constructing the most peak-shaped resolving kernel (peaked around each model parameter location), while moderating at most the propagated noise into the model estimate. The key idea in the SOLA method is to specify an a *priori* "target form" for each resolving (averaging) kernel. One needs to specify M target resolving-kernels (hereafter, target kernels) such that their spatial extent represents some *a priori* estimate of the spatial resolving-length (around each parameter location). As an example, for 2-D tomographic studies the simplest target form could be circular (isotropic resolving-length); each target kernel would be constant inside such a circle and zero outside. Rather than minimizing the spread of each resolving kernel, as in the original Backus-Gilbert formulation, in the SOLA approach one aims at minimizing the integrated squared difference between each resolving kernel and its associated target kernel. Each *row* of the SOLA generalized inverse is individually computed by solving a specific minimization problem—the full computation of $\widehat{\mathbf{G}}^{\dagger}$ is then extremely parallel. The k-th row, $\widehat{\mathbf{G}}^{\dagger}_{k.} = \left(\widehat{G}^{\dagger}_{ki}\right)_{1\le i\le N}$, is found such that:

$$\min_{\widehat{\mathbf{G}}^{\dagger}_{k.}} \quad \underbrace{\sum_{j=1}^{M}\left(\widehat{R}_{kj}-T^{(k)}_{j}\right)^{2}}_{\text{resolution misfit}} + \eta_k^2 \quad \underbrace{\sigma^2_{\widehat{m}_k}}_{\text{model variance}} \quad , \quad \text{s.t.} \quad \sum_{j=1}^{M}\widehat{R}_{kj}=1, \tag{30}$$

where η_k and $\mathbf{t}^{(k)} = \left(T^{(k)}_{j}\right)_{1\le j\le M}$ are the k-th tradeoff parameter (resolution misfit *versus* model variance) and target resolving-kernel vector, respectively; k is the index of considered model parameter. Because of the additional constraint in (30), the k-th parameter estimate, $\widehat{m}_k$, is expected to be *unbiased* (provided that its corresponding resolving kernel is (mostly) non-negative)—so for the model estimate $\widehat{\mathbf{m}}$. Though not strictly necessary, here all M target kernels are imposed to be unimodular:

$$\sum_{j=1}^{M} T^{(k)}_{j} = 1, \quad \forall k \in [1,\cdots,M]. \tag{31}$$

The system to be solved for the k-th row of the SOLA generalized inverse then writes as follows:

$$\left(\mathbf{G}\,\mathbf{G}^{T}+\eta_k^2\mathbf{I}_N\right)\widehat{\mathbf{G}}^{\dagger}_{k.} = \mathbf{G}\mathbf{t}^{(k)}, \quad \text{s.t.} \quad \sum_{j=1}^{M}\sum_{i=1}^{N}\widehat{G}^{\dagger}_{ki}G_{ij}=1. \tag{32}$$

As a remark, since only a single (k-th) parameter index is treated at a time in (32), it could be difficult to ensure that all M selected values for the tradeoff parameters ($\eta^{(k)}$) would lead to "globally coherent" model solutions. However, it seems [10, 17] that globally coherent tomographic images can be obtained when using: (1) target kernels whose size is tuned to the spatially irregular data coverage (for instance using seismic ray-paths density as a proxy for the spatial variations of the local resolving-length); and (2) constant-valued tradeoff parameters, that is:

$$\eta_k = \eta, \quad \forall k \in [1, \cdots, M]. \tag{33}$$

In practice, it seems that η may (roughly) be determined from analyzing a few curves of tradeoff between $\sum_j \left(\widehat{R}_{kj} - T_j^{(k)}\right)^2$ and $\sigma^2_{\widehat{m}_k}$, for some randomly chosen parameter index (k). Let us now define the following quantities [10, 17, 30]:

$$\begin{cases} \mathrm{x}^{(\mathrm{k})} = \left(x_i^{(k)}\right)_{1 \le i \le N}, & x_i^{(k)} = \widehat{G}^{\dagger}_{ki} \\ \widehat{\mathrm{x}}^{(\mathrm{k})} = \left(x_i^{(k)}\right)_{2 \le i \le N} & c_i = \sum_{j=1}^{M} G_{ij} \\ \mathrm{c} = (c_i)_{1 \le i \le N}, \\ \widehat{\mathrm{c}} = (c_i/c_1)_{2 \le i \le N} \\ \mathrm{e}_1 = (\delta_{i1})_{1 \le i \le N} \\ \mathbf{B} = \begin{pmatrix} -\widehat{\mathrm{c}}^{\mathrm{T}} \\ \mathbf{I}_{N-1} \end{pmatrix} \\ \mathbf{Q}^{(\eta)} = \begin{pmatrix} \mathbf{G}^{\mathrm{T}}\mathbf{B} \\ -\eta\widehat{\mathrm{c}}^{\mathrm{T}} \end{pmatrix} \\ \mathbf{y}^{(k,\eta)} = \begin{pmatrix} \mathrm{t}^{(k)} - c_1^{-1}\mathbf{G}^T\mathrm{e}_1 \\ -c_1^{-1}\eta \end{pmatrix}, \end{cases} \tag{34}$$

where c_1 is assumed to be nonzero and δ denotes the Kronecker symbol. Solving (32) therefore consists in solving for $\widehat{\mathbf{x}}^{(k)}$ the following normal equations:

$$\begin{pmatrix} \mathbf{Q}^{(\eta)} \\ \eta\mathbf{I}_{N-1} \end{pmatrix} \widehat{\mathrm{X}}^{(k)} = \begin{pmatrix} \mathbf{y}^{(k,\eta)} \\ \mathbf{0}_{N-1} \end{pmatrix}, \tag{35}$$

using for instance the LSQR algorithm [31], and then to infer the final solution $\mathrm{x}^{(\mathrm{k})}$ (i.e., the k-th row of the SOLA generalized inverse) from $\widehat{\mathbf{x}}^{(k)}$ such that:

$$\mathrm{x}^{(k)} = \mathbf{B}\widehat{\mathrm{x}}^{(k)} + c_1^{-1}\mathrm{e}_1. \tag{36}$$

Last, but not least, we now aim to discuss about the computational efficiency of the SOLA approach for computing the full generalized inverse (see [10]). First, the rows of the generalized inverse matrix can be computed in *parallel* on P processors, so that computing all M rows would take $t \times M/P$ CPU-time, where t is the average CPU-time to numerically solve (35). A crucial point is that the matrix $\mathbf{Q}^{(\eta)}$, of size $(M+1) \times (N-1)$, does *not* depend on the parameter index (k), so that it does not need to be recomputed M times—as it was required in the original Backus-Gilbert approach (see [10]). The vector $\mathbf{y}^{(k,\eta)}$ has to be recomputed M times, but that task is computationally cheap. $\mathbf{Q}^{(\eta)}$ and $\mathbf{y}^{(k,\eta)}$ can easily be reconstructed if one aims at investigating different η values (only the last row of $\mathbf{Q}^{(\eta)}$ and last element of $\mathbf{y}^{(k,\eta)}$ depend on

η). Finally, simply re-ordering the rows of the sensitivity matrix **G** (and corresponding data), such that the *first* row of **G** is the sparsest one, allows the matrix $\mathbf{Q}^{(\eta)}$ to be almost as *sparse* as G —this sparsity property is very useful when solving (35), in terms of storage, efficiency of the LSQR algorithm, and memory footprint.

Figure 8 shows an example of the SOLA method applied to global-scale seismic tomography [10], for which there are M = 38, 125 model parameters and N = 79, 765 data (teleseismic shear-wave time-residuals). Tomographic images represent isotropic, 3–D shear-wave velocity perturbations within the whole Earth's mantle (with respect to some reference, radial absolute velocity model). **Figure 8a** and **b** displays the tomographic model $\widehat{\mathbf{m}}$, at about 600 km depth, and its uncertainty $\sigma\widehat{m}$ computed as

$$\sigma_{\widehat{m}} = \left(\sigma_{\widehat{m}_k}\right)_{1\le k\le M'} \quad \sigma_{\widehat{m}_k} = \sqrt{\sum_{i=1}^{N}\left(\widehat{G}^{\dagger}_{ki}\right)^2}, \tag{37}$$

(since the data are normalized by their errors), respectively. The form of each target kernel is that of a 3-D spheroid, corresponding to *a priori* lateral and radial resolving lengths that may

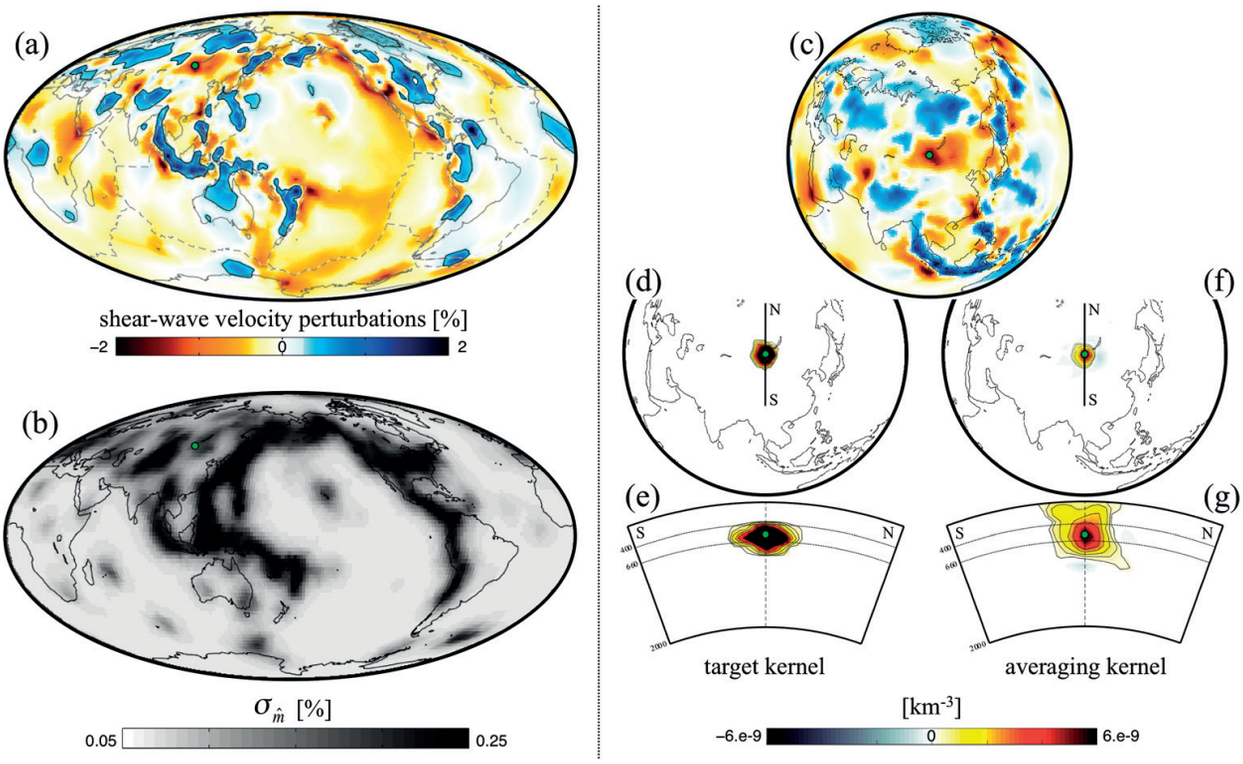


Figure 8. Example of a global geotomographical model and its associated resolution and uncertainty, obtained from using a "SOLA Backus-Gilbert" inversion approach [10]. (a) Model estimate, $\widehat{\mathbf{m}}$, shown at 600 km depth; (c) model uncertainty, $\sigma_{\widehat{m}}$, shown at 600 km depth; (b) zoom-in on $\widehat{\mathbf{m}}$ (600 km depth) around the k'-th parameter location, i.e., the green dot; (d, e) and (f, g) horizontal (600 km depth) and vertical cross-sections through the k'-th target (spheroid shape) and averaging kernels, respectively.

be expected locally, at best, given the data coverage. Let us focus on the k'-th model parameter, marked by a green dot in **Figure 8**; a zoom-in on the tomographic model is shown in **Figure 8c**. Horizontal (600 km depth) and vertical cross-sections through the k'-th target kernel are displayed in **Figure 8d** and **e**, respectively. The corresponding k'-th resolving (averaging) kernel is similarly displayed in **Figure 8f** and **g**.

Finally, the "SOLA Backus-Gilbert" approach, introduced and adapted to large-scale, linear, discrete tomographic problems by Zaroli [10], allows to efficiently compute unbiased models, including their full resolution and covariance—enabling quantitative model interpretations [17].

9. Conclusion

In this work, we have presented several techniques useful to the practitioner in the field of inverse problems, with the aim to give an idea of when and how these techniques should be employed for various linear and nonlinear applications. We have discussed techniques such as sparse matrix storage, the use of pivoted factorizations for direct solves, ℓ_2, ℓ_1 and intermediate penalty-based regularization strategies, nonlinear least squares problems, the construction and use of low-rank factorizations, and an application of the Backus-Gilbert inversion approach tailored to seismic tomography.

Author details

Sergey Voronin[1]* and Christophe Zaroli[2]

*Address all correspondence to: svoronin@i-a-i.com

1 Intelligent Automation Inc, Rockville, MD, USA

2 Institut de Physique du Globe de Strasbourg, Université de Strasbourg, EOST/CNRS, France

References

[1] Golub GH, Van Loan CF. Matrix Computations. Vol. 3. JHU Press; 2012

[2] Bell T, McKenzie B. Compression of sparse matrices by arithmetic coding. In: Data Compression Conference, 1998 (DCC'98) Proceedings; IEEE; 1998. pp. 23-32

[3] Candès EJ, Wakin MB. An introduction to compressive sampling. IEEE Signal Processing Magazine. 2008;**25**(2):21-30

[4] Voronin S, Zaroli C, Cuntoor NP. Conjugate gradient based acceleration for inverse problems. GEM-International Journal on Geomathematics. 2017;**8**(2):219-239

[5] Beck A, Teboulle M. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM Journal on Imaging Sciences. 2009;**2**(1):183-202

[6] Voronin S, Woerdeman HJ. A new iterative firm-thresholding algorithm for inverse problems with sparsity constraints. Applied and Computational Harmonic Analysis. 2013;**35**(1):151-164

[7] Voronin S, Chartrand R. A new generalized thresholding algorithm for inverse problems with sparsity constraints. In: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2013; IEEE; 2013. pp. 1636-1640

[8] Voronin S, Daubechies I. An iteratively reweighted least squares algorithm for sparse regularization. arXiv preprint: arXiv:1511.08970. 2015

[9] Wang Y, Yang J, Yin W, Zhang Y. A new alternating minimization algorithm for total variation image reconstruction. SIAM Journal on Imaging Sciences. 2008;**1**(3):248-272

[10] Zaroli C. Global seismic tomography using Backus-Gilbert inversion. Geophysical Journal International. 2016;**207**(2):876-888

[11] Hansen PC. The L-Curve and its Use in the Numerical Treatment of Inverse Problems. IMM, Department of Mathematical Modelling, Technical University of Denmark; 1999

[12] Eckart C, Young G. A principal axis transformation for non-Hermitian matrices. Bulletin of the American Mathematical Society. 1939;**45**(2):118-121

[13] Voronin S, Martinsson P-G. Rsvdpack: Subroutines for computing partial singular value decompositions via randomized sampling on single core, multi core, and gpu architectures. arXiv preprint: arXiv:1502.05366, 2:16. 2015

[14] Martinsson P-G, Voronin S. A randomized blocked algorithm for efficiently computing rank-revealing factorizations of matrices. SIAM Journal on Scientific Computing. 2016; **38**(5):S485-S507

[15] Aster RC, Borchers B, Thurber C. Parameter Estimation and Inverse Problems. Revised ed. Elsevier; 2012

[16] Nolet G. A Breviary of Seismic Tomography. Cambridge, UK: Cambridge University Press; 2008

[17] Zaroli C, Koelemeijer P, Lambotte S. Toward seeing the earth's interior through unbiased tomographic lenses. Geophysical Research Letters. 2017;**44**(22):11399-11408. DOI: 10.10 02/2017GL074996

[18] Backus G, Gilbert JF. Numerical applications of a formalism for geophysical inverse problems. Geophysical Journal of the Royal Astronomical Society. 1967;**13**:247-276

[19] Backus G, Gilbert JF. The resolving power of gross earth data. Geophysical Journal of the Royal Astronomical Society. 1968;**16**:169-205

[20] Backus G, Gilbert JF. Uniqueness in the inversion of inaccurate gross earth data. Philosophical Transactions of the Royal Society A. 1970;**266**(1173)

[21] Menke W. Geophysical Data Analysis: Discrete Inverse Theory. Revised ed. San Diego: Academic Press; 1989

[22] Parker RL. Geophysical Inverse Theory. Princeton: Princeton University Press; 1994

[23] Trampert J. Global seismic tomography: The inverse problem and beyond. Inverse Problems. 1998;**14**:371-385

[24] Pijpers FP, Thompson MJ. Faster formulations of the optimally localized averages method for helioseismic inversions. Astronomy and Astrophysics. 1992;**262**:L33-L36

[25] Pijpers FP, Thompson MJ. The sola method for helioseismic inversion. Astronomy and Astrophysics. 1994;**281**:231-240

[26] Jackiewicz J, Birch AC, Gizon L, Hanasoge SM, Hohage T, Ruffio J-B, Svanda M. Multichannel three-dimensional SOLA inversion for local helioseismology. Solar Physics. 2012;**276**:19-33

[27] Rabello-Soares MC, Basu S, Christensen-Dalsgaard J. On the choice of parameters in solar-structure inversion. Monthly Notices of the Royal Astronomical Society. 1999;**309**: 35-47

[28] Larsen RM, Hansen PC. Efficient implementations of the sola mollifier method. Astronomy and Astrophysics Supplement Series. 1997;**121**:587-598

[29] Louis AK, Maass P. A mollifier method for linear operator equations of the first kind. Inverse Problems. 1990;**6**:427-490

[30] Nolet G. Solving or resolving inadequate and noisy tomographic systems. Journal of Computational Physics. 1985;**61**:463-482

[31] Paige CC, Saunders MA. Lsqr: An algorithm for sparse, linear equations and sparse least squares. ACM Transactions on Mathematical Software. 1982;**8**:43-71

IntechOpen