Regularization Tools

A Matlab Package for
Analysis and Solution of Discrete Ill-Posed Problems

Version 3.0 for Matlab 5.2

Per Christian Hansen

Department of Mathematical Modelling
Building 305, Technical University of Denmark
DK-2800 Lyngby, Denmark

`pch@imm.dtu.dk`
`http://www.imm.dtu.dk/~pch`

June 1992
Last revision March 1998

The software described in this report is published in
Numerical Algorithms **6** (1994), pp. 1–35, and is available via
Netlib (`netlib@research.att.com`) in the file `numeralgo/na4`.

Contents

Changes Since Version 2.0

The following is a list of the major changes since Version 2.0 of the package.

- Replaced gsvd by cgsvd which has a *different* sequence of output arguments.
- Removed the obsolete function csdecomp (which replaced the function csd)
- Deleted the function mgs.
- Changed the storage format of bidiagonal matrices to sparse, instead of a dense matrix with two columns.
- Removed the obsolete function bsvd.
- Added the function regutm that generates random test matrices for regularization methods.
- Added the blur test problem.
- Functions tsvd and tgsvd now allow k = 0, and functions tgsvd and tikhonov now allow a square L.
- Added output arguments rho and eta to functions dsvd, mtsvd, tgsvd, tikhonov, and tsvd.
- Added a priori guess x_0 as input to tikhonov.
- Corrected get_l such that the sign of L*x is correct.
- Added MGS reorthogonalization of the normal equation residual vectors in the two functions cgls and pcgls.
- Added the method 'ttls' to the function fil_fac.
- More precise computation of the regularization parameter in gcv, lcurve, and quasiopt.
- Changed heb_new and newton to work with λ instead of λ^2.
- Added legend to lagrange and picard.

1. Introduction

Ill-posed problems—and regularization methods for computing stabilized solutions to the ill-posed problems—occur frequently enough in science and engineering to make it worthwhile to present a general framework for their numerical treatment. The purpose of this package of Matlab routines is to provide the user with easy-to-use routines, based on numerically robust and efficient algorithms, for doing experiments with analysis and solution of discrete ill-posed problems by means of regularization methods.

The theory for ill-posed problems is well developed in the literature. We can easily illustrate the main difficulties associated with such problems by means of a small numerical example. Consider the following least squares problem

$$\min_{\mathbf{x}} \|A\,\mathbf{x} - \mathbf{b}\|_2$$

with coefficient matrix A and right-hand side $\mathbf{b}$ given by

$$A = \begin{pmatrix} 0.16 & 0.10 \\ 0.17 & 0.11 \\ 2.02 & 1.29 \end{pmatrix}, \qquad \mathbf{b} = \begin{pmatrix} 0.27 \\ 0.25 \\ 3.33 \end{pmatrix} .$$

Here, the right-hand side $\mathbf{b}$ is generated by adding a small perturbation to an exact right-hand side corresponding to the exact solution $\bar{\mathbf{x}}^T = (1\ 1)$:

$$\mathbf{b} = \begin{pmatrix} 0.16 & 0.10 \\ 0.17 & 0.11 \\ 2.02 & 1.29 \end{pmatrix} \begin{pmatrix} 1.00 \\ 1.00 \end{pmatrix} + \begin{pmatrix} 0.01 \\ -0.03 \\ 0.02 \end{pmatrix} .$$

The difficulty with this least squares problem is that the matrix A is ill-conditioned; its condition number is $1.1 \cdot 10^3$. This implies that the computed solution is potentially very sensitive to perturbations of the data. Indeed, if we compute the ordinary least-squares solution $\mathbf{x}_{\mathrm{LSQ}}$ by means of a QR factorization of A, then we obtain

$$\mathbf{x}_{\mathrm{LSQ}} = \begin{pmatrix} 7.01 \\ -8.40 \end{pmatrix} .$$

This solution is obviously worthless, and something must be done in order to compute a better approximation to the exact solution $\bar{\mathbf{x}}^T = (1\ 1)$.

The large condition number implies that the columns of A are nearly linearly dependent. One could therefore think of replacing the ill-conditioned matrix $A = (\mathbf{a}_1\ \mathbf{a}_2)$ with either $(\mathbf{a}_1\ \mathbf{0})$ or $(\mathbf{0}\ \mathbf{a}_2)$, both of which are well conditioned. The two corresponding so-called basic solutions are

$$\mathbf{x}_{\mathrm{B}}^{(1)} = \begin{pmatrix} 1.65 \\ 0.00 \end{pmatrix} , \qquad \mathbf{x}_{\mathrm{B}}^{(2)} = \begin{pmatrix} 0.00 \\ 2.58 \end{pmatrix} .$$

Although these solutions are much less sensitive to perturbations of the data, and although the corresponding residual norms are both small,

$$\|A\,\mathbf{x}_{\rm B}^{(1)} - \mathbf{b}\|_2 = 0.031 \ , \qquad \|A\,\mathbf{x}_{\rm B}^{(2)} - \mathbf{b}\|_2 = 0.036 \ ,$$

the basic solutions nevertheless have nothing in common with $\bar{\mathbf{x}}^T = (1\ 1)$.

A major difficulty with the ordinary least squares solution $\mathbf{x}_{\rm LSQ}$ is that its norm is significantly greater than the norm of the exact solution. One may therefore try another approach to solving the least squares problem by adding the side constraint that the solution norm must not exceed a certain value α,

$$\min_{\mathbf{x}} \|A\,\mathbf{x} - \mathbf{b}\|_2 \quad \text{subject to} \quad \|\mathbf{x}\|_2 \leq \alpha \ .$$

The such computed solution $\mathbf{x}_\alpha$ depends in a non-linear way on α, and for α equal to 0.1, 1, 1.385, and 10 we obtain

$$\mathbf{x}_{0.1} = \begin{pmatrix} 0.08 \\ 0.05 \end{pmatrix} , \qquad \mathbf{x}_1 = \begin{pmatrix} 0.84 \\ 0.54 \end{pmatrix} , \qquad \mathbf{x}_{1.385} = \begin{pmatrix} 1.17 \\ 0.74 \end{pmatrix} , \qquad \mathbf{x}_{10} = \begin{pmatrix} 6.51 \\ -7.60 \end{pmatrix} .$$

We see that by a proper choice of α we can indeed compute a solution $\mathbf{x}_{1.385}$ which is fairly close to the desired exact solution $\bar{\mathbf{x}}^T = (1\ 1)$. However, care must be taken when choosing α, and the proper choice of α is not obvious.

Although the above example is a small one, it highlights the three main difficulties associates with discrete ill-posed problems:

1. the condition number of the matrix A is large
2. replacing A by a well-conditioned matrix derived from A does not necessarily lead to a useful solution
3. care must taken when imposing additional constraints.

The purpose of numerical regularization theory is to provide efficient and numerically stable methods for including proper side constraints that lead to useful stabilized solutions, and to provide robust methods for choosing the optimal weight given to these side constraints such that the regularized solution is a good approximation to the desired unknown solution.

The routines provided in this package are examples of such procedures. In addition, we provide a number of utility routines for analyzing the discrete ill-posed problems in details, for displaying these properties, and for easy generation of simple test problems. By means of the routines in Regularization Tools, the user can—at least for small to medium-size problems—experiment with different regularization strategies, compare them, and draw conclusions from these experiments that would otherwise require a major programming effort. For discrete ill-posed problems, which are indeed difficult to treat numerically, such an approach is certainly superior to a single black-box routine.

The package was mainly developed in the period 1990–1992 at UNI•C and to some extent it reflects the author's own work. Prof. Dianne P. O'Leary and Dr. Martin Hanke helped with the iterative methods. Prof. Lars Eldén—and his 1979 Simula package [22] with the same purpose as Regularization Tools—provided a great source of inspiration. The package was also inspired by a paper by Natterer [58] where a "numerical analyst's toolkit for ill-posed problems" is suggested. The Fortran programs by Drake [19], te Riele [66], and Wahba and her co-workers [6] also deserve to be mentioned here.

2. Discrete Ill-Posed Problems and their Regularization

In this chapter we give a brief introduction to discrete ill-posed problems, we discuss some numerical regularization methods, and we introduce several numerical "tools" such as the singular value decomposition, the discrete Picard condition, and the L-curve, which are suited for analysis of the discrete ill-posed problems. A more complete treatment of all these aspects is given in [47].

2.1. Discrete Ill-Posed Problems

The concept of ill-posed problems goes back to Hadamard in the beginning of this century, cf. e.g. [34]. Hadamard essentially defined a problem to be *ill-posed* if the solution is not unique or if it is not a continuous function of the data—i.e., if an arbitrarily small perturbation of the data can cause an arbitrarily large perturbation of the solution. Hadamard believed that ill-posed problems were "artificial" in that they would not describe physical systems. He was wrong, though, and today there is a vast amount of literature on ill-posed problems arising in many areas of science and engineering, cf. e.g. [14, 15, 16, 32, 57, 61, 63, 69, 79].

The classical example of an ill-posed problem is a Fredholm integral equation of the first kind with a square integrable kernel [31],

$$\int_a^b K(s,t)\, f(t)\, dt = g(s) \;, \qquad c \le s \le d \;, \tag{2.1}$$

where the right-hand side g and the kernel K are given, and where f is the unknown solution. If the solution f is perturbed by

$$\Delta f(t) = \epsilon \sin(2\pi p\, t) \;, \qquad p = 1, 2, \ldots \;, \qquad \epsilon = \text{constant}$$

then the corresponding perturbation of the right-hand side g is given by

$$\Delta g(s) = \epsilon \int_a^b K(s,t) \sin(2\pi p\, t)\, dt \;, \qquad p = 1, 2, \ldots$$

and due to the Riemann-Lebesgue lemma it follows that $\Delta g \to 0$ as $p \to \infty$ [31, p. 2]. Hence, the ratio $\|\Delta f\|/\|\Delta g\|$ can become arbitrary large by choosing the integer p large enough, thus showing that (2.1) is an ill-posed problem. In particular, this example illustrates that Fredholm integral equations of the first kind with square integrable kernels are extremely sensitive to high-frequency perturbations.

Strictly speaking, ill-posed problems must be infinite dimensional—otherwise the ratio $\|\Delta f\|/\|\Delta g\|$ stays bounded, although it may become very large. However, certain finite-dimensional discrete problems have properties very similar to those of ill-posed problems, such as being highly sensitive to high-frequency perturbations, and it is natural to associate the term *discrete ill-posed problems* with these problems. We can be more precise with this characterization for linear systems of equations

$$A\,\mathbf{x} = \mathbf{b}\ , \qquad A \in \mathbb{R}^{n\times n} \tag{2.2}$$

and linear least-squares problems

$$\min_{\mathbf{x}} \|A\,\mathbf{x} - \mathbf{b}\|_2\ , \qquad A \in \mathbb{R}^{m\times n}\ , \qquad m > n\ . \tag{2.3}$$

We say that these are discrete ill-posed problems if both of the following criteria are satisfied:

1. the singular values of A decay gradually to zero
2. the ratio between the largest and the smallest nonzero singular values is large.

Singular values are discussed in detail in Section 2.3. Criterion 2 implies that the matrix A is ill-conditioned, i.e., that the solution is potentially very sensitive to perturbations; criterion 1 implies that there is no "nearby" problem with a well-conditioned coefficient matrix and with well-determined numerical rank.

The typical manifestations of discrete ill-posed problems are systems of linear equations and linear least-squares problems arising from discretization of ill-posed problems. E.g., if a Galerkin-type method [3] is used to discretize the Fredholm integral equation (2.1), then a problem of the form (2.2) or (2.3) arises—depending on the type of collocation method used—with the elements a_{ij} and b_i of the matrix A and the right-hand side $\mathbf{b}$ given by

$$a_{ij} = \int_a^b \int_c^d K(s,t)\,\phi_i(s)\,\psi_j(t)\,ds\,dt\ , \qquad b_i = \int_c^d \phi_i(s)\,g(s)\,ds\ , \tag{2.4}$$

where ϕ_i and ψ_j are the particular basis functions used in the Galerkin method. For such problems, the close relationship between the ill-posedness of the integral equation and the ill-conditioning of the matrix A are well understood [1, 39, 78]. In particular, it can be shown that the singular values of A decay in such a way that both criteria 1 and 2 above are satisfied.

An interesting and important aspect of discrete ill-posed problems is that the ill-conditioning of the problem does not mean that a meaningful approximate solution cannot be computed. Rather, the ill-conditioning implies that standard methods in numerical linear algebra [9, 29] for solving (2.2) and (2.3), such as LU, Cholesky, or QR factorization, cannot be used in a straightforward manner to compute such a solution. Instead, more sophisticated methods must be applied in order to ensure the computation of a meaningful solution. This is the essential goal of regularization methods.

The package Regularization Tools provides a collection of easy-to-use Matlab routines for the numerical treatment of discrete ill-posed problems. The philosophy behind Regularization Tools is modularity and regularity between the routines. Many routines require the SVD of the coefficient matrix A—this is not necessarily the best approach in a given application, but it is certainly well suited for Matlab [54] and for this package.

The numerical treatment of integral equations in general is treated in standard references such as [4, 5, 13, 17, 18], and surveys of regularization theory can be found in, e.g., [7, 10, 31, 32, 46, 47, 50, 51, 65, 73, 79].

2.2. Regularization Methods

The primary difficulty with the discrete ill-posed problems (2.2) and (2.3) is that they are essentially underdetermined due to the cluster of small singular values of A. Hence, it is necessary to incorporate further information about the desired solution in order to stabilize the problem and to single out a useful and stable solution. This is the purpose of *regularization*.

Although many types of additional information about the solution $\mathbf{x}$ is possible in principle, the dominating approach to regularization of discrete ill-posed problems is to require that the 2-norm—or an appropriate seminorm—of the solution be small. An initial estimate $\mathbf{x}^*$ of the solution may also be included in the side constraint. Hence, the side constraint involves minimization of the quantity

$$\Omega(\mathbf{x}) = \|L\,(\mathbf{x} - \mathbf{x}^*)\|_2 \ . \tag{2.5}$$

Here, the matrix L is typically either the identity matrix I_n or a $p \times n$ discrete approximation of the $(n-p)$-th derivative operator, in which case L is a banded matrix with full row rank. In some cases it is more appropriate that the side constraint be a Sobolev norm of the form

$$\Omega(\mathbf{x})^2 = \alpha_0^2 \|\mathbf{x} - \mathbf{x}^*\|_2^2 + \sum_{i=1}^{q} \alpha_i^2 \|L_i\,(\mathbf{x} - \mathbf{x}^*)\|_2^2 \ ,$$

where L_i approximates the ith derivative operator. Notice that this Ω can always be written in the form (2.5) by setting L equal to the Cholesky factor of the matrix $\alpha_0^2 I_n + \sum_{i=1}^{q} \alpha_i^2 L_i^T L_i$. By means of the side constraint Ω one can therefore control the smoothness of the regularized solution.

When the side constraint $\Omega(\mathbf{x})$ is introduced, one must give up the requirement that $A\,\mathbf{x}$ equals $\mathbf{b}$ in the linear system (2.2) and instead seek a solution that provides a fair balance between minimizing $\Omega(\mathbf{x})$ and minimizing the residual norm $\|A\,\mathbf{x} - \mathbf{b}\|_2$. The underlying idea is that a regularized solution with small (semi)norm and a suitably small residual norm is not too far from the desired, unknown solution to the unperturbed problem underlying the given problem. The same idea of course also applies to the least squares problem (2.3).

Undoubtedly, the most common and well-known form of regularization is the one known as *Tikhonov regularization* [62, 67, 68]. Here, the idea is to define the regularized solution $\mathbf{x}_\lambda$ as the minimizer of the following weighted combination of the residual norm and the side constraint

$$\mathbf{x}_\lambda = \operatorname{argmin} \left\{ \|A\,\mathbf{x} - \mathbf{b}\|_2^2 + \lambda^2 \|L\,(\mathbf{x} - \mathbf{x}^*)\|_2^2 \right\} \ , \tag{2.6}$$

where the *regularization parameter* λ controls the weight given to minimization of the side constraint relative to minimization of the residual norm. Clearly, a large λ (equivalent to a large amount of regularization) favors a small solution seminorm at the cost of a large residual norm, while a small λ (i.e., a small amount of regularization) has the opposite effect. As we shall see in Eq. (2.14), λ also controls the sensitivity of the regularized solution $\mathbf{x}_\lambda$ to perturbations in A and $\mathbf{b}$, and the perturbation bound is proportional to λ^{-1}. Thus, the regularization parameter λ is an important quantity which controls the properties of the regularized solution, and λ should therefore be chosen with care. In Section 2.9 we return to numerical methods for actually computing λ.

We remark that an underlying assumption for the use of Tikhonov regularization in the form of Eq. (2.6) is that the errors in the right-hand side are unbiased and that their covariance matrix is proportional to the identity matrix. If the latter condition is not satisfied one should incorporate the additional information and rescale the problem or use a regularized version of the general Gauss-Markov linear model:

$$\min \left\{ \|\mathbf{u}\|_2^2 + \lambda^2 \|L\,\mathbf{x}\|_2^2 \right\} \qquad \text{subject to} \qquad A\,\mathbf{x} + C\,\mathbf{u} = \mathbf{b}\ , \tag{2.7}$$

where C is the Cholesky factor of the covariance matrix. The latter approach using (2.7) must be used if the covariance matrix is rank deficient, i.e., if C is not a square matrix. For a discussion of this approach and a numerical algorithm for solving (2.7), cf. [80].

Besides Tikhonov regularization, there are many other regularization methods with properties that make them better suited to certain problems or certain computers. We return to these methods in Sections 2.7 and 2.8, but first it is convenient to introduce some important numerical "tools" for analysis of discrete ill-posed problems in Sections 2.3–2.5. As we shall demonstrate, getting insight into the discrete ill-posed problem is often at least as important as computing a solution, because the regularized solution should be computed with such care. Finally, in Section 2.9 we shall describe some methods for choosing the regularization parameter.

2.3. SVD and Generalized SVD

The superior numerical "tools" for analysis of discrete ill-posed problems are the *singular value decomposition* (SVD) of A and its generalization to two matrices, the *generalized singular value decomposition* (GSVD) of the matrix pair (A, L) [29, §2.5.3 and §8.7.3]. The SVD reveals all the difficulties associated with the ill-conditioning of the matrix A while the GSVD of (A, L) yields important insight into the regularization problem involving both the coefficient matrix A and the regularization matrix L, such as in (2.6).

2.3.1. The Singular Value Decomposition

Let $A \in \mathbb{R}^{m\times n}$ be a rectangular matrix with $m \geq n$. Then the SVD of A is a decomposition of the form

$$A = U\,\Sigma\,V^T = \sum_{i=1}^{n} \mathbf{u}_i\,\sigma_i\,\mathbf{v}_i^T\ , \tag{2.8}$$

where $U = (\mathbf{u}_1, \ldots, \mathbf{u}_n)$ and $V = (\mathbf{v}_1, \ldots, \mathbf{v}_n)$ are matrices with orthonormal columns, $U^T U = V^T V = I_n$, and where $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_n)$ has non-negative diagonal elements appearing in non-increasing order such that

$$\sigma_1 \geq \ldots \geq \sigma_n \geq 0\ . \tag{2.9}$$

The numbers σ_i are the *singular values* of A while the vectors $\mathbf{u}_i$ and $\mathbf{v}_i$ are the left and right singular vectors of A, respectively. The condition number of A is equal to the ratio σ_1/σ_n.

From the relations $A^T A = V\,\Sigma^2\,V^T$ and $A\,A^T = U\,\Sigma^2 U^T$ we see that the SVD of A is strongly linked to the eigenvalue decompositions of the symmetric positive semi-definite matrices $A^T A$ and $A\,A^T$. This shows that the SVD is unique for a given matrix A—except for singular vectors associated with multiple singular values. In connection with discrete ill-posed problems, two characteristic features of the SVD of A are very often found.

- The singular values σ_i decay gradually to zero with no particular gap in the spectrum. An increase of the dimensions of A will increase the number of small singular values.
- The left and right singular vectors $\mathbf{u}_i$ and $\mathbf{v}_i$ tend to have more sign changes in their elements as the index i increases, i.e., as σ_i decreases.

Although these features are found in many discrete ill-posed problems arising in practical applications, they are unfortunately very difficult—or perhaps impossible—to prove in general.

To see how the SVD gives insight into the ill-conditioning of A, consider the following relations which follow directly from Eq. (2.8):

$$\left.\begin{array}{rcl} A\,\mathbf{v}_i &=& \sigma_i\,\mathbf{u}_i \\ \|A\,\mathbf{v}_i\|_2 &=& \sigma_i \end{array}\right\} \qquad i = 1,\dots,n\ . \tag{2.10}$$

We see that a small singular value σ_i, compared to $\|A\|_2 = \sigma_1$, means that there exists a certain linear combination of the columns of A, characterized by the elements of the right singular vector $\mathbf{v}_i$, such that $\|A\,\mathbf{v}_i\|_2 = \sigma_i$ is small. In other words, one or more small σ_i implies that A is nearly rank deficient, and the vectors $\mathbf{v}_i$ associated with the small σ_i are numerical null-vectors of A. From this and the characteristic features of A we conclude that the matrix in a discrete ill-posed problem is always highly ill-conditioned, and its numerical null-space is spanned by vectors with many sign changes.

The SVD also gives important insight into another aspect of discrete ill-posed problems, namely the smoothing effect typically associated with a square integrable kernel. Notice that as σ_i decreases, the singular vectors $\mathbf{u}_i$ and $\mathbf{v}_i$ become more and more oscillatory. Consider now the mapping $A\,\mathbf{x}$ of an arbitrary vector $\mathbf{x}$. Using the SVD, we get $\mathbf{x} = \sum_{i=1}^n (\mathbf{v}_i^T\mathbf{x})\,\mathbf{v}_i$ and

$$A\,\mathbf{x} = \sum_{i=1}^n \sigma_i\,(\mathbf{v}_i^T\mathbf{x})\,\mathbf{u}_i\ .$$

This clearly shows that the due to the multiplication with the σ_i the high-frequency components of $\mathbf{x}$ are more damped in $A\,\mathbf{x}$ than then low-frequency components. Moreover, the inverse problem, namely that of computing $\mathbf{x}$ from $A\,\mathbf{x} = \mathbf{b}$ or $\min \|A\,\mathbf{x} - \mathbf{b}\|_2$, must have the opposite effect: it amplifies the high-frequency oscillations in the right-hand side $\mathbf{b}$.

2.3.2. The Generalized Singular Value Decomposition

The GSVD of the matrix pair (A, L) is a generalization of the SVD of A in the sense that the generalized singular values of (A, L) are the square roots of the generalized eigenvalues of the matrix pair $(A^T A, L^T L)$. In order to keep our exposition simple, we assume that the dimensions of $A \in \mathbb{R}^{m\times n}$ and $L \in \mathbb{R}^{p\times n}$ satisfy $m \geq n \geq p$, which is always the case in connection with discrete ill-posed problems. Then the GSVD is a decomposition of A and L in the form

$$A = U \begin{pmatrix} \Sigma & 0 \\ 0 & I_{n-p} \end{pmatrix} X^{-1} , \qquad L = V\,(M\ ,\ 0)\, X^{-1} , \tag{2.11}$$

where the columns of $U \in \mathbb{R}^{m\times n}$ and $V \in \mathbb{R}^{p\times p}$ are orthonormal, $X \in \mathbb{R}^{n\times n}$ is nonsingular, and Σ and M are $p\times p$ diagonal matrices: $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_p)$, $M = \mathrm{diag}(\mu_1, \ldots, \mu_p)$. Moreover, the diagonal entries of Σ and M are non-negative and ordered such that

$$0 \leq \sigma_1 \ \leq \ldots \leq \sigma_p \leq 1 \ , \qquad 1 \geq \mu_1 \ \geq \ldots \geq \mu_p > 0 \ ,$$

and they are normalized such that

$$\sigma_i^2 + \mu_i^2 = 1 \ , \qquad i = 1, \ldots, p \ .$$

Then the *generalized singular values* γ_i of (A, L) are defined as the ratios

$$\gamma_i = \sigma_i/\mu_i \ , \qquad i = 1, \ldots, p \ , \tag{2.12}$$

and they obviously appear in non-decreasing order. For historical reasons, this ordering is the opposite of the ordering of the ordinary singular values of A.

For $p < n$ the matrix $L \in \mathbb{R}^{p\times n}$ always has a nontrivial null-space $\mathcal{N}(L)$. E.g., if L is an approximation to the second derivative operator on a regular mesh, $L = \mathrm{tridiag}(1, -2, 1)$, then $\mathcal{N}(L)$ is spanned by the two vectors $(1, 1, \ldots, 1)^T$ and $(1, 2, \ldots, n)^T$. In the GSVD, the last $n - p$ columns $\mathbf{x}_i$ of the nonsingular matrix X satisfy

$$L\,\mathbf{x}_i = \mathbf{0} \ , \qquad i = p+1, \ldots, n \tag{2.13}$$

and they are therefore basis vectors for the null-space $\mathcal{N}(L)$.

There is a slight notational problem here because the matrices U, Σ, and V in the GSVD of (A, L) are *different* from the matrices with the same symbols in the SVD of A. However, in this presentation it will always be clear from the context which decomposition is used. When L is the identity matrix I_n, then the U and V of the GSVD are identical to the U and V of the SVD, and the generalized singular values of (A, I_n) are identical to the singular values of A—except for the ordering of the singular values and vectors.

In general, there is no connection between the generalized singular values/vectors and the ordinary singular values/vectors. For discrete ill-posed problems, though, we can actually say something about the SVD-GSVD connection because L is typically a reasonably well-conditioned matrix. When this is the case, then it can be shown that the matrix X in (2.11) is also well-conditioned. Hence, the diagonal matrix Σ must display the ill-conditioning of A, and since $\gamma_i = \sigma_i\,(1 - \sigma_i^2)^{1/2} \approx \sigma_i$ for small σ_i the generalized singular values must decay gradually to zero as the ordinary singular values do. Moreover, the oscillation properties (i.e., the increase in sign changes) of the right singular vectors carries over to the columns of X in the GSVD: the smaller the γ_i the more sign changes in $\mathbf{x}_i$. For more specific results, cf. [41].

As an immediate example of the use of GSVD in the analysis of discrete regularization problems, we mention the following perturbation bound for Tikhonov regularization derived in [40]. Let E and $\mathbf{e}$ denote the perturbations of A and $\mathbf{b}$, respectively, and let $\bar{\mathbf{x}}_\lambda$ denote the exact solution to the unperturbed problem; then the relative error in the perturbed solution $\mathbf{x}_\lambda$ satisfies

$$\frac{\|\mathbf{x}_\lambda - \bar{\mathbf{x}}_\lambda\|_2}{\|\bar{\mathbf{x}}_\lambda\|_2} \leq \frac{\|A\|_2 \, \|X\|_2 \, \lambda^{-1}}{1 - \|E\|_2 \, \|X\|_2 \, \lambda^{-1}} \times \left((1 + \mathrm{cond}(X)) \frac{\|E\|_2}{\|A\|_2} + \frac{\|\mathbf{e}\|_2}{\|\mathbf{b}_\lambda\|_2} + \|E\|_2 \, \|X\|_2 \, \lambda^{-1} \frac{\|\mathbf{r}_\lambda\|_2}{\|\mathbf{b}_\lambda\|_2} \right) \tag{2.14}$$

where we have defined $\mathbf{b}_\lambda = A\,\mathbf{x}_\lambda$ and $\mathbf{r}_\lambda = \mathbf{b} - \mathbf{b}_\lambda$. The important conclusion we can make from this relation is that for all reasonable λ the perturbation bound for the regularized solution $\mathbf{x}_\lambda$ is proportional to λ^{-1} and to the norm of the matrix X. The latter quantity is analyzed in [41] where it is shown that $\|X\|_2$ is approximately bounded by $\|L^\dagger\|_2$, i.e., by the inverse of the smallest singular value of L. Hence, in addition to controlling the smoothness of the regularized solution, λ and L also control its sensitivity to perturbations of A and $\mathbf{b}$.

The SVD and the GSVD are computed by means of routines **csvd** and **cgsvd** in this package.

2.4. The Discrete Picard Condition and Filter Factors

As we have seen in Section 2.3, the integration in Eq. (2.1) with a square integrable kernel K (2.1) has a smoothing effect on f. The opposite operation, namely, that of solving the first kind Fredholm integral equation for f, therefore tends to amplify oscillations in the right-hand side g. Hence, if we require that the solution f be a square integrable solution with finite L_2-norm, then not all functions are valid as right-hand side g. Indeed, g must be sufficiently smooth to "survive" the inversion back to f. The mathematical formulation of this smoothness criterion on g—once the kernel K is given—is called the Picard condition [31, §1.2].

For discrete ill-posed problems there is, strictly speaking, no Picard condition because the norm of the solution is always bounded. Nevertheless, it makes sense to introduce a discrete Picard condition as follows. In a real-world application, the right-hand side $\mathbf{b}$ is always contaminated by various types or errors, such as measurement errors, approximation errors, and rounding errors. Hence, $\mathbf{b}$ can be written as

$$\mathbf{b} = \bar{\mathbf{b}} + \mathbf{e} \ , \tag{2.15}$$

where $\mathbf{e}$ are the errors, and $\bar{\mathbf{b}}$ is the unperturbed right-hand side. Both $\bar{\mathbf{b}}$ and the corresponding unperturbed solution $\bar{\mathbf{x}}$ represent the underlying unperturbed and unknown problem. Now, if we want to be able to compute a regularized solution $\mathbf{x}_{\mathrm{reg}}$ from the given right-hand side $\mathbf{b}$ such that $\mathbf{x}_{\mathrm{reg}}$ approximates the exact solution $\bar{\mathbf{x}}$, then it is shown in [44] that the corresponding exact right-hand side $\bar{\mathbf{b}}$ must satisfy a criterion very similar to the Picard condition:

The discrete Picard condition. The unperturbed right-hand side $\bar{\mathbf{b}}$ in a discrete ill-posed problem with regularization matrix L satisfies the discrete Picard condition if the

Fourier coefficients $|\mathbf{u}_i^T \bar{\mathbf{b}}|$ on the average decay to zero faster than the generalized singular values γ_i.

The discrete Picard condition is not as "artificial" as it first may seem: it can be shown that if the underlying integral equation (2.1) satisfies the Picard condition, then the discrete ill-posed problem obtained by discretization of the integral equation satisfies the discrete Picard condition [39]. See also [73, 74].

The main difficulty with discrete ill-posed problems is caused by the errors $\mathbf{e}$ in the given right-hand side $\mathbf{b}$ (2.15), because such errors typically tend to have components along all the left singular vectors $\mathbf{u}_i$. For example, if $\|\mathbf{e}\|_2 = \epsilon$ and if the elements of $\mathbf{e}$ are unbiased and uncorrelated, then the expected value of the Fourier coefficients of $\mathbf{e}$ satisfy

$$\mathcal{E}\left(|\mathbf{u}_i^T \mathbf{e}|\right) = m^{-\frac{1}{2}}\epsilon \; , \qquad i = 1, \ldots, n \; . \tag{2.16}$$

As a consequence, the Fourier coefficients $|\mathbf{u}_i^T \mathbf{b}|$ of the perturbed right-hand side level off at approximately $m^{-1/2}\epsilon$ even if the unperturbed right-hand side $\bar{\mathbf{b}}$ satisfies the discrete Picard condition, because these Fourier coefficients are dominated by $|\mathbf{u}_i^T \mathbf{e}|$ for large i.

Consider now the linear system (2.2) and the least squares problem (2.3), and assume for simplicity that A has no exact zero singular values. Using the SVD, it is easy to show that the solutions to both systems are given by the same equation:

$$\mathbf{x}_{\mathrm{LSQ}} = \sum_{i=1}^{n} \frac{\mathbf{u}_i^T \mathbf{b}}{\sigma_i} \mathbf{v}_i \; . \tag{2.17}$$

This relation clearly illustrates the difficulties with the standard solution to (2.2) and (2.3). Since the Fourier coefficients $|\mathbf{u}_i^T \mathbf{b}|$ corresponding to the smaller singular values σ_i do not decay as fast as the singular values—but rather tend to level off—the solution $\mathbf{x}_{\mathrm{LSQ}}$ is dominated by the terms in the sum corresponding to the smallest σ_i. As a consequence, the solution $\mathbf{x}_{\mathrm{LSQ}}$ has many sign changes and thus appears completely random.

With this analysis in mind, we can see that the purpose of a regularization method is to dampen or filter out the contributions to the solution corresponding to the small generalized singular values. Hence, we will require that a regularization method produces a regularized solution $\mathbf{x}_{\mathrm{reg}}$ which, for $\mathbf{x}^* = \mathbf{0}$, can be written as follows

$$\mathbf{x}_{\mathrm{reg}} = \sum_{i=1}^{n} f_i \, \frac{\mathbf{u}_i^T \mathbf{b}}{\sigma_i} \mathbf{v}_i \qquad \text{if} \quad L = I_n \tag{2.18}$$

$$\mathbf{x}_{\mathrm{reg}} = \sum_{i=1}^{p} f_i \, \frac{\mathbf{u}_i^T \mathbf{b}}{\sigma_i} \mathbf{x}_i + \sum_{i=p+1}^{n} (\mathbf{u}_i^T \mathbf{b}) \, \mathbf{x}_i \quad \text{if} \quad L \neq I_n \; . \tag{2.19}$$

Here, the numbers f_i are *filter factors* for the particular regularization method. The filter factors must have the important property that as σ_i decreases, the corresponding f_i tends to zero in such a way that the contributions $(\mathbf{u}_i^T \mathbf{b}/\sigma_i)\,\mathbf{x}_i$ to the solution from the smaller σ_i are effectively filtered out. The difference between the various regularization methods lies essentially in the way that these filter factors f_i are defined. Hence, the filter factors play an important role in connection with regularization theory, and it is worthwhile to characterize the filter factors for the various regularization methods that we shall present below.

For Tikhonov regularization, which plays a central role in regularization theory, the filter factors are either $f_i = \sigma_i^2/(\sigma_i^2+\lambda^2)$ (for $L = I_n$) or $f_i = \gamma_i^2/(\gamma_i^2+\lambda^2)$ (for $L \neq I_n$), and the filtering effectively sets in for $\sigma_i < \lambda$ and $\gamma_i < \lambda$, respectively. In particular, this shows that discrete ill-posed problems are essentially unregularized by Tikhonov's method for $\lambda > \sigma_1$ and $\lambda > \gamma_p$, respectively.

Filter factors for various regularization methods can be computed by means of routine fil_fac in this package, while routine picard plots the important quantities σ_i, $|\mathbf{u}_i^T\mathbf{b}|$, and $|\mathbf{u}_i^T\mathbf{b}|/\sigma_i$ if $L = I_n$, or γ_i, $|\mathbf{u}_i^T\mathbf{b}|$, and $|\mathbf{u}_i^T\mathbf{b}|/\gamma_i$ if $L \neq I_n$.

2.5. The L-Curve

Perhaps the most convenient graphical tool for analysis of discrete ill-posed problems is the so-called *L-curve* which is a plot—for all valid regularization parameters—of the (semi)norm $\|L\,\mathbf{x}_{\mathrm{reg}}\|_2$ of the regularized solution versus the corresponding residual norm $\|A\,\mathbf{x}_{\mathrm{reg}} - \mathbf{b}\|_2$. In this way, the L-curve clearly displays the compromise between minimization of these two quantities, which is the heart of any regularization method. The use of such plots in connection with ill-conditioned least squares problems goes back to Miller [55] and Lawson & Hanson [52].

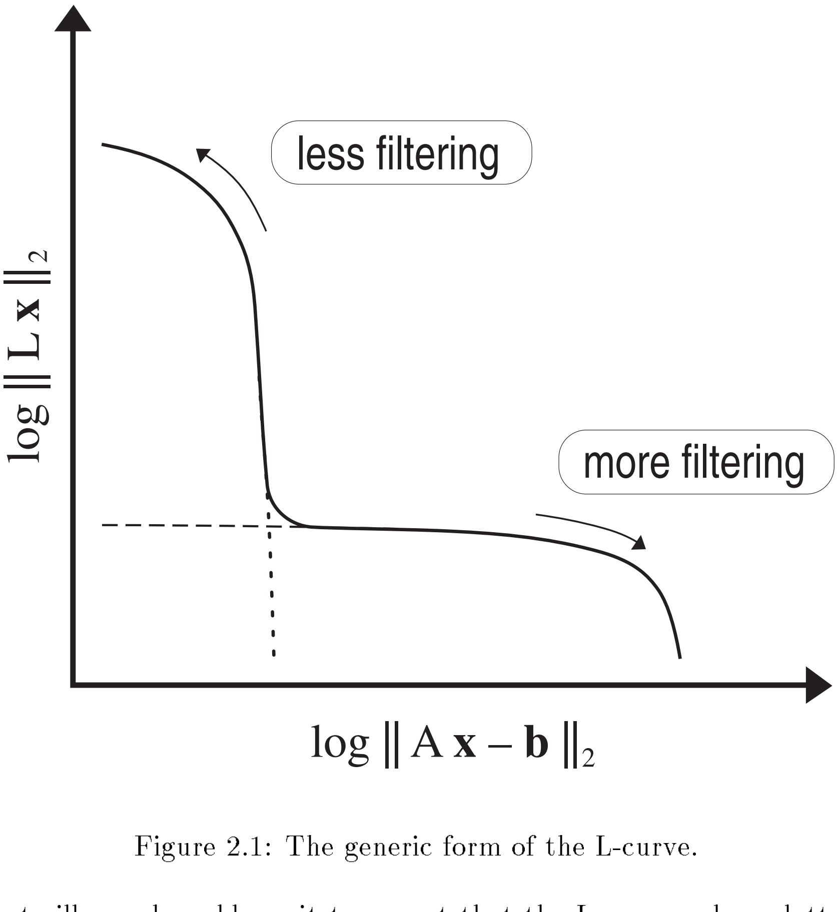


Figure 2.1: The generic form of the L-curve.

For discrete ill-posed problems it turns out that the L-curve, when plotted in *log-log scale*, almost always has a characteristic L-shaped appearance (hence its name) with a distinct corner separating the vertical and the horizontal parts of the curve. To see why this is so, we notice that if $\bar{\mathbf{x}}$ denotes the exact, unregularized solution corresponding to the exact right-hand side $\bar{\mathbf{b}}$ in Eq. (2.15), then the error $\mathbf{x}_{\rm reg} - \bar{\mathbf{x}}$ in the regularized solution consists of two components, namely, a perturbation error from the error $\mathbf{e}$ in the given right-hand side $\mathbf{b}$, and a regularization error due to the regularization of the error-free component $\bar{\mathbf{b}}$ in the right-hand side. The vertical part of the L-curve corresponds to solutions where $\|L\,\mathbf{x}_{\rm reg}\|_2$ is very sensitive to changes in the regularization parameter because the perturbation error $\mathbf{e}$ from dominates $\mathbf{x}_{\rm reg}$ and because $\mathbf{e}$ does not satisfy the discrete Picard condition. The horizontal part of the L-curve corresponds to solutions where it is the residual norm $\|A\,\mathbf{x}_{\rm reg} - \mathbf{b}\|_2$ that is most sensitive to the regularization parameter because $\mathbf{x}_{\rm reg}$ is dominated by the regularization error—as long as $\bar{\mathbf{b}}$ satisfies the discrete Picard condition.

We can substantiate this by means of the relations for the regularized solution $\mathbf{x}_{\rm reg}$ in terms of the filter factors. For general-form regularization ($L \neq I_n$) Eq. (2.19) yields the following expression for the error in $\mathbf{x}_{\rm reg}$:

$$\mathbf{x}_{\rm reg} - \bar{\mathbf{x}} = \left(\sum_{i=1}^{p} f_i \, \frac{\mathbf{u}_i^T \mathbf{e}}{\sigma_i} \, \mathbf{x}_i + \sum_{i=p+1}^{n} (\mathbf{u}_i^T \mathbf{e})\, \mathbf{x}_i \right) + \sum_{i=1}^{p} (f_i - 1)\, \frac{\mathbf{u}_i^T \bar{\mathbf{b}}}{\sigma_i} \, \mathbf{x}_i \ . \tag{2.20}$$

Here, the term in the parenthesis is the *perturbation error* due to the perturbation $\mathbf{e}$, and the second term is the *regularization error* caused by regularization of the unperturbed component $\bar{\mathbf{b}}$ of the right-hand side. When only little regularization is introduced, most of the filter factors f_i are approximately one and the error $\mathbf{x}_{\rm reg} - \bar{\mathbf{x}}$ is dominated by the perturbation error. On the other hand, with plenty of regularization most filter factors are small, $f_i \ll 1$, and $\mathbf{x}_{\rm reg} - \bar{\mathbf{x}}$ is dominated by the regularization error.

In [45, 48] Eq. (2.20) was used to analyze the relationship between the error in $\mathbf{x}_{\rm reg}$ and the behavior of the L-curve. The result is that if $\bar{\mathbf{b}}$ satisfies the discrete Picard condition, then the horizontal part of the curve corresponds to solutions where the regularization error dominates—i.e., where so much filtering is introduced that the solution stays very smooth and $\|L\,\mathbf{x}_{\rm reg}\|_2$ therefore only changes a little with the regularization parameter. In contrast, the vertical part of the L-curve corresponds to solutions that are dominated by the perturbation error, and due to the division by the small σ_i it is clear that $\|L\,\mathbf{x}_{\rm reg}\|_2$ varies dramatically with the regularization parameter while, simultaneously, the residual norm does not change much. Moreover, it is shown in [48] that a log-log scale emphasizes the different appearances of the vertical and the horizontal parts. In this way, the L-curve clearly displays the trade-off between minimizing the residual norm and the side constraint.

We note in passing that the L-curve is a continuous curve when the regularization parameter is continuous as in Tikhonov regularization. For regularization methods with a discrete regularization parameter, such as truncated SVD, we plot the L-curve as a finite set of points. How to make such a "discrete L-curve" continuous is discussed in [48]. In this reference, alternative norms for plotting the L-curve, depending on the regularization method, are also discussed.

The L-curve for Tikhonov regularization plays a central role in connection with regularization methods for discrete ill-posed problems because it divides the first quadrant into two regions. It is impossible to construct any solution that corresponds to a point below the Tikhonov L-curve; any regularized solution must lie on or above this curve. The solution computed by Tikhonov regularization is therefore optimal in the sense that for a given residual norm there does not exist a solution with smaller seminorm than the Tikhonov solution—and the same is true with the roles of the norms interchanged. A consequence of this is that one can compare other regularization methods with Tikhonov regularization by inspecting how close the L-curve for the alternative method is to the Tikhonov L-curve. If $\bar{\mathbf{b}}$ satisfies the discrete Picard condition, then the two L-curves are close to each other and the solutions computed by the two regularization methods are similar [45].

For a given fixed right-hand side $\mathbf{b} = \bar{\mathbf{b}} + \mathbf{e}$, there is obviously an optimal regularization parameter that balances the perturbation error and the regularization error in $\mathbf{x}_{\mathrm{reg}}$. An essential feature of the L-curve is that this optimal regularization parameter—defined in the above sense—is not far from the regularization parameter that corresponds to the L-curve's corner [45]. In other words, by locating the corner of the L-curve one can compute an approximation to the optimal regularization parameter and thus, in turn, compute a regularized solution with a good balance between the two error types. We return to this aspect in Section 2.9; suffice it here to say that for continuous L-curves, a computationally convenient definition of the L-curve's corner is the point with maximum curvature in log-log scale.

In the Regularization Tools package, routine l_curve produces a log-log plot of the L-curve and—if required—also locates the corner and identifies the corresponding regularization parameter. Given a discrete set of values of $\|A\,\mathbf{x}_{\mathrm{reg}} - \mathbf{b}\|_2$ and $\|L\,\mathbf{x}_{\mathrm{reg}}\|_2$, routine plot_lc plots the corresponding L-curve, while routine l_corner locates the L-curve's corner.

2.6. Transformation to Standard Form

A regularization problem with side constraint $\Omega(\mathbf{x}) = \|L\,(\mathbf{x} - \mathbf{x}^*)\|_2$ (2.5) is said to be in *standard form* if the matrix L is the identity matrix I_n. In many applications, regularization in standard form is not the best choice, i.e., one should use some $L \neq I_n$ in the side constraint $\Omega(\mathbf{x})$. The proper choice of matrix L depends on the particular application, but often an approximation to the first or second derivative operator gives good results.

However, from a numerical point of view it is much simpler to treat problems in standard form, basically because only one matrix, A, is involved instead of the two matrices A and L. Hence, it is convenient to be able to transform a given regularization problem in general form into an equivalent one in standard form by means of numerically stable methods. For example, for Tikhonov regularization we want a numerically stable method for transforming the general-form problem (2.6) into the following standard-form problem

$$\min \left\{ \|\bar{A}\,\bar{\mathbf{x}} - \bar{\mathbf{b}}\|_2^2 + \lambda^2\,\|\bar{\mathbf{x}} - \bar{\mathbf{x}}^*\|_2^2 \right\} , \tag{2.21}$$

where the new matrix $\bar{A}$, the new right-hand side $\bar{\mathbf{b}}$, and the vector $\bar{\mathbf{x}}^*$ are derived from the original quantities A, L, $\mathbf{b}$, and $\mathbf{x}^*$. Moreover, we also want a numerically stable scheme for transforming the solution $\bar{\mathbf{x}}_\lambda$ to (2.21) back to the general-form setting. Finally, we prefer a transformation that leads to a simple relationship between the SVD of $\bar{A}$ and the GSVD of (A, L), for then we have a perfect understanding of the relationship between the two regularization problems.

For the simple case where L is square and invertible, the transformation is obvious: $\bar{A} = A\,L^{-1}$, $\bar{\mathbf{b}} = \mathbf{b}$, $\bar{\mathbf{x}}^* = L\,\mathbf{x}^*$, and the back-transformation simply becomes $\mathbf{x}_\lambda = L^{-1}\bar{\mathbf{x}}_\lambda$.

In most applications, however, the matrix L is not square, and the transformation becomes somewhat more involved than just a matrix inversion. It turns out that it is a good idea to distinguish between direct and iterative regularization methods—cf. the next two sections. For the direct methods we need to be able to compute the matrix $\bar{A}$ explicitly by standard methods such as the QR factorization. For the iterative methods, on the other hand, we merely need to be able to compute the matrix-vector product $\bar{A}\,\bar{\mathbf{x}}$ efficiently. Below, we describe two methods for transformation to standard form which are suited for direct and iterative methods, respectively. We assume that the matrix $L \in \mathbb{R}^{p\times n}$ has full row rank, i.e., the rank of L is p.

2.6.1. Transformation for Direct Methods

The standard-form transformation for direct methods described here was developed by Eldén [21], and it is based on two QR factorizations. The description of this transformation is quite algorithmic, and it is summarized below (where, for convenience, the subscripts p, o, and q denote matrices with p, $n-p$, and $m-(n-p)$ columns, respectively). First, compute a QR factorization of L^T,

$$L^T = K\,R = (K_p\;K_o)\begin{pmatrix} R_p \\ 0 \end{pmatrix} . \tag{2.22}$$

We remark that since L has full rank, its pseudoinverse is simply $L^\dagger = K_p\,R_p^{-T}$. Moreover, the columns of K_o are an orthonormal basis for the null space of L. Next, form the "skinny" matrix $A\,K_o \in \mathbb{R}^{m\times(n-p)}$ and compute its QR factorization,

$$A\,K_o = H\,T = (H_o\;H_q)\begin{pmatrix} T_o \\ 0 \end{pmatrix} . \tag{2.23}$$

Then the transformed quantities $\bar{A}$ and $\bar{\mathbf{b}}$ are given by the following identities

$$\bar{A} = H_q^T A\,L^\dagger = H_q^T A\,K_p\,R_p^{-T} , \qquad \bar{\mathbf{b}} = H_q^T\,\mathbf{b} , \tag{2.24}$$

and we stress that the most efficient way to compute $\bar{A}$ and $\bar{\mathbf{b}}$ is to apply the orthogonal transformations that make up K and H "on the fly" to A and $\mathbf{b}$ as the QR factorizations in (2.22) and (2.23) are computed. When (2.21) has been solved for $\bar{\mathbf{x}}$, the transformation back to the general-form setting then takes the form

$$\mathbf{x} = L^\dagger\bar{\mathbf{x}} + K_o\,T_o^{-1}H_o^T(\mathbf{b} - A\,L^\dagger\bar{\mathbf{x}}) . \tag{2.25}$$

The SVD of the matrix $\bar{A}$ is related to the GSVD of (A, L) as follows: let $\bar{A} = \bar{U}\,\bar{\Sigma}\,\bar{V}^T$ denote the SVD of $\bar{A}$, and let E_p denote the $p\times p$ exchange matrix $E_p = \mathrm{antidiag}(1,\ldots,1)$. Also, let U, V, Σ, M, and X denote the GSVD matrices from Eq. (2.11). Then

$$U = (H_q\,\bar{U}\,E_p\;,\;H_o)\;, \qquad \Sigma\,M^{-1} = E_p\,\bar{\Sigma}\,E_p \tag{2.26}$$

$$V = \bar{V}\,E_p\;, \qquad X = \begin{pmatrix} M^{-1}V^T L \\ H_o^T A \end{pmatrix}^{-1} . \tag{2.27}$$

Moreover, the last $n-p$ columns of X are given by $K_o\,T_o^{-1}$. For proofs of (2.26)–(2.27) and an investigation of the accuracy of the GSVD matrices computed this way, cf. [41, 43].

2.6.2. Transformation for Iterative Methods

For the iterative methods the matrix $\bar{A}$ is never computed explicitly. Instead, one merely needs to be able to pre-multiply a vector with $\bar{A}$ and $\bar{A}^T$ efficiently. If K_o is an orthonormal basis for the null space of L, e.g. computed by (2.22), then $\mathbf{y} = L^\dagger \bar{\mathbf{x}}$ and $\hat{\mathbf{y}} = (L^\dagger)^T \mathbf{x}$, both of which are used in the iterative procedures, can easily by computed by the following algorithms:

$$\begin{aligned} \mathbf{y} &\leftarrow \begin{pmatrix} I_{n-p} \; 0 \\ L \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{0} \\ \bar{\mathbf{x}} \end{pmatrix} \\ \mathbf{y} &\leftarrow \mathbf{y} - K_o K_o^T \mathbf{y} \end{aligned} \qquad \begin{aligned} \mathbf{x} &\leftarrow \mathbf{x} - K_o K_o^T \mathbf{x} \\ \begin{pmatrix} \hat{\mathbf{y}} \\ \mathbf{z} \end{pmatrix} &\leftarrow \begin{pmatrix} L \\ 0 \; I_{n-p} \end{pmatrix}^{-T} \mathbf{x} \; . \end{aligned} \tag{2.28}$$

By means of these simple algorithms, which are described in [8], the above standard-transformation method can also be used for iterative methods. Notice that a basis for the null space of L is required—often, the basis vectors can be computed explicitly, or they can be computed from L by, say, a rank revealing factorization [12].

The algorithm from §2.6.1 can be reformulated in such a way that the pseudoinverse $L^\dagger$ is replaced by a weaker generalized inverse, using an idea from [23] (and later advocated in [35, 36, 37]). This reformulation has certain advantages for iterative methods, as we shall see in §2.8.4. Define the *A-weighted generalized inverse of* L as follows

$$L_A^\dagger = X \begin{pmatrix} M^{-1} \\ 0 \end{pmatrix} V^T \; , \tag{2.29}$$

where we emphasize that $L_A^\dagger$ is generally different from the pseudoinverse $L^\dagger$ when $p < n$. Also, define the vector

$$\mathbf{x}_0 = \sum_{i=p+1}^{n} (\mathbf{u}_i^T \mathbf{b}) \, \mathbf{x}_i \; , \tag{2.30}$$

which is the unregularized component of the regularized solution $\mathbf{x}_{\mathrm{reg}}$, cf. Eq. (2.19), i.e., $\mathbf{x}_0$ is the component of $\mathbf{x}_{\mathrm{reg}}$ that lies in the null space of L. Then the standard-form quantities $\bar{A}$ and $\bar{\mathbf{b}}$ in the alternative version of the algorithm are defined as follows

$$\bar{A} = A \, L_A^\dagger \; , \qquad \bar{\mathbf{b}} = \mathbf{b} - A \, \mathbf{x}_0 \; . \tag{2.31}$$

Moreover, the transformation back to the general-form setting takes the simple form

$$\mathbf{x} = L_A^\dagger \bar{\mathbf{x}} + \mathbf{x}_0 \; . \tag{2.32}$$

This backtransformation is mathematically equivalent to the one in (2.25) since we have

$$L_A^\dagger = (I_n - K_o T_o^{-1} H_o^T A) \, L^\dagger \qquad \text{and} \qquad \mathbf{x}_0 = K_o T_o^{-1} H_o^T \mathbf{b} \; . \tag{2.33}$$

To use this alternative formulation of the standard-form transformation, we need to compute $\mathbf{x}_0$ as well as the matrix-vector products $L_A^\dagger \bar{\mathbf{x}}$ and $(L_A^\dagger)^T \mathbf{x}$ efficiently. Given a basis W for $\mathcal{N}(L)$, the vector $\mathbf{x}_0$ is computed by the following algorithm:

$$\begin{aligned} S &\leftarrow (A \, W)^\dagger \\ \mathbf{x}_0 &\leftarrow W \, S \, \mathbf{b} \; . \end{aligned} \tag{2.34}$$

This algorithm involves $O(mn(n-p))$ operations. To compute $L_A^\dagger \bar{\mathbf{x}}$ and $(L_A^\dagger)^T \mathbf{x}$ efficiently (we emphasize that $L_A^\dagger$ is never computed explicitly), we partition $L = (L_{11}\,,\, L_{12})$, $T = (T_{11}\,,\, T_{12})$ and $\mathbf{x}^T = (\mathbf{x}_1^T\,,\, \mathbf{x}_2^T)$ such that L_{11} and T_{11} have p columns and $\mathbf{x}_1$ has p elements. For efficiency, we also need to compute the $(n-p) \times n$ matrix

$$T \leftarrow S\,A\;. \tag{2.35}$$

Then $\mathbf{y} = L_A^\dagger \bar{\mathbf{x}}$ and $\hat{\mathbf{y}} = (L_A^\dagger)^T \mathbf{x}$ are computed by:

$$\begin{array}{lcl} \mathbf{y} & \leftarrow & L_{11}^{-1}\bar{\mathbf{x}} \\ \mathbf{y} & \leftarrow & \begin{pmatrix} \mathbf{y} \\ 0 \end{pmatrix} - W\,T_{11}\,\mathbf{y} \end{array} \qquad\qquad \begin{array}{lcl} \mathbf{x} & \leftarrow & \mathbf{x} - T_{11}^T W^T \mathbf{x} \\ \hat{\mathbf{y}} & \leftarrow & L_{11}^{-T}\mathbf{x}\;. \end{array} \tag{2.36}$$

In the above formulas, W need not have orthonormal columns, although this is the best choice from a numerical point of view. For more details about these algorithms, cf. [47, Section 2.3.2].

For the latter standard-form transformation there is an even simpler relation between the SVD of $\bar{A}$ and part of the GSVD of (A, L) than in §2.6.1 because $A\,L_A^\dagger = \sum_{i=i}^p \mathbf{u}_i\,\gamma_i\,\mathbf{v}_i^T$. I.e., except for the ordering the GSVD quantities $\mathbf{u}_i$, γ_i, and $\mathbf{v}_i$ are identical to the similar SVD quantities, and with the same notation as in Eq. (2.26) we have

$$(\mathbf{u}_1 \ldots \mathbf{u}_p) = \bar{U}\,E_p\;, \qquad V = \bar{V}\,E_p\;, \qquad \Sigma\,M^{-1} = E_p\,\bar{\Sigma}\,E_p\;. \tag{2.37}$$

This relation is very important in connection with the iterative regularization methods.

2.6.3. Norm Relations etc.

From the above relations (2.26) and (2.37) between the SVD of $\bar{A}$ and the GSVD of (A, L) we obtain the following very important relations between the norms related to the two regularization problems

$$\|L\,(\mathbf{x} - \mathbf{x}^*)\|_2 = \|\bar{\mathbf{x}} - \bar{\mathbf{x}}^*\|_2\;, \qquad \|A\,\mathbf{x} - \mathbf{b}\|_2 = \|\bar{A}\,\bar{\mathbf{x}} - \bar{\mathbf{b}}\|_2\;, \tag{2.38}$$

where $\mathbf{x}$ denotes the solution obtained by transforming $\bar{\mathbf{x}}$ back to the general-form setting. These relations are very important in connection with methods for choosing the regularization parameter because they show that any parameter-choice strategy based on these norms will yield the *same* regularization parameter when applied to the general-form problem and the transformed standard-form problem.

Several routines are included in Regularization Tools for computations related to the standard-form transformations. First of all, the routines **std_form** and **gen_form** perform both transformations to standard from and back to general form. These routines are mainly included for pedagogical reasons. Routine **pinit** computes the vector $\mathbf{x}_0$ in Eq. (2.30) as well as the matrix $T\,A$ by means of Algorithm (2.34). Finally, the two routines **lsolve** and **ltsolve** compute $L_A^\dagger \bar{\mathbf{x}}$ and $(L_A^\dagger)^T \mathbf{x}$ by the algorithms in (2.36).

Regarding the matrix L, discrete approximations to derivative operators on a regular mesh can be computed by routine **get_l** which also provides a matrix W with orthonormal basis vectors for the null space of L.

2.7. Direct Regularization Methods

In this and the next section we shall briefly review the regularization methods for numerical treatment of discrete ill-posed problems included in the REGULARIZATION TOOLS package. Our aim is not to compare these and other methods, because that is outside the scope of this paper. In fact, very little has been done in this area, cf. [2, 37, 42, 53]. This section focuses on methods which are essentially direct, i.e., methods where the solution is defined by a direct computation (which may still involve an iterative root-finding procedure, say), while regularization methods which are intrinsically iterative are treated in the next section.

2.7.1. Tikhonov Regularization

Tikhonov's method is of course a direct method because the regularized solution $\mathbf{x}_\lambda$, defined by Eq. (2.6), is the solution to the following least squares problem

$$\min \left\| \begin{pmatrix} A \\ \lambda L \end{pmatrix} \mathbf{x} - \begin{pmatrix} \mathbf{b} \\ \lambda L\,\mathbf{x}^* \end{pmatrix} \right\|_2 , \tag{2.39}$$

and it is easy to see that $\mathbf{x}_\lambda$ is unique if the null spaces of A and L intersect trivially (as they usually do in practice). The most efficient algorithm for numerical treatment of Tikhonov's method for a general regularization matrix L consists of three steps [21]. First, the problem is transformed into standard form by means of Eqs. (2.22)–(2.24) from §2.6.1 ($\bar{A} = A$ if $L = I_n$), then the matrix $\bar{A}$ is transformed into a $p \times p$ upper bidiagonal matrix $\bar{B}$ by means of left and right orthogonal transformations,

$$\bar{A} = \bar{U}\,\bar{B}\,\bar{V}^T ,$$

and finally the resulting sparse problem with a banded $\bar{B}$ is solved for $\bar{V}^T\bar{\mathbf{x}}_\lambda$ and the solution is transformed back to the original setting by means of (2.25).

In this package we take another approach to solving (2.39), namely, by using the filter factors and the GSVD explicitly (or the SVD, if $L = I_n$), cf. Eqs. (2.18) and (2.19). This approach, which is implemented in routine `tikhonov`, is more suited to Matlab's coarse granularity. For pedagogical reasons, we also include a routine `bidiag` for bidiagonalization of a matrix.

2.7.2. Least Squares with a Quadratic Constraint

There are two other regularization methods which are almost equivalent to Tikhonov's method, and which can be treated numerically by essentially the same technique as mentioned above involving a transformation to standard form followed by bidiagonalization of the coefficient matrix. These two methods are the following least squares problems with a quadratic inequality constraint

$$\min \|A\,\mathbf{x} - \mathbf{b}\|_2 \quad \text{subject to} \quad \|L\,(\mathbf{x} - \mathbf{x}^*)\|_2 \le \alpha \tag{2.40}$$

$$\min \|L\,(\mathbf{x} - \mathbf{x}^*)\|_2 \quad \text{subject to} \quad \|A\,\mathbf{x} - \mathbf{b}\|_2 \le \delta , \tag{2.41}$$

where α and δ are nonzero parameters each playing the role as regularization parameter in (2.40) and (2.41), respectively. The solution to both these problems is identical to $\mathbf{x}_\lambda$ from

Tikhonov's method for suitably chosen values of λ that depend in a nonlinear way on α and δ. The solution to the first problem (2.40) is computed as follows: if $\|L\,(\mathbf{x}_{\mathrm{LSQ}}-\mathbf{x}_0)\|_2 \le \alpha$ then $\lambda \leftarrow 0$ and $\mathbf{x}_\lambda \leftarrow \mathbf{x}_{\mathrm{LSQ}}$, else use an iterative scheme to solve

$$\min \|A\,\mathbf{x}_\lambda - \mathbf{b}\|_2 \qquad \text{subject to} \qquad \|L\,(\mathbf{x}_\lambda - \mathbf{x}^*)\|_2 = \alpha$$

for λ and $\mathbf{x}_\lambda$. Similarly, the solution to the second problem (2.41) is computed as follows (where $\mathbf{x}_0$ is given by Eq. (2.30)): if $\|A\,\mathbf{x}_0 - \mathbf{b}\|_2 \le \delta$ then $\lambda \leftarrow \infty$ and $\mathbf{x}_\lambda \leftarrow \mathbf{x}_0$, else use an iterative scheme to solve

$$\min \|L\,(\mathbf{x}_\lambda - \mathbf{x}^*)\|_2 \qquad \text{subject to} \qquad \|A\,\mathbf{x}_\lambda - \mathbf{b}\|_2 = \delta$$

for λ and $\mathbf{x}_\lambda$. In REGULARIZATION TOOLS, routines lsqi and discrep solve (2.40) and (2.41), respectively. The name "discrep" is related to the discrepancy principle for choosing the regularization parameter, cf. Section 2.9. An efficient algorithm for solving (2.40) when A is large and sparse, based on Gauss quadrature and Lanczos bidiagonalization, is described in [30].

2.7.3. TSVD, MTSVD, and TGSVD

A fundamental observation regarding the abovementioned methods is that they circumvent the ill conditioning af A by introducing a new problem (2.39) with a well-conditioned coefficient matrix $\begin{pmatrix} A \\ \lambda L \end{pmatrix}$ with full rank. A different way to treat the ill-conditioning of A is to derive a new problem with a well-conditioned *rank deficient* coefficient matrix. A fundamental result about rank deficient matrices, which can be derived from the SVD of A, is that the closest rank-k approximation A_k to A—measured in the 2-norm—is obtained by truncating the SVD expansion in (2.8) at k, i.e., A_k is given by

$$A_k = \sum_{i=1}^{k} \mathbf{u}_i\,\sigma_i\,\mathbf{v}_i^T\,, \qquad k \le n\,. \tag{2.42}$$

The truncated SVD (TSVD) [72, 38, 42] and the modified TSVD (MTSVD) [49] regularization methods are based on this observation in that one solves the problems

$$\min \|\mathbf{x}\|_2 \qquad \text{subject to} \qquad \min \|A_k\,\mathbf{x} - \mathbf{b}\|_2 \tag{2.43}$$

$$\min \|L\,\mathbf{x}\|_2 \qquad \text{subject to} \qquad \min \|A_k\,\mathbf{x} - \mathbf{b}\|_2\,, \tag{2.44}$$

where A_k is the rank-k matrix in Eq. (2.42). The solutions to these two problems are given by

$$\mathbf{x}_k = \sum_{i=1}^{k} \frac{\mathbf{u}_i^T\mathbf{b}}{\sigma_i}\,\mathbf{v}_i \tag{2.45}$$

$$\mathbf{x}_{L,k} = \mathbf{x}_k - V_k\,(L\,V_k)^\dagger L\,\mathbf{x}_k\,, \tag{2.46}$$

where $(L\,V_k)^\dagger$ is the pseudoinverse of $L\,V_k$, and $V_k \equiv (\mathbf{v}_{k+1}, \ldots, \mathbf{v}_n)$. In other words, the correction to $\mathbf{x}_k$ in (2.46) is the solution to the following least squares problem

$$\min \|(L\,V_k)\,\mathbf{z} - L\,\mathbf{x}_k\|_2\,.$$

We note in passing that the TSVD solution $\mathbf{x}_k$ is the only regularized solution which has no component in the numerical null-space of A, spanned by the columns of V_k. All other regularized solutions, exemplified by the MTSVD solution $\mathbf{x}_{L,k}$, has some component in A's numerical null space in order to achieve the desired properties of the solution, as controlled by the matrix L.

As an alternative to the abovementioned MTSVD method for general-form problems one can generalize the TSVD method to the GSVD setting [41, 44]. The resulting method, truncated GSVD (TGSVD), is easiest to introduce via the standard-form transformation from §2.6.2 with $\bar{A} = A\, L_A^\dagger$, $\bar{\mathbf{b}} = \mathbf{b} - A\,\mathbf{x}_0$, and $\mathbf{x} = L_A^\dagger \bar{\mathbf{x}} + \mathbf{x}_0 \Rightarrow L\,\mathbf{x} = \bar{\mathbf{x}}$. In analogy with the TSVD method we now introduce a rank-k approximation $\bar{A}_k$ to $\bar{A}$ via its SVD. Due to the SVD-GSVD relations between $\bar{A}$ and (A, L), computation of the matrix $\bar{A}_k$ is essentially a "truncated GSVD" because $\bar{A}_k = \sum_{i=p-k+1}^{p} \mathbf{u}_i\, \gamma_i\, \mathbf{v}_i^T$. Then we define the truncated GSVD (TGSVD) solution as $\hat{\mathbf{x}}_{L,k} = L_A^\dagger \bar{\mathbf{x}}_k + \mathbf{x}_0$, where $\bar{\mathbf{x}}_k$ solves the problem

$$\min \|\bar{\mathbf{x}}\|_2 \qquad \text{subject to} \qquad \min \|\bar{A}_k\, \bar{\mathbf{x}} - \bar{\mathbf{b}}\|_2 \ . \tag{2.47}$$

Definition (2.47) together with the GSVD of (A, L) then immediately lead to the following simple expression of the TGSVD solution

$$\hat{\mathbf{x}}_{k,L} = \sum_{i=p-k+1}^{p} \frac{\mathbf{u}_i^T \mathbf{b}}{\sigma_i}\, \mathbf{x}_i + \sum_{i=p+1}^{n} (\mathbf{u}_i^T \mathbf{b})\, \mathbf{x}_i \ , \tag{2.48}$$

where the last term is the component $\mathbf{x}_0$ (2.30) in the null space of L. Defined this way, the TGSVD solution is a natural generalization of the TSVD solution $\mathbf{x}_k$. The TGSVD method is also a generalization of TSVD because both $\mathbf{x}_k$ and $\hat{\mathbf{x}}_{k,L}$ can be derived from the corresponding Tikhonov solutions (2.18) and (2.19) by substituting 0's and 1's for the Tikhonov filter factors f_i.

The TSVD, MTSVD, and TGSVD solutions are computed by the routines with the obvious names `tsvd`, `mtsvd`, and `tgsvd`.

2.7.4. Damped SVD/GSVD

A less know regularization method which is based on the SVD or the GSVD is the damped SVD/GSVD (damped SVD was introduced in [20], and our generalization to damped GSVD is obvious). Here, instead of using filter factors 0 and 1 as in TSVD and TGSVD, one introduces a smoother cut-off by means of filter factors f_i defined as

$$f_i = \frac{\sigma_i}{\sigma_i + \lambda} \quad (\text{for } L = I_n) \qquad \text{and} \qquad f_i = \frac{\sigma_i}{\sigma_i + \lambda\, \mu_i} \quad (\text{for } L \neq I_n) \ . \tag{2.49}$$

These filter factors decay slower than the Tikhonov filter factors and thus, in a sense, introduce less filtering. The damped SVD/GSVD solutions are computed by means of routine `dsvd`.

2.7.5. Maximum Entropy Regularization

This regularization method is frequently used in image reconstruction and related applications where a solution with positive elements is sought. In maximum entropy regularization, the following nonlinear function is used as side constraint:

$$\Omega(\mathbf{x}) = \sum_{i=1}^{n} x_i \log(w_i\, x_i) \;, \tag{2.50}$$

where x_i are the positive elements of the vector $\mathbf{x}$, and $w_1, \ldots, w_n$ are n weights. Notice that $-\Omega(\mathbf{x})$ measures the entropy of $\mathbf{x}$, hence the name of this regularization method. The mathematical justification for this particular choice of $\Omega(\mathbf{x})$ is that it yields a solution $\mathbf{x}$ which is most objective, or maximally uncommitted, with respect to missing information in the right-hand side, cf. e.g. [64].

Maximum entropy regularization is implemented in REGULARIZATION TOOLS in the routine `maxent` which uses a nonlinear conjugate gradient algorithm [28, §4.1] with inexact line search to compute the regularized solution. The typical step in this method has the form

$$\begin{aligned} \mathbf{x}^{(k+1)} &\leftarrow \mathbf{x}^{(k)} + \alpha_k\, \mathbf{p}^{(k)} \\ \mathbf{p}^{(k+1)} &\leftarrow -\nabla F(\mathbf{x}^{(k+1)}) + \beta_k\, \mathbf{p}^{(k)} \end{aligned} \tag{2.51}$$

in which F is the function to be minimized,

$$F(\mathbf{x}) = \|A\,\mathbf{x} - \mathbf{b}\|_2^2 + \lambda^2 \sum_{i=1}^{n} x_i \log(w_i\, x_i) \;,$$

and F's gradient is given by

$$\nabla F(\mathbf{x}) = 2\, A^T (A\,\mathbf{x} - \mathbf{b}) + \lambda^2 \begin{pmatrix} 1 + \log(w_1 x_1) \\ \vdots \\ 1 + \log(w_n x_n) \end{pmatrix} \;.$$

In Algorithm (2.51), the step-length parameter α_k miminizes $F(\mathbf{x}^{(k)} + \alpha_k\, \mathbf{p}^{(k)})$ with the constraint that all elements of $\mathbf{x}^{(k)} + \alpha_k\, \mathbf{p}^{(k)}$ be positive, and it is computed by means of an inexact line search. Then β_k is computed by

$$\beta_k = (\nabla F(\mathbf{x}^{(k+1)}) - \nabla F(\mathbf{x}^{(k)}))^T \nabla F(\mathbf{x}^{(k+1)}) \,/\, \|\nabla F(\mathbf{x}^{(k+1)})\|_2^2 \;.$$

This choice of β_k has the potential advantage that it gives "automatic" restart to the steepest descent direction in case of slow convergence.

2.7.6. Truncated Total Least Squares

The last direct regularization method included in REGULARIZATION TOOLS is truncated total least squares (TTLS). For rank deficient matrices, total least squares [26] takes its basis in an SVD of the compound matrix $(A\ \mathbf{b}) = \tilde{U}\,\tilde{\Sigma}\,\tilde{V}^T$ with the matrix $\tilde{V} \in \mathbb{R}^{(n+1)\times(n+1)}$ partitioned such that

$$\tilde{V} = \begin{pmatrix} \tilde{V}_{11} & \tilde{V}_{12} \\ \tilde{V}_{21} & \tilde{V}_{22} \end{pmatrix} \;, \qquad \tilde{V}_{11} \in \mathbb{R}^{n\times k} \;, \qquad \tilde{V}_{22} \in \mathbb{R}^{1\times(n+1-k)} \;, \tag{2.52}$$

where k is the numerical rank of A. Then the TLS solution is given by

$$\tilde{\mathbf{x}}_k = -\tilde{V}_{12}\,\tilde{V}_{22}^{\dagger} = -\tilde{V}_{12}\,\tilde{V}_{22}^T \,/\, \|\tilde{V}_{22}\|_2^2 \;. \tag{2.53}$$

The TLS solution is robust to perturbations of A because inaccuracies in A are explicitly taken into account in the TLS method. Therefore, for discrete ill-posed problems with no gap in the singular value spectrum of A, it makes sense to define a truncated TLS solution by means of (2.53) where k then plays the role of the regularization parameter; see [27] for more details. The truncated TLS solution is computed by means of routine **ttls**.

2.8. Iterative Regularization Methods

This section describes the iterative regularization methods included in REGULARIZATION TOOLS. We stress that our Matlab routines should be considered as model implementations; real implementations should incorporate any sparsity and/or structure of the matrix A. We shall first describe standard-form versions of the methods and then describe the extension necessary for treating general-form problems. For more details about these and other iterative methods, cf. [37, Chapter 6–7].

2.8.1. Conjugate Gradients and LSQR

The conjugate gradient (CG) algorithm is a well-known method for solving sparse systems of equations with a symmetric positive definite coefficient matrix. In connection with discrete ill-posed problems, it is an interesting fact that when the CG algorithm is applied to the unregularized normal equations $A^T A\, \mathbf{x} = A^T \mathbf{b}$ (implemented such that $A^T A$ is not formed) then the low-frequency components of the solution tend to converge faster than the high-frequency components. Hence, the CG process has some inherent regularization effect where the number of iterations plays the role of the regularization parameter. The kth step of the CG process essentially has the form

$$\begin{array}{lll} \beta_k & \leftarrow & \|\mathbf{q}^{(k-1)}\|_2^2 / \|\mathbf{q}^{(k-2)}\|_2^2 \\ \mathbf{p}^{(k)} & \leftarrow & \mathbf{q}^{(k-1)} + \beta_k\, \mathbf{p}^{(k-1)} \\ \alpha_k & \leftarrow & \|\mathbf{q}^{(k-1)}\|_2^2 / \|A^T A\, \mathbf{p}^{(k)}\|_2^2 \\ \mathbf{x}^{(k)} & \leftarrow & \mathbf{x}^{(k-1)} + \alpha_k\, \mathbf{p}^{(k)} \\ \mathbf{q}^{(k)} & \leftarrow & \mathbf{q}^{(k-1)} - \alpha_k\, A^T A\, \mathbf{p}^{(k)} \end{array} \tag{2.54}$$

where $\mathbf{x}^{(k)}$ is the approximation to $\mathbf{x}$ after k iterations, while $\mathbf{p}^{(k)}$ and $\mathbf{q}^{(k)}$ are two auxiliary iteration vectors of length n.

To explain this regularizing effect of the CG method, we introduce the Krylov subspace

$$\mathcal{K}_k(A^T A, A^T \mathbf{b}) = \mathrm{span}\{A^T \mathbf{b}, A^T A\, A^T \mathbf{b}, \ldots, (A^T A)^{k-1} A^T \mathbf{b}\}$$

associated with the kth step of the CG algorithm applied to $A^T A\, \mathbf{x} = A^T \mathbf{b}$. It is also convenient to introduce the Ritz polynomial P_k associated with step k:

$$P_k(\sigma) = \prod_{j=1}^{k} \frac{(\theta_j^{(k)})^2 - \sigma^2}{(\theta_j^{(k)})^2} \, . \tag{2.55}$$

Here, $(\theta_j^{(k)})^2$ are the Ritz values, i.e., the k eigenvalues of $A^T A$ restricted to the Krylov subspace $\mathcal{K}_k(A^T A, A^T \mathbf{b})$. The large Ritz values are approximations to some of the large eigenvalues σ_i^2 of the cross-product matrix $A^T A$. Then the filter factors associated with the solution after k steps of the CG algorithm are given by

$$f_i^{(k)} = 1 - P_k(\sigma_i) \ , \qquad i = 1, \dots, k \ . \tag{2.56}$$

As k increases, and the Ritz values converge to some of the eigenvalues of $A^T A$, then for selected i and j we have $\theta_j^{(k)} \approx \sigma_i$. Moreover, as k increases these approximations improve while, simultaneously, more eigenvalues of $A^T A$ are being approximated by the additional Ritz values.

Equations (2.55) and (2.56) for the CG filter factors shed light on the regularizing property of the CG method. After k iterations, if all the largest Ritz values $(\theta_j^{(k)})^2$ have converged to all the largest eigenvalues σ_i^2 of $A^T A$, then the corresponding $P_k(\sigma_i) \approx 0$ and the filter factors associated with these σ_i will therefore be close to one. On the other hand, for all those eigenvalues smaller than the smallest Ritz value, the corresponding filter factors satisfy

$$f_i^{(k)} = \sigma_i^2 \sum_{j=1}^{k} (\theta_j^{(k)})^{-2} + O\left(\sigma_i^4 (\theta_k^{(k)})^{-2} (\theta_{k-1}^{(k)})^{-2}\right) \ ,$$

showing that these filter factors decay like σ_i^2 for $\sigma_i < \theta_k(k)$.

From this analysis of the CG filter factors we see that the CG process indeed has a regularizing effect if the Ritz values converge to the eigenvalues of $A^T A$ in their natural order, starting with the largest. When this is the case, we are sure that the CG algorithm is a regularizing process with the number of iterations k as the regularization parameter. Unfortunately, proving that the Ritz values actually converge in this order is a difficult task. For problems with a gap in the singular value spectrum of A it is proved in [47, §6.4] that all the large eigenvalues of $A^T A$ will be approximated by Ritz values before any of the small eigenvalues of $A^T A$ get approximated. The case where the singular values of A decay gradually to zero with no gap in the spectrum is more difficult to analyze—but numerical examples and model problems [70, 71] indicate that the desired convergence of the Ritz values actually holds as long as the discrete Picard condition is satisfied for the unperturbed component of the right-hand side and there is a good separation among the large singular values of A.

To put the CG method into the common framework from the previous section, we notice that the solution $\mathbf{x}^{(k)}$ after k CG steps can be defined as

$$\min \|A\,\mathbf{x} - \mathbf{b}\|_2 \qquad \text{subject to} \qquad \mathbf{x} \in \mathcal{K}_k(A^T A, A^T \mathbf{b}) \ , \tag{2.57}$$

where $\mathcal{K}_k(A^T A, A^T \mathbf{b})$ is the Krylov subspace associated with the normal equations. Thus, we see that CG replaces the side constraint $\Omega(\mathbf{x}) = \|\mathbf{x}\|_2$ with the side constraint $\mathbf{x} \in \mathcal{K}_k(A^T A, A^T \mathbf{b})$. Obviously, if the Ritz values converge as desired, then the Krylov subspace satisfies $\mathcal{K}_k(A^T A, A^T \mathbf{b}) \approx \text{span}\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$ indicating that the CG solution $\mathbf{x}^{(k)}$ is similar to, say, the TSVD solution $\mathbf{x}_k$.

Even the best implementation of the normal-equation CG algorithm suffers from some loss of accuracy due to the implicit use of the cross-product matrix $A^T A$. An alternative iterative algorithm that avoids $A^T A$ completely is the algorithm LSQR [60]. This algorithm uses the Lanczos bidiagonalization algorithm [29, §9.3.4] to build up a lower bidiagonal matrix and, simultaneously, updates a QR factorization of this bidiagonal matrix. The QR factorization, in turn, is used to update the LSQR solution in each step. If B_k denotes the $(k+1) \times k$ bidiagonalization matrix generated in the kth step of LSQR, then the quantities $\theta_j^{(k)}$ in Eq. (2.55) are the singular values of this B_k. Hence, the LSQR algorithm is mathematically equivalent to the normal-equation CG algorithm in that the kth iteration vectors $\mathbf{x}^{(k)}$ in CG and LSQR are identical in exact arithmetic.

In real computations, the convergence of CG and LSQR is delayed due to the influence of the finite precision arithmetic, and the dimension of the subspace in which $\mathbf{x}^{(k)}$ lies does not increase in each step. As a consequence, $\mathbf{x}^{(k)}$ typically stays almost unchanged for a few steps, then changes to a new vector and stays unchanged again for some steps, etc. (The underlying phenomenon is related to CG and LSQR computing "ghost" eigenvalues and singular values, respectively, cf. [29, §9.2.5]). In LSQR, it is possible to store the so-called Lanczos vectors generated during the process and apply some reorthogonalization scheme to them, which prevents the abovementioned delay—but in practice it is usually less computationally demanding just to use LSQR without any reorthogonalization. Orthogonalization can also be applied to the normal equation residual vectors $A^T(A\,\mathbf{x}^{(k)} - \mathbf{b})$ in the CG algorithm.

There are several ways to implement the CG algorithm for the normal equations in a numerically stable fashion. The one used in the routine cgls in Regularization Tools is from [9, p. 289]. The implementation lsqr is identical to the original LSQR algorithm from [60].

Regarding the filter factors for CG and LSQR, we have found that the expression (2.56) using the Ritz polynomial is extremely sensitive to rounding errors. Instead, we compute the filter factors $f_i^{(k)}$ by means of numerically more robust recursions derived in [76] (see also [47]). Notice that the exact singular values σ_i of A are required to compute the filter factors; hence this option is mainly of pedagogical interest.

2.8.2. Bidiagonalization with Regularization

It is possible to modify the LSQR algorithm and derive a hybrid between a direct and an iterative regularization algorithm. The idea is to use the abovementioned Lanczos algorithm to build up the bidiagonal matrix B_k sequentially, and in each step to replace LSQR's QR factorization of B_k with a direct regularization scheme such as Tikhonov regularization or TSVD. These ideas are outlined in [8, 59]; see also the discussion [37, Chapter 7]. The work involved in the direct regularization process is small compared to the work in the iterative process because of the bidiagonal form of B_k. Again, reorthogonalization of the Lanczos vectors is possible but rarely used in practice.

One rationale behind this "hybrid" algorithm is that if the number k of Lanczos bidiagonalization steps is so large that B_k becomes ill-conditioned and needs regularization—because the singular values $\theta_k^{(k)}$ of B_k start to approximate some of the smaller singular values of A—then hopefully all the *large* singular values of A are approximated by singular values of B_k. When this is the case, then we are ensured that the "hybrid" method computes a proper regularized solution, provided of course that the explicit regularization in each step properly filters out the influence of the small singular values.

The second, and perhaps most important, rational behind the "hybrid" algorithm is that it requires a different stopping criterion which is not as dependent on choosing the correct k as the previously mentioned methods. Provided again that the explicit regularization scheme in each step is successful in filtering out the influence of the small singular values, then after a certain stage k the iteration vector $\mathbf{x}^{(k)}$ of the "hybrid" algorithm will hardly change. This is so because all the components associated with the large singular values have been captured while the components associated with the small singular values are filtered out. Thus, the stopping criteria should now be based on the relative change in $\mathbf{x}^{(k)}$. With this stopping criteria, we see that taking too many steps in the "hybrid" algorithm will not deteriorate the iteration vector, but merely increase the computational effort.

For convenience, Regularization Tools provides the routine lanc_b for computing the lower bidiagonal matrix B_k as well as the corresponding left and right Lanczos vectors. The routine csvd computes the SVD of B_k. The user can then combine the necessary routines to form a specific "hybrid" algorithm.

2.8.3. The ν-Method

Both CG and LSQR converge rather fast to a regularized solution with damped high-frequency components, and if the iterations are continued then the high-frequency components very soon start to dominate the iteration vector. For some methods for choosing the regularization parameter, i.e., the number of iterations k (such as the L-curve criterion described in Section 2.9), this is a favorable property. However, there are other circumstances in which is is more desirable to have an iterative scheme that converges slower and thus is less sensitive to the choice of k.

This is exactly the philosophy behind the ν-method [11] which is similar to the CG method except that the coefficients α_k and β_k used to update the iteration vectors in Algorithm (2.54) are problem independent and depend only on the iteration number k and a prespecified constant ν satisfying $0 < \nu < 1$:

$$\alpha_k = 4\,\frac{(k+\nu)(k+\nu+\frac{1}{2})}{(k+2\nu)(k+2\nu+\frac{1}{2})}\,, \qquad \beta_k = \frac{(k+\nu)(k+1)(k+\frac{1}{2})}{(k+2\nu)(k+2\nu+\frac{1}{2})(k+\nu+1)}\,. \tag{2.58}$$

(It can be shown that the filter factors can be expressed in terms of Jacobi polynomials).

A slight inconvenience with the ν-method is that it requires the problem to be scaled such that $\|A\|_2$ is slightly less than one, otherwise the method either diverges or converges too slow. A practical way to treat this difficulty [37, §6.3] is use the Lanczos bidiagonalization algorithm to compute a good approximation $\theta_1^{(k)} = \|B_k\|_2$ to $\|A\|_2$ and then rescale A and $\mathbf{b}$ by $0.99/\theta_1^{(k)}$. Usually a few Lanczos steps are sufficient. This initialization process can also be used to provide the ν-method with a good initial guess, namely, the iteration vector $\mathbf{x}^{(k)}$ after a few LSQR steps.

The routine nu in Regularization Tools implements the ν-method as described in [11] with the abovementioned rescaling. The LSQR start-vector is not used, but can be supplied by the user if desired.

2.8.4. Extension to General-Form Problems

So far we have described several iterative methods for treating regularization problems in standard form; we shall now briefly describe the necessary extension to these methods for treating problems in general form. The idea presented here is originally from [35, 36]. From the discussion of the standard-form transformation for iterative methods in §2.6.2, we see that essentially we must apply the above standard-form iterative methods to a transformed problem with $\bar{A}$ and $\bar{\mathbf{b}}$. I.e., according to (2.57) we must compute the solution $\bar{\mathbf{x}}^{(k)}$ to the problem

$$\min \|\bar{A}\,\bar{\mathbf{x}} - \bar{\mathbf{b}}\|_2 \qquad \text{subject to} \qquad \bar{\mathbf{x}} \in \mathcal{K}_k(\bar{A}^T \bar{A}, \bar{A}^T \bar{\mathbf{b}}) \ . \tag{2.59}$$

If we use the alternative formulation from §2.6.2 with $\bar{A} = A\,L_A^\dagger$ and $\bar{\mathbf{b}} = \mathbf{b} - A\,\mathbf{x}_0$, then the standard-form transformation can be "built into" the iterative scheme. In this way, we work directly with $\mathbf{x}^{(k)}$ and avoid the back-transformation from $\bar{\mathbf{x}}^{(k)}$ to $\mathbf{x}^{(k)}$. To derive this technique, consider the side constraint in (2.59) which implies that there exist constants $\xi_0, \ldots, \xi_{k-1}$ such that

$$\bar{\mathbf{x}}^{(k)} = \sum_{i=0}^{k-1} \xi_i \, (\bar{A}^T \bar{A})^i \, \bar{A}^T \bar{\mathbf{b}} \ .$$

If we insert $\bar{A} = A\,L_A^\dagger$ and $\bar{\mathbf{b}} = \mathbf{b} - A\,\mathbf{x}_0$ into this relation, we obtain

$$\bar{\mathbf{x}}^{(k)} = \sum_{i=0}^{k-1} \xi_i \left((L_A^\dagger)^T A^T A\, L_A^\dagger \right)^i \, (L_A^\dagger)^T A^T (\mathbf{b} - A\,\mathbf{x}_0) \ .$$

Using Eqs. (2.31) and (2.30) for $L_A^\dagger$ and $\mathbf{x}_0$ together with the GSVD it is straightforward to show that $(L_A^\dagger)^T A^T A\,\mathbf{x}_0 = \mathbf{0}$. Thus, by inserting the above expression for $\bar{\mathbf{x}}^{(k)}$ into the back-transformation $\mathbf{x}^{(k)} = L_A^\dagger \bar{\mathbf{x}}^{(k)} + \mathbf{x}_0$, we obtain

$$\mathbf{x}^{(k)} = \sum_{i=0}^{k-1} \xi_i \left(L_A^\dagger \, (L_A^\dagger)^T A^T A \right)^i \, L_A^\dagger \, (L_A^\dagger)^T A^T \mathbf{b} + \mathbf{x}_0 \ . \tag{2.60}$$

From this relation we see that we can consider the matrix $L_A^\dagger \, (L_A^\dagger)^T$ a "preconditioner" for the iterative methods, and we stress that the purpose of the "preconditioner" is not to improve the condition number of the iteration matrix but rather to ensure that the "preconditioned" iteration vector $\mathbf{x}^{(k)}$ lies in the correct subspace and thus minimizes $\|L\,\mathbf{x}^{(k)}\|_2$. Minimization of the correct residual norm is ensured by Eq. (2.38).

"Preconditioning" is easy to build into CG, LSQR, and the ν-method by means of the algorithms in (2.36) from §2.6.2. The special "preconditioned" versions are implemented as routines pcgls, plsqr, and pnu in Regularization Tools.

2.9. Methods for Choosing the Regularization Parameter

No regularization package is complete without routines for computation of the regularization parameter. As we have already discussed in Section 2.5 a good regularization parameter should yield a fair balance between the perturbation error and the regularization error in the regularized solution. Throughout the years a variety of parameter-choice strategies have been proposed. These methods can roughly be divided into two classes depending on their assumption about $\|\mathbf{e}\|_2$, the norm of the perturbation of the right-hand side. The two classes can be characterized as follows:

1. Methods based on knowledge, or a good estimate, of $\|\mathbf{e}\|_2$.

2. Methods that do not require $\|\mathbf{e}\|_2$, but instead seek to extract the necessary information from the given right-hand side.

For many of these methods, the convergence rate for the solution as $\|\mathbf{e}\|_2 \to 0$ has been analyzed [25, 33, 75]. Four parameter-choice routines are included in REGULARIZATION TOOLS, one from class 1 and three from class 2.

The only method belonging to class 1 is the *discrepancy principle* [56, §27] which, in all simplicity, amounts to choosing the regularization parameter such that the residual norm for the regularized solution satisfies

$$\|A\,\mathbf{x}_{\rm reg} - \mathbf{b}\|_2 = \|\mathbf{e}\|_2 \ . \tag{2.61}$$

When a good estimate for $\|\mathbf{e}\|_2$ is known, this method yields a good regularization parameter corresponding to a regularized solution immediately to the right of the L-curve's corner. Due to the steep part of the L-curve we see that an underestimate of $\|\mathbf{e}\|_2$ is likely to produce an underregularized solution with a very large (semi)norm. On the other hand, an overestimate of $\|\mathbf{e}\|_2$ produces an overregularized solution with too large regularization error.

The three methods from class 2 that we have included in REGULARIZATION TOOLS are the L-curve criterion, generalized cross-validation, and the quasi-optimality criterion. The *L-curve criterion* has already been discussed in connection with the introduction of the L-curve in Section 2.5. Our implementation follows the description in [48] closely. For a continuous regularization parameter λ we compute the curvature of the curve

$$(\log \|A\,\mathbf{x}_\lambda - \mathbf{b}\|_2 \, , \, \log \|L\,\mathbf{x}_\lambda\|_2)$$

(with λ as its parameter) and seek the point with maximum curvature, which we then define as the L-curve's corner. When the regularization parameter is discrete we approximate the discrete L-curve in log-log scale by a 2D spline curve, compute the point on the spline curve with maximum curvature, and define the corner of the discrete L-curve as that point which is closest to the corner of the spline curve.

Generalized cross-validation (GCV) is based on the philosophy that if an arbitrary element b_i of the right-hand side $\mathbf{b}$ is left out, then the corresponding regularized solution should predict this observation well, and the choice of regularization parameter should be independent of an orthogonal transformation of $\mathbf{b}$; cf. [77, Chapter 4] for more details. This leads to choosing the regularization parameter which minimizes the GCV function

$$G \equiv \frac{\|A\,\mathbf{x}_{\rm reg} - \mathbf{b}\|_2^2}{(\mathrm{trace}(I_m - A\,A^I))^2} \ , \tag{2.62}$$

where A^I is a matrix which produces the regularized solution $\mathbf{x}_{\rm reg}$ when multiplied with $\mathbf{b}$, i.e., $\mathbf{x}_{\rm reg} = A^I\mathbf{b}$. Note that G is defined for both continuous and discrete regularization parameters. The denominator in (2.62) can be computed in $O(n)$ operations if the bidiagonalization algorithm from Section 2.7 is used [24]. Alternatively, the filter factors can be used to evaluate the denominator by means of the simple expression

$$\mathrm{trace}(I_m - A\,A^I) = m - (n-p) - \sum_{i=1}^{p} f_i \ . \tag{2.63}$$

This is the approach used in routine **gcv**. In [45] it is illustrated that the GCV method indeed seeks to balance the perturbation and regularization errors and thus, in turn, is related to the corner of the L-curve.

The final method included in REGULARIZATION TOOLS is the *quasi-optimality criterion* [56, §27]. This method is, strictly speaking, only defined for a continuous regularization parameter λ and amounts to minimizing the function

$$Q \equiv \lambda \left\| \frac{d\mathbf{x}_\lambda}{d\lambda} \right\|_2 = \left(\sum_{i=1}^{p} \left(f_i\,(1-f_i)\,\frac{\mathbf{u}_i^T\mathbf{b}}{\gamma_i} \right)^2 \right)^{1/2} . \tag{2.64}$$

As demonstrated in [45], under certain assumptions the approach also corresponds to finding a good balance between perturbation and regularization errors in $\mathbf{x}_\lambda$. For a discrete regularization parameter k, we use $\lambda = \gamma_k$ and the approximations

$$\left\| \frac{d\mathbf{x}_\lambda}{d\lambda} \right\|_2 \approx \frac{\|\Delta\mathbf{x}_k\|_2}{|\Delta\lambda|} \ , \qquad \|\Delta\mathbf{x}_k\|_2 = \frac{\mathbf{u}_k^T\mathbf{b}}{\gamma_k} \ , \qquad \Delta\lambda = \gamma_{k+1} - \gamma_k \approx \gamma_k$$

to obtain the expressions

$$Q \approx \frac{\mathbf{u}_k^T\mathbf{b}}{\sigma_k} \quad \text{if} \quad L = I_n \ , \qquad Q \approx \frac{\mathbf{u}_k^T\mathbf{b}}{\gamma_k} \quad \text{if} \quad L \neq I_n \ . \tag{2.65}$$

The discrepancy principle is implemented in routine **discrep** described in §2.7.2 in connection with direct regularization methods. The L-curve criterion is implemented in the two routines **l_curve** and **l_corner**, while GCV is is provided by routine **gcv**. Finally, the quasi-optimality criterion is implemented in routine **quasiopt**.

3. Regularization Tools Tutorial

The purpose of this section is to give a brief introduction to the use of the routines in Regularization Tools by means of some fairly simple examples. In particular, we show how to compute regularized solutions and how to evaluate these solutions by various graphical tools. Although the examples given below do not touch upon all the features of Regularization Tools, they illustrate the fundamental ideas underlying the package, namely, modularity and regularity between the routines. For convenience, the examples are also available in the annotated script regudemo, with appropriate pause statements added.

3.1. The Discrete Picard Condition

We shall first illustrate the use of the routine picard for visually checking the discrete Picard condition, cf. Section 2.4. First, we generate a discrete ill-posed problem using one of the many built-in test problems; the one used here is shaw which is a one-dimensional model of an image restoration problem. Then we add white noise to the right-hand side, thus producing a more "realistic" problem.

Before performing the analysis of the problem, we compute the SVD of the coefficient matrix; this is the typical situation in Regularization Tools, since most of the routines make use of either the SVD (for standard-form problems) or the GSVD (for general-form problems). We then use picard to plot the singular values and the Fourier coefficients for both the unperturbed and the perturbed problem, see Fig. 3.1.

```
[A,b_bar,x] = shaw (32);
randn ('seed',70957);
e = 1e-3*rand (size (b_bar)); b = b_bar + e;
[U,s,V] = csvd (A);
subplot (2,1,1); picard (U,s,b_bar);
subplot (2,2,2); picard (U,s,b);
```

Clearly, most of the Fourier coefficients for the unperturbed problem satisfy the discrete Picard condition—although eventually, for large i, both the singular values and the Fourier coefficients become dominated by rounding errors. For the "noisy" test problem, we see that the Fourier coefficients for the right-hand side become dominated by the perturbation for i much smaller than before. We also see that in the left part of curve the Fourier coefficients still decay faster than the singular values, indicating that the unperturbed right-hand side satisfies the discrete Picard condition. To regularize this problem, we

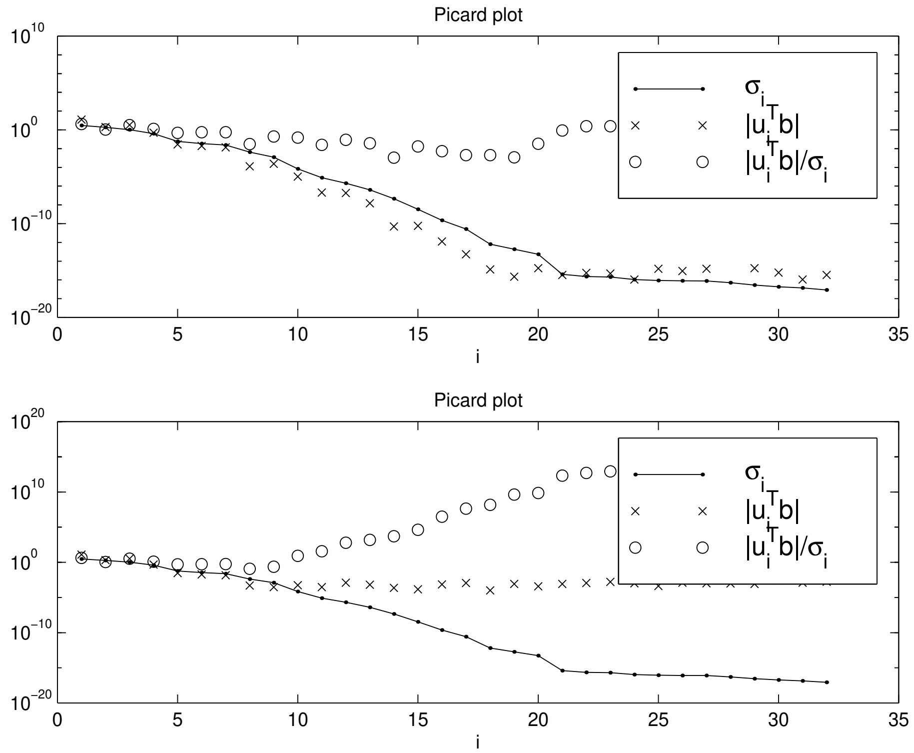


Figure 3.1: Output from picard for the "pure" and the "noisy" test problems.

should preferably dampen the components for which the perturbation dominates, and leave the rest of the components (for small i) intact.

3.2. Filter Factors

In this part of the tutorial we consider only the "noisy" test problem from Example 3.1, we compute regularized solutions by means of Tikhonov's method and LSQR, and we compute the filter factors for two regularization methods.

```
lambda = [1,3e-1,1e-1,3e-2,1e-2,3e-3,1e-3,3e-4,1e-4,3e-5];
X_tikh = tikhonov (U,s,V,b,lambda);
F_tikh = fil_fac (s,lambda);
iter = 30; reorth = 0;
[X_lsqr,rho,eta,F_lsqr] = lsqr (A,b,iter,reorth,s);
subplot (2,2,1); surf (X_tikh), axis ('ij'), title ('Tikhonov solutions')
subplot (2,2,2); surf (log10 (F_tikh)), axis ('ij'), title ('Tikh filter factors, log scale')
subplot (2,2,3); surf (X_lsqr (:,1:17)), axis ('ij'), title ('LSQR solutions')
subplot (2,2,4); surf (log10 (F_lsqr)), axis ('ij'), title ('LSQR filter factors, log scale')
```

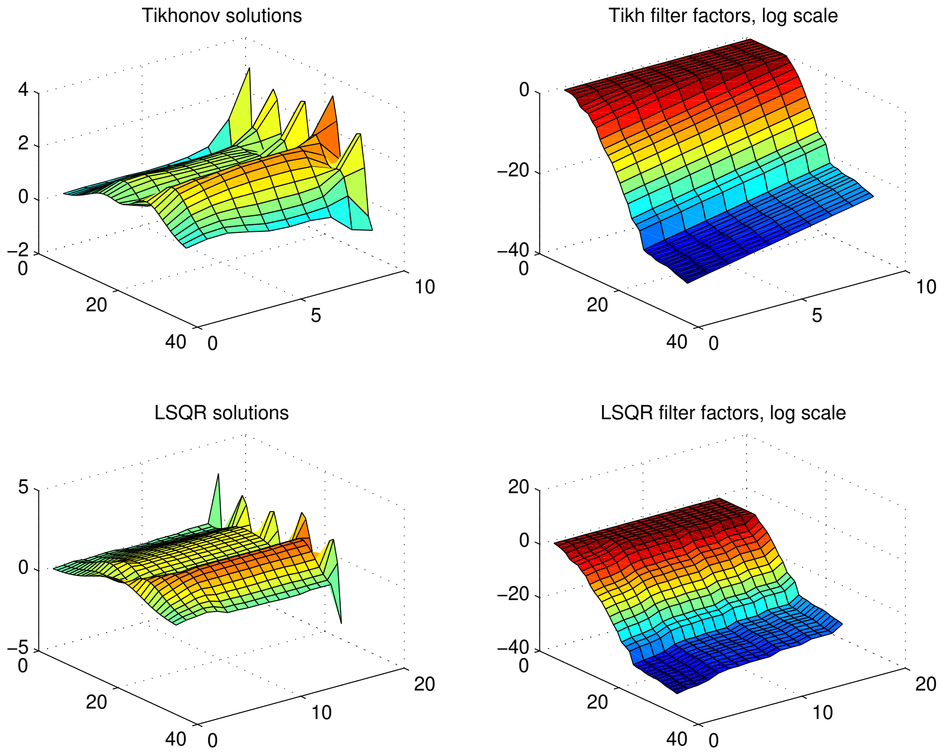


Figure 3.2: Regularized solutions and filter factors for the “noisy” test problem.

We use the command **mesh** to plot all the regularized solutions, cf. Fig. 3.2. This is a very convenient to show the dependence of the solution on the regularization parameter. The same technique is used to display the the corresponding filter factors. We see the typical situation for regularization methods: first, when we apply much regularization, the solution is overregularized, i.e., it is too smooth; then it becomes a better approximation to the underlying, unperturbed solution; and eventually the solution becomes underregularized and starts to be dominated by the perturbation errors, and its (semi)norm “explodes”.

3.3. The L-Curve

The L-curve analysis plays an important role in the analysis phase of regularization problems. The L-curve displays the tradeoff between minimizing the two quantities in the regularization problem, namely, the residual norm and the solution (semi)norm, and it shows how these quantities depend on the regularization parameter. In addition, the L-curve can be used to compute the “optimal” regularization parameter as explained in Section 2.9 (we return to this aspect in the next example). The L-curve can also be used to investigate the similarity between different regularization methods—if their L-curves are close, then the regularized solutions are similar, cf. [45].

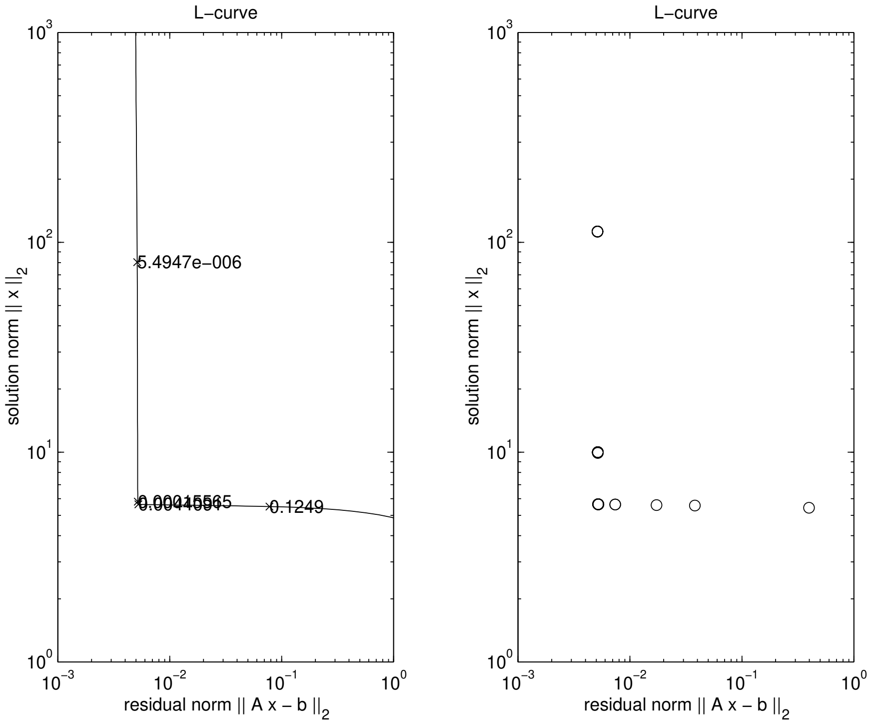


Figure 3.3: The L-curves for Tikhonov regularization and for LSQR.

```
subplot (1,2,1); l_curve (U,s,b); axis ([1e-3,1,1,1e3])
subplot (1,2,2); plot_lc (rho,eta,'o'), axis ([1e-3,1,1,1e3]
```

For the particular problem considered here, the "noisy" test problem from Example 3.1, the L-curves for both Tikhonov regularization and for LSQR, shown in Fig. 3.3, have a particularly sharp corner. This corner corresponds to a regularized solution where the perturbation error and the regularization error are balanced. The similarity of the L-curves for the two methods indicate the similarity between the methods.

3.4. Regularization Parameters

We shall now use the L-curve criterion and the GCV method to compute "optimal" regularization parameters for two methods: Tikhonov regularization and truncated SVD. This is very easy to do by means of the routines l_curve and gcv. We then use our knowledge about the exact solution to compute the relative errors in the four regularized solutions. The best result may depend on the particular computer's floating point arithmetic—for the Toshiba PC used here, the combination of truncated SVD and the L-curve criterion gives the best result.

If the Spline Toolbox is not available, then k_l = NaN and x_tsvd_l is set to **0**.

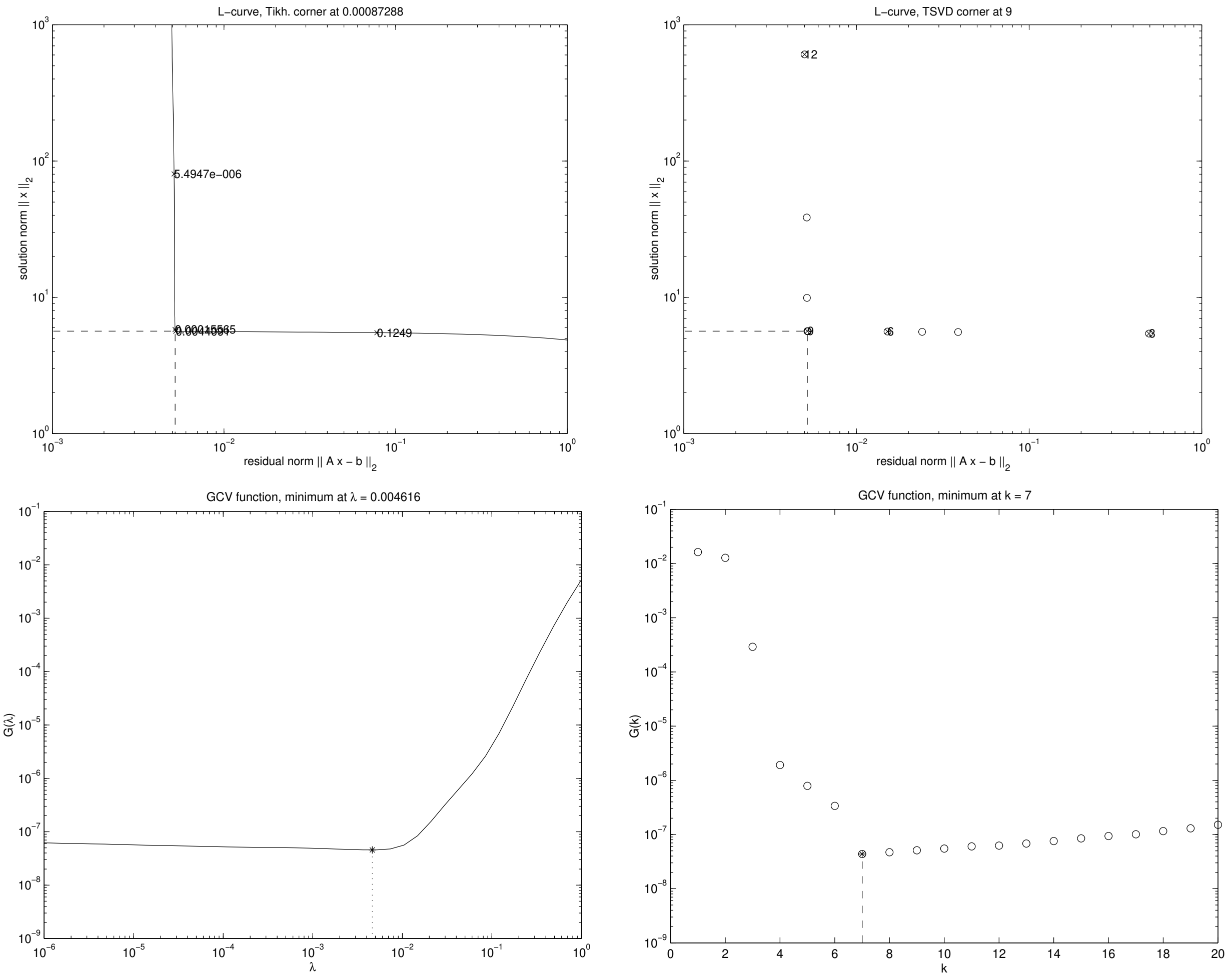


Figure 3.4: Comparison of the L-curve criterion and the GCV method for computing the "optimal" regularization parameter for Tikhonov's method and for truncated SVD.

```
lambda_l = l_curve (U,s,b); axis ([1e-3,1,1,1e3])
k_l = l_curve (U,s,b,'tsvd'); axis ([1e-3,1,1,1e3])
lambda_gcv = gcv (U,s,b); axis ([-6,0,-9,-1])
k_gcv = gcv (U,s,b,'tsvd'); axis ([0,20,1e-9,1e-1])

x_tikh_l = tikhonov (U,s,V,b,lambda_l);
x_tikh_gcv = tikhonov (U,s,V,b,lambda_gcv);
x_tsvd_l = tsvd (U,s,V,b,k_l);
x_tsvd_gcv = tsvd (U,s,V,b,k_gcv);
[norm (x−x_tikh_l),norm (x−x_tikh_gcv),...
    norm (x−x_tsvd_l),norm (x−x_tsvd_gcv)]/norm (x)
```

3.5. Standard Form Versus General Form

In this example we illustrate the difference between regularization in standard and general form. In particular, we show that regularization in general form is sometimes *necessary* to ensure the computation of a satisfactory solution. Unfortunately, this judgement cannot be automated, but requires insight from the user about the desired solution.

The test problem used here is the inverse Laplace transformation, and we generate a problem whose exact solution is $f(t) = 1 - \exp(-t/2)$. This solution obviously satisfies $f \to 1$ for $t \to \infty$, and the horizontal asymptotic part in the discretized solution for $n = 16$ is visible for indices $i > 8$.

```
n = 16; [A,b,x] = ilaplace (n,2);
b = b + 1e-4*randn (b);
L = get_l (n,1);
[U,s,V] = csvd (A); [UU,sm,XX] = cgsvd (A,L);
I = 1; for i=[3,6,9,12]
    subplot (2,2,I); plot (1:n,V(:,i)); axis ([1,n,-1,1])
    xlabel (['i = ',num2str(i)]), I = I + 1;
end
subplot (2,2,1), text (12,1.2,'Right singular vectors V(:,i)')
I = 1; for i=[n-2,n-5,n-8,n-11]
    subplot (2,2,I); plot (1:n,XX(:,i)); axis ([1,n,-1,1])
    xlabel (['i = ',num2str(i)]), I = I + 1;
end
subplot (2,2,1), text (10,1.2,'Right generalized singular vectors XX(:,i)')

k_tsvd = 7; k_tgsvd = 6;
X_I = tsvd (U,s,V,b,1:k_tsvd);
X_L = tgsvd (UU,sm,XX,b,1:k_tgsvd);
subplot (2,1,1); plot (1:n,X_I,1:n,x,'x'), axis ([1,n,0,1.2]), xlabel ('L = I')
subplot (2,2,2), plot (1:n,X_L,1:n,x,'x'), axis ([1,n,0,1.2]), xlabel ('L \neq I')
```

In this example we choose minimization of the first derivative as side constraint in the general-form regularization, i.e., we use $L = \mathrm{tridiag}(-1, 1)$. It is very instructive to first inspect the right singular vectors $\mathbf{v}_i$ and $\mathbf{x}_i$ from the SVD and the GSVD, respectively, cf. Fig. 3.5. Notice that none of the SVD vectors $\mathbf{v}_i$ include the abovementioned asymptotic part; hence, these vectors are not suited as basis vectors for a regularized solution. The GSVD vectors $\mathbf{x}_i$, on the other hand, do posess the the necessary asymptotic part, and they are therefore much better suited as basis vectors. For a thorough discussion of these aspects, cf. [74].

Let us now use the truncated SVD and the truncated GSVD to compute regularized solutions to this problem. The first 7 TSVD solutions and the first 6 TGSVD solutions are shown in Fig. 3.6. From our investigation of the right singular vectors, it is no surprise

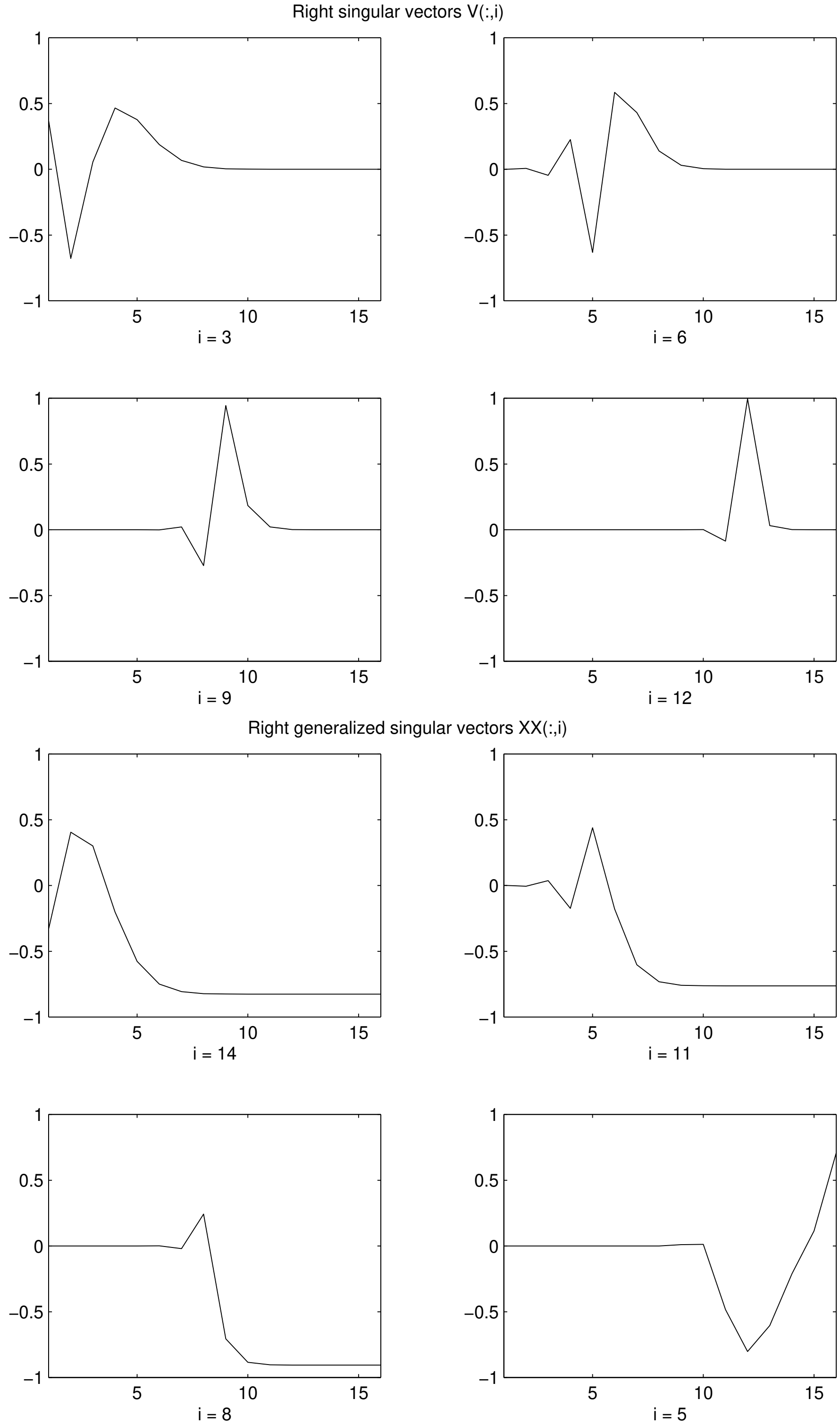


Figure 3.5: Comparison of the right singular vectors for SVD and GSVD for the inverse Laplace transformation test-problem.

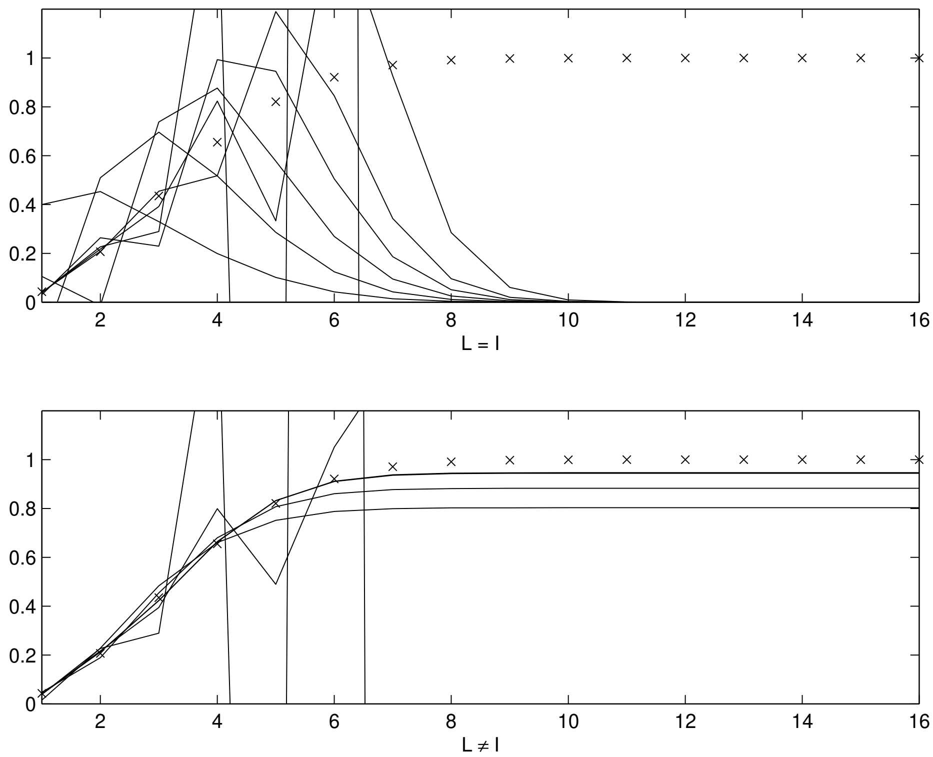


Figure 3.6: The first 7 TSVD solutions and the first 6 TGSVD solutions for the same test problem as in Fig. 3.5. The exact solution is shown with $\times$-markers.

that TSVD cannot reproduce the desired solution, while TGSVD indeed succeeds in doing so. The optimal regularization parameter for TGSVD is $k = 5$.

We notice that without our a priori knowledge part of the solution, we could not immediately discard the TSVD solutions!

3.6. No Square Integrable Solution

In this tutorial's final example we use the routine ursell to generate a test problem arising from discretization of a Fredholm integral equation of the first kind (2.1) with no square integrable solution [18, p. 6], i.e., the integral equation does not satisfy the Picard condition. The integral equation is

$$\int_0^1 \frac{1}{s+t+1} f(t)\, dt = 1 \ , \qquad 0 \le s \le 1 \ .$$

When we use picard to plot the singular values and the Fourier coefficients for the discrete problem, see Fig. 3.7, we immediately see that discrete ill-posed problem does not satisfy the discrete Picard condition, which indicates trouble! We stress, however, that such an analysis cannot show whether the trouble comes from the integral equation itself, from the discretization, or possibly from other sources.

```
[A,b] = ursell (32); [U,s,V] = csvd (A); picard (U,s,b);
```

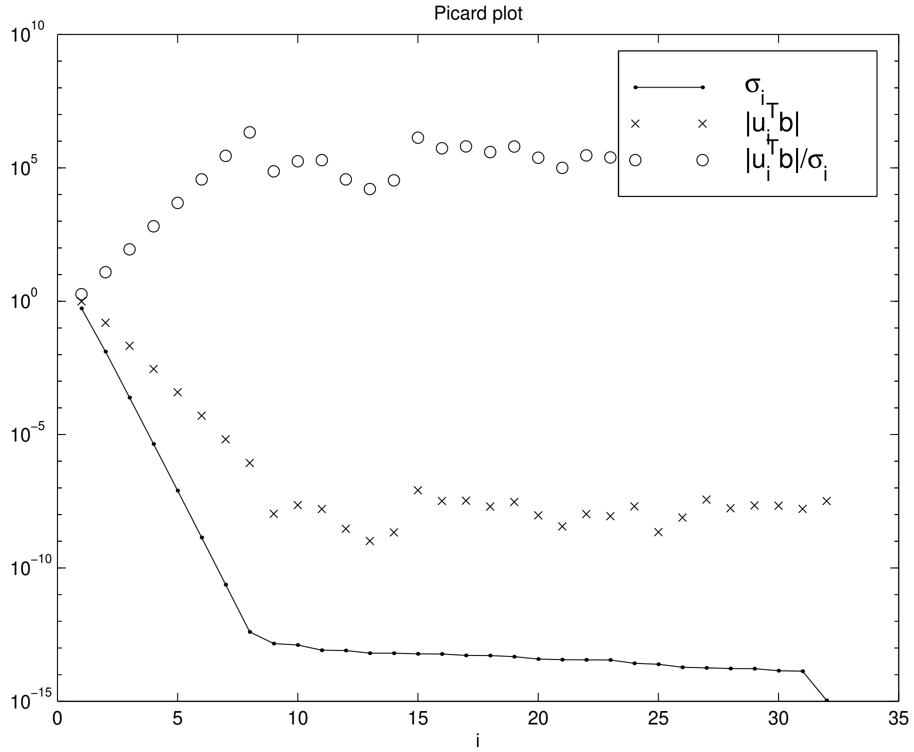


Figure 3.7: The output from picard for the test problem that does not satisfy the discrete Picard condition.

4. Regularization Tools Reference

This section contains detailed descriptions of all Regularization Tools routines. It begins with a list of the routines grouped by subject area and continues with the reference entries en alphabetical order. Information is also available through the on-line help facility.

REGULARIZATION ROUTINES	
cgls	Computes the least squares solution based on k steps of the conjugate gradient algorithm
discrep	Minimizes the solution (semi)norm subject to an upper bound on the residual norm (discrepancy principle)
dsvd	Computes a damped $SVD/GSVD$ solution
lsqi	Minimizes the residual norm subject to an upper bound on the (semi)norm of the solution
lsqr	Computes the least squares solution based on k steps of the $LSQR$ algorithm
maxent	Computes the maximum entropy regularized solution
mtsvd	Computes the modified $TSVD$ solution
nu	Computes the solution based on k steps of Brakhage's iterative ν-method
pcgls	Same as cgls, but for general-form regularization
plsqr	Same as lsqr, but for general-form regularization
pnu	Same as nu, but for general-form regularization
tgsvd	Computes the truncated $GSVD$ solution
tikhonov	Computes the Tikhonov regularized solution
tsvd	Computes the truncated SVD solution
ttls	Computes the truncated TLS solution

ANALYSIS ROUTINES	
fil_fac	Computes filter factors for some regularization methods
gcv	Plots the GCV function and computes its minimum
lagrange	Plots the Lagrange function $\|A\,\mathbf{x} - \mathbf{b}\|_2^2 + \lambda^2\,\|L\,\mathbf{x}\|_2^2$ and its derivative
l_corner	Locates the L-shaped corner of the L-curve
l_curve	Computes the L-curve, plots it, and computes its corner
picard	Plots the (generalized) singular values, the Fourier coefficients for the right-hand side, and a possibly smoothed curve of the solution's Fourier-coefficients
plot_lc	Plots an L-curve
quasiopt	Plots the quasi-optimality function and computes its minimum

TEST PROBLEMS	
baart	First kind Fredholm integral equation
blur	Image deblurring test problem
deriv2	Computation of second derivative
foxgood	Severely ill-posed test problem
heat	Inverse heat equation
ilaplace	Inverse Laplace transformation
parallax	Stellar parallax problem with real observations
phillips	Phillips' "famous" test problem
shaw	One-dimensional image restoration model
spikes	Test problem with a "spiky" solution
ursell	Integral equation with no square integrable solution
wing	Test problem with a discontinuous solution

STANDARD-FORM TRANSFORMATION	
gen_form	Transforms a standard-form solution back into the general-form setting
std_form	Transforms a general-form problem into one in standard form

UTILITY ROUTINES	
bidiag	Bidiagonalization of a matrix by Householder transformations
cgsvd	Computes the compact generalized SVD of a matrix pair
csvd	Computes the compact SVD of an $m \times n$ matrix
get_l	Produces a $(n-d) \times n$ matrix which is the discrete approximation to the dth order derivative operator
lanc_b	Performs k steps of the Lanczos bidiagonalization process with/without reorthogonalization
regutm	Generates random test matrices for regularization methods

AUXILIARY ROUTINES	
app_hh_l	Applies a Householder transformation from the left
gen_hh	Generates a Householder transformation
heb_new	Newton-Raphson iteration with Hebden's rational approximation, used in lsqi
lsolve	Inversion with A-weighted generalized inverse of L
ltsolve	Inversion with transposed A-weighted generalized inverse of L
newton	Newton-Raphson iteration, used in discrep
pinit	Initialization for treating general-form problems
pythag	Computes $\sqrt{a^2+b^2}$
regudemo	Tutorial introduction to REGULARIZATION TOOLS
spleval	Computes points on a spline or spline curve

The Test Problems

There are 12 built-in test problems in REGULARIZATION TOOLS. Ten of them are taken from the literature (cf. the following manual pages for references) while the remaining, spikes, is "cooked up" for this package. All of them have in common that they are easy to generate, and they share the characteristic features of discrete ill-posed problems mentioned in Section 2.3.

All the test problems are derived from discretizations of a Fredholm integral equation of the first kind. Two different discretization techniques are used: the quadrature method and the Galerkin method with orthonormal basis functions. In the *quadrature method* [18, Chapter 6], the integral is approximated by a weighted sum, i.e.,

$$\int_a^b K(s,t)\, f(t)\, dt \approx I_n(s) = \sum_{i=1}^{n} w_j\, K(s,t_j)\, f(T_j) \ .$$

In particular, for the midpoint rule we use $w_j = (b-a)/n$ and $t_j = (j - \frac{1}{2})(b-a)/n$, $j = 1,\ldots,n$. Collocation in the n points $s_1,\ldots,s_n$ then leads to the requirements $I_n(s_i) = g(s_i)$, $i = 1,\ldots,n$. Hence, we obtain a system of linear algebraic equations $A\,\mathbf{x} = \mathbf{b}$ with elements given by $a_{ij} = w_j\, K(s_i,t_j)$ and $b_i = g(s_i)$ for $i,j = 1,\ldots,n$. If the solution f is known then we represent it by $\mathbf{x}$ with elements $x_j = f(t_j)$, $j = 1,\ldots,n$. In the *Galerkin method*, we choose the following orthonormal box functions as basis functions:

$$\psi_i(s) = \begin{cases} h_s^{-\frac{1}{2}} & , \quad s \in [s_{i-1}, s_i] \\ 0 & , \quad \text{elsewhere} \end{cases} \quad , \quad \phi_i(t) = \begin{cases} h_t^{-\frac{1}{2}} & , \quad t \in [t_{i-1}, t_i] \\ 0 & , \quad \text{elsewhere} \end{cases}$$

in which $h_s = (d-c)/n$, $h_t = (b-a)/n$, and $s_i = i h_s$, $t_i = i h_t$, $i = 0,\ldots,n$. Then the Galerkin method [3] leads to a linear system of equations $A\,\mathbf{x} = \mathbf{b}$ with elements given by Eq. (2.4). Similarly, we represent the solution f by the vector $\mathbf{x}$ with elements $x_j = \int_a^b \phi_j(t)\, f(t)\, dt$, $j = 1,\ldots,n$. We stress that for both methods the product $A\,\mathbf{x}$ is, in general, *different* from $\mathbf{b}$. The table below gives an overview of the 12 test problems, while graphs of $\mathbf{x}$ for $n = 100$ are given in the individual manual pages.

test problem	discretization	$A\,\mathbf{x} = \mathbf{b}$
baart	Galerkin	no
blur	–	yes
deriv2	Galerkin	yes
foxgood	quadrature	no
heat	quadrature	yes
ilaplace	quadrature	yes
parallax	Galerkin	$\mathbf{x}$ not known
phillips	Galerkin	no
shaw	quadrature	yes
spikes	"cooked up"	yes
ursell	Galerkin	no $\mathbf{x}$ exists
wing	Galerkin	no

app_hh

Purpose:

Apply a Householder transformation.

Synopsis:

A = app_hh (A,beta,v)

Description

app_hh applies the Householder transformation, defined by the vector v and the scaler beta, to the matrix A; i.e.,

$$\mathsf{A} \leftarrow (I_n - \mathsf{beta}\,\mathsf{v}\,\mathsf{v}^T)\,\mathsf{A}\ .$$

See also:

gen_hh

baart

Purpose:

Test problem: Fredholm integral equation of the first kind.

Synopsis:

[A,b,x] = baart (n)

Description:

Discretization of an artificial Fredholm integral equation of the first kind (2.1) with kernel K and right-hand side g given by

$$K(s,t) = \exp(s \cos t) \ , \qquad g(s) = 2\,\frac{\sin s}{s} \ ,$$

and with integration intervals $s \in [0, \frac{\pi}{2}]$ and $t \in [0, \pi]$. The solution is given by

$$f(t) = \sin t \ .$$

The size of the matrix A is $\mathsf{n} \times \mathsf{n}$.

Examples:

Generate a "noisy" problem of size n = 32:

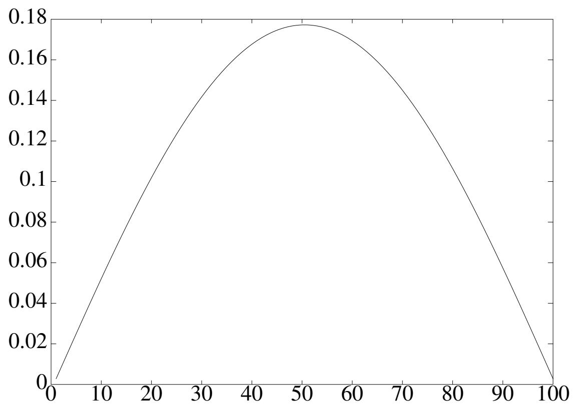
[A,b,x] = baart (32); b = b + 1e-3*randn (size(b));

Limitations:

The order n must be even.

References:

1. M. L. Baart, *The use of auto-correlation for pseudo-rank determination in noisy ill-conditioned linear least-squares problems*, IMA J. Numer. Anal. **2** (1982), 241–247.



bidiag

Purpose:

Bidiagonalization of an $m \times n$ matrix with $m \geq n$.

Synopsis:

B = bidiag (A)

[U,B,V] = bidiag (A)

Description:

If A is an $m \times n$ matriz with $m \geq n$ then bidiag uses Householder transformations to compute a bidiagonalization of A:

$$\mathsf{A} = \mathsf{U}\,\mathsf{B}\,\mathsf{V}^T,$$

where $\mathsf{B} \in \mathbb{R}^{n \times n}$ is upper bidiagonal,

$$\mathsf{B} = \begin{pmatrix} b_{11} & b_{12} & & \\ & b_{22} & b_{23} & \\ & & \ddots & \ddots \\ & & & b_{nn} \end{pmatrix},$$

and the matrices $\mathsf{U} \in \mathbb{R}^{m \times n}$ and $\mathsf{V} \in \mathbb{R}^{n \times n}$ have orthonormal columns. The bidiagonal matrix B is stored as a sparse matrix.

Examples:

Compute the bidiagonalization of A and compare the singular value of A with those of B (they should be identical to about machine precision):

B = bidiag (A); [svd (A),csvd (B)]

Algorithm:

Alternating left and right Householder transformations are used to bidiagonalized A. If U and V are also required, then the Householder transformations are accumulated.

Limitations:

The case $m < n$ is not allowed.

See also:

bsvd, lanc_b

References:

1. L. Eldén, *Algorithms for regularization of ill-conditioned least-squares problems*, BIT **17** (1977), 134-145.

blur

Purpose:

Test problem: deblurring of images degraded by atmospheric turbulence blur.

Synopsis:

[A,b,x] = blur (N,band,sigma)

Description:

This image deblurring problem arises in connection with the degradation of digital images by atmospheric turbulence blur, modelled by a Gaussian point-spread function:

$$h(x,y) = \frac{1}{2\,\pi\,\mathsf{sigma}^2}\exp\left(-\frac{x^2+y^2}{2\,\mathsf{sigma}^2}\right).$$

The matrix A is a symmetric $\mathsf{N}^2 \times \mathsf{N}^2$ doubly Toeplitz matrix, stored in sparse format, and given by

$$\mathsf{A} = (2\,\pi\,\mathsf{sigma}^2)^{-1}\,T \otimes T,$$

where T is an $\mathsf{N} \times \mathsf{N}$ symmetric banded Toeplitz matrix whose first row is given by

$$\mathsf{z} = [\mathsf{exp}(-([0:\mathsf{band}-1].\hat{\ }2)/(2*\mathsf{sigma}\hat{\ }2)); \mathsf{zeros}(1,\mathsf{N}-\mathsf{band})].$$

Only elements within a distance $\mathsf{band} - 1$ from the diagonal are stored; i.e., band is the half-bandwidth of the matrix T. If band is not specified, then $\mathsf{band} = 3$ is used.

The parameter sigma controls the shape of the Gaussian point spread function and thus the amount of smoothing (the larger the sigma, the wider the function, and the less ill posed the problem). If sigma is not specified, then $\mathsf{sigma} = 0.7$ is used.

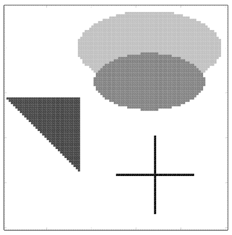
The vector x is a columnwise stacked version of a simple test image, while b holds a columnwise stacked version of the blurrred image; i.e, b = A*x.

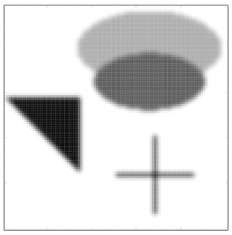
Limitations:

The integer N should not be too small; we recommend $\mathsf{N} \geq 16$.

Reference:

1. M. Hanke & P. C. Hansen, *Regularization methods for large-scale problems*, Surv. Math. Ind. **3** (1993), 253–315.





cgls

Purpose:

Conjugate gradient algorithm applied implicitly to the normal equations.

Synopsis:

[X,rho,eta,F] = cgls (A,b,k,reorth,s)

Description:

Performs k steps of the conjugate gradient algorithm applied implicitly to the normal equations $A^T A\,x = A^T b$.

The routine returns all k solutions, stored as columns of the matrix X. The corresponding solution norms and residual norms are returned in eta and rho, respectively.

If the singular values s are also provided, cgls computes the filter factors associated with each step and stores them columnwise in the matrix F.

Reorthogonalization of the normal equation residual vectors $A^T(A\,X(:,i) - b)$, $i = 1,\ldots,k$ is controlled by means of reorth as follows:

reorth = 0 : no reorthogonalization
reorth = 1 : reorthogonalization by means of MGS.

No reorthogonalization is assumed if reorth is not specified.

A "preconditioned" version of cgls for the general-form problem, where one minimizes $\|L\,x\|_2$ instead of $\|x\|_2$ in each step, is implemented in routine pcgls.

Examples:

Perform 25 iterations and plot the corresponding L-curve:

[X,rho,eta] = cgls (A,b,25); plot_lc (rho,eta,'o');

Algorithm:

The algorithm, which avoids explicit formation of the normal-equation matrix $A^T A$, is described in [1]. The computation of the filter factors, using the recurrence relations for the solution and the residual vector, is described in [2].

See also:

lsqr, pcgls, plsqr

References:

1. Å. Björck, *Numerical Methods for Least Squares cProblems*, SIAM, Philadelphia, 1996.
2. C. R. Vogel, *Solving ill-conditioned linear systems using the conjugate gradient method*, Report, Dept. of Mathematical Sciences, Montana State University, 1987.

cgsvd

Purpose:

Compute the compact generalized SVD of a matrix pair $\mathsf{A} \in \mathbb{R}^{m\times n}$ and $\mathsf{L} \in \mathbb{R}^{p\times n}$ with $m \geq n \geq p$.

Synopsis:

sm = cgsvd (A,L)

[U,sm,X,V] = cgsvd (A,L) ,   sm = [sigma,mu]

Description:

Computes the generalized singular value decomposition of the matrix pair (A, L):

$$\mathsf{A} = \mathsf{U} \begin{pmatrix} \mathrm{diag}(\mathsf{sigma}) & 0 \\ 0 & I_{n-p} \end{pmatrix} \mathsf{X}^{-1} , \qquad \mathsf{L} = \mathsf{V} \, (\mathrm{diag}(\mathsf{mu}) \;\, 0) \, \mathsf{X}^{-1} ,$$

where the matrices $\mathsf{U} \in \mathbb{R}^{m\times n}$ and $\mathsf{V} \in \mathbb{R}^{p\times p}$ have orthonormal columns, and the matrix $\mathsf{X} \in \mathbb{R}^{n\times n}$ is nonsingular. Moreover, the vectors sigma and mu have length p, and sigma./mu are the generalized singular values of (A, L).

Notice that the c, s-pairs sigma and mu are returned in a $p \times 2$ array sm. The matrix V is not required by any of the routines in this package.

Algorithm:

Calls the Matlab routine gsvd and stores the c, s-pairs in a $p\times 2$ array. The gsvd routine is included in Matlab Version 5.2, and it is based on [1].

Limitations:

The dimensions *must* satisfy $m \geq n \geq p$, which is no restriction in connection with regularization problems.

See also:

csvd

References:

1. C. F. Van Loan, *Computing the CS and the generalized singular value decomposition*, Numer. Math. **46** (1985), 479–491.

csvd

Purpose:

Compute the compact singular value decomposition.

Synopsis:

s = csvd (A)

[U,s,V] = csvd (A)

[U,s,V] = csvd (A,'full')

Description:

Computes the compact form of the singular value decomposition:

$$\mathsf{A} = \mathsf{U}\,\mathrm{diag}(\mathsf{s})\,\mathsf{V}^T \ .$$

If the matrix A is $m \times n$, then the dimensions of the computed quantities are:

s	$\min(m,n) \times 1$
U	$m \times \min(m,n)$
V	$n \times \min(m,n)$

If a second argument is present, then the full U and V are returned.

Algorithm:

Calls the Matlab routine svd and stores the singular values in a column vector s.

See also:

cgsvd

deriv2

Purpose:

Test problem: computation of the second derivative.

Synopsis:

[A,b,x] = deriv2 (n,case)

Description:

This is a mildly ill-posed problem; i.e., its singular values decay slowly to zero. It is a discretization of a first kind Fredholm integral equation (2.1) whose kernel K is the Green's function for the second derivative:

$$K(s,t) = \begin{cases} s(t-1) & , \quad s < t \\ t(s-1) & , \quad s \geq t \end{cases} .$$

Both integration intervals are $[0,1]$, and as right-hand side g and corresponding solution f one can choose between the following:

case = 1 $\quad g(s) = (s^3 - s)/6 \ , \quad f(t) = t$

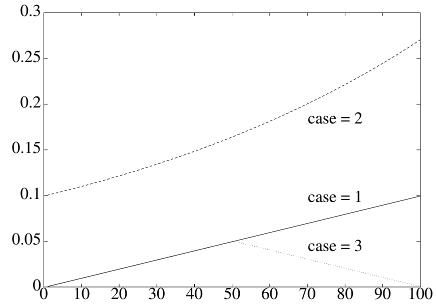
case = 2 $\quad g(s) = \exp(s) + (1-e)s - 1 \ , \quad f(t) = \exp(t)$

case = 3 $\quad g(s) = \begin{cases} (4s^3 - 3s)/24 & , \quad s < \frac{1}{2} \\ (-4s^3 + 12s^2 - 9s + 1)/24 & , \quad s \geq \frac{1}{2} \end{cases} , \quad f(t) = \begin{cases} t & , \quad t < \frac{1}{2} \\ 1-t & , \quad t \geq \frac{1}{2} \end{cases}$

The first two examples are from [1, p. 315] while the third example is from [2]. The size of the matrix A is n $\times$ n.

References:

1. L. M. Delves & J. L. Mohamed, *Computational Methods for Integral Equations*, Cambridge University Press, Cambridge, 1985.
2. A. K. Louis & P. Maass, *A mollifier method for linear operator equations of the first kind*, Inverse Problems **6** (1990), 427–440.



discrep

Purpose:

Disrepancy principle criterion for choosing the regularization parameter.

Synopsis:

[x_delta,lambda] = discrep (U,s,V,b,delta,x_0)

[x_delta,lambda] = discrep (U,sm,X,b,delta,x_0) ,   sm = [sigma,mu]

Description:

Least squares minimization with a quadratic inequality constraint:

$$\min \|\mathbf{x} - \mathsf{x\_0}\|_2 \quad \text{subject to} \quad \|A\,\mathbf{x} - \mathsf{b}\|_2 \le \mathsf{delta}$$
$$\min \|L\,(\mathbf{x} - \mathsf{x\_0})\|_2 \quad \text{subject to} \quad \|A\,\mathbf{x} - \mathsf{b}\|_2 \le \mathsf{delta}$$

where x_0 is an initial guess of the solution, and delta is a positive constant. The routine discrep requires either the compact SVD of A stored as U, s, and V, or part of the GSVD of (A, L) saved as U, sm, and X. The regularization parameter lambda corresponding to the solution x_delta is also returned.

If delta is a vector, then x_delta is a matrix such that

$$\mathsf{x\_delta} = [\,\mathsf{x\_delta}(1),\, \mathsf{x\_delta}(2),\, \ldots\,]\ .$$

If x_0 is not specified, x_0 = 0 is used.

The "opposite" problem, namely that of minimizing the residual norm subject to an upper bound on the solution (semi)norm, is treated by routine lsqi.

Examples:

Use discrep to solve the test problem shaw:

```
[A,b,x] = shaw (n); [U,s,V] = csvd (A); e = 1e-3*randn (size(b)); b = b + e;
x_lambda = discrep (U,s,V,b,norm (e)); plot ([x,x_lambda]);
```

Algorithm:

The algorithm, which uses a Newton method for computing the regularization parameter lambda (implemented in routine newton), is described in [1]. The starting value of lambda is set equal to that singular value s_k of A, or that generalized singular value γ_k of (A, L), for which the corresponding TSVD/TGSVD residual norm $\|A\,\mathbf{x}_k - \mathsf{b}\|_2$ is closest to delta.

See also:

lsqi, l_curve, newton, quasiopt

References:

1. V. A Morozov, *Methods for Solving Incorrectly Posed Problems*, Springer, New York, 1984; Chapter 26.

dsvd

Purpose:

Compute the damped SVD solution.

Synopsis:

[x_lambda,rho,eta] = dsvd (U,s,V,b,lambda)

[x_lambda,rho,eta] = dsvd (U,sm,X,b,lambda) ,   sm = [sigma,mu]

Description:

Computes the damped SVD solution, defined as

$$\begin{aligned} \mathsf{x\_lambda} &= \mathsf{V}\,(\mathrm{diag}(\mathsf{s}+\mathsf{lambda}))^{-1}\mathsf{U}^T\mathsf{b} && \text{(standard form)} \\ \mathsf{x\_lambda} &= \mathsf{X}\begin{pmatrix} \mathrm{diag}(\mathsf{sigma}+\mathsf{lambda}*\mathsf{mu}) & 0 \\ 0 & I_{n-p} \end{pmatrix}^{-1}\mathsf{U}^T\mathsf{b} && \text{(general form)} \end{aligned}$$

If lambda is a vector, then x_lambda is a matrix such that

$$\mathsf{x\_lambda} = [\,\mathsf{x\_lambda}(1),\, \mathsf{x\_lambda}(2), \ldots]\ .$$

The solution and residual norms are returned in eta and rho.

Algorithm:

Straightforward use of the definitions are used to compute x_lambda.

References:

1. M. P. Ekstrom & R. L Rhodes, *On the application of eigenvector expansions to numerical deconvolution*, J. Comp. Phys. **14** (1974), 319–340.

fil_fac

Purpose:

Compute the filter factors for some regularization methods.

Synopsis:

f = fil_fac (s,reg_param,method)

f = fil_fac (sm,reg_param,method) , sm = [sigma,mu]

f = fil_fac (s,reg_param,'ttls',s1,V1)

Description:

Computes all the filter factors corresponding to the singular values in s and the regularization parameter reg_param, for the following methods:

method	=	'dsvd'	:	damped SVD or GSVD
method	=	'tsvd'	:	truncated SVD or GSVD
method	=	'Tikh'	:	Tikhonov regularizaton
method	=	'ttls'	:	truncated TLS

If sm = [sigma,mu] is specified, then the filter factors for the corresponding generalized methods are computed. If method is not specified, 'Tikh' is default.

If method = 'ttls' then the singular values s1 and the right singular matrix V1 of $(A\ \mathbf{b})$ must also be supplied.

Examples:

Compute the filter factors for Tikhonov regularization corresponding to the regularization parameter lambda $= 10^{-3}$:

f = fil_fac (s,1e-3);

Algorithm:

The filter factors are computed by means of the definitions from Section 2.7. See also [1] for more details.

Limitations:

The routine fil_fac cannot be used to compute filter factors for the iterative methods; these filter factors must be computed by the respective routines. Neither can fil_fac be used to compute filter factors for MTSVD or maximum entropy regularization. For truncated TLS, the small filter factors are not computed accurately.

References:

1. P. C. Hansen, *Rank-Deficient and Discrete Ill-Posed Problems. Numerical Aspects of Linear Inversion*, SIAM, Philadelphia, 1997.

foxgood

Purpose:

Severely ill-posed test problem.

Synopsis:

[A,b,x] = foxgood (n)

Description:

Discretization of a Fredholm integral equation of the first kind (2.1) with both integration intervals equal to $[0,1]$, with kernel K and right-hand side g given by

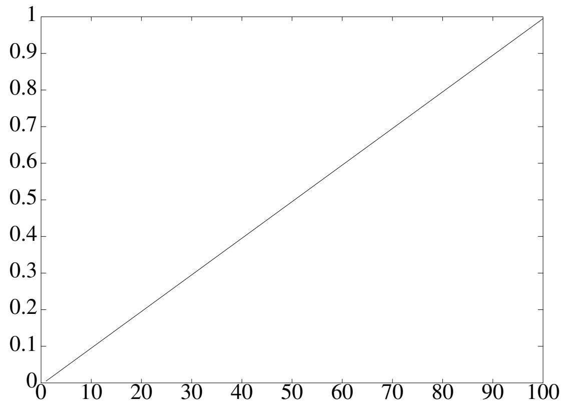
$$K(s,t) = (s^2+t^2)^{\frac{1}{2}} \ , \qquad g(s) = \frac{1}{3}\left((1+s^2)^{\frac{3}{2}} - s^3\right) \ ,$$

and with the solution $f = t$. The problem was first used by Fox & Goodwin.

This is an artificial discrete severely ill-posed problem which does not satisfy the discrete Picard condition for the smaller singular values. The matrix A is $\mathsf{n} \times \mathsf{n}$.

References:

1. C. T. H. Baker, *The Numerical Treatment of Integral Equations*, Clarendon Press, Oxford, 1977; p. 665.



gcv

Purpose:

Plot the GCV function and find its minimum.

Synopsis:

[reg_min,G,reg_param] = gcv (U,s,b,method)

[reg_min,G,reg_param] = gcv (U,sm,b,method) , sm = [sigma,mu]

Description:

Plots the generalized cross-validation (GCV) function

$$\mathsf{G} = \frac{\|A\,\mathbf{x} - \mathbf{b}\|_2^2}{(\mathrm{trace}\,(I_m - A\,A^I))^2}$$

as a function of the regularization parameter reg_param, depending on the method chosen. Here, A^I is a matrix which produces the regularized solution $\mathbf{x}$ when multiplied with the right-hand side $\mathbf{b}$, i.e., $\mathbf{x} = A^I\mathbf{b}$. The following methods are allowed:

method	=	'dsvd'	:	damped SVD or GSVD
method	=	'tsvd'	:	truncated SVD or GSVD
method	=	'Tikh'	:	Tikhonov regularizaton

If sm = [sigma,mu] is specified, then the filter factors for the corresponding generalized methods are computed. If method is not specified, 'Tikh' is default.

If any output arguments are specified, then the minimum of the GCV function is identified and the corresponding regularization parameter reg_min is returned.

Examples:

Generate a test problem and use gcv to compute the optimal regularization parameter lambda for Tikhonov regularization in standard form:

```
[A,b,x] = phillips (n); b = b + 1e-3*randn (size(b)); [U,s,V] = csvd (A);
lambda = gcv (U,s,b); x_lambda = tikhonov (U,s,V,b,lambda);
plot (1:n,x,1:n,x_lambda,'o')
```

Algorithm:

For Tikhonov regularization and damped SVD/GSVD, 200 logarithmically distributed regularization parameters are generated, and G is plotted for these values. Then the minimizer of the GCV function is computed via fmin, using the minimizer of G as initial guess. For truncated SVD/GSVD/TLS, G is computed and plotted for all valid values of the discrete regularization parameter.

Limitations:

gcv cannot be used to compute the GCV function for the iterative regularization methods. Use instead the filter factors and Eq. (2.63). If lsqr is used with reorthogonalization, then G can be approximated by (rho./(m − [1 : k]')).^2, where k is the number of LSQR steps.

Diagnostics:

The number of points on the GCV curve for Tikhonov regularization and damped SVD/GSVD is determined by the parameter npoints which can easily be changed by the user to give a finer resolution.

See also:

discrep, l_curve, quasiopt

References:

1. G. Wahba, *Spline Models for Observational Data*, CBMS-NSF Regional Conference Series in Applied Mathematics, Vol. 59, SIAM, Philadelphia, 1990.

gen_form

Purpose:

Transformation of a standard-form solution back to the general-form setting.

Synopsis:

x = gen_form (L_p,x_s,A,b,K,M) (method 1)
x = gen_form (L_p,x_s,x_0) (method 2)

Description:

Transforms the standard-form solution x_s back to the required solution x in the general-form setting. Notice that x_s may have more than one column. Two methods are available, and the distinction between them is controlled by the number of input parameters to gen_form.

In method 1, described in [1], the transformation takes the form

$$\mathsf{x} = \mathsf{L\_p}\,\mathsf{x\_s} + \mathsf{K}\,\mathsf{M}\,(\mathsf{b} - \mathsf{A}\,\mathsf{L\_p}\,\mathsf{x\_s})\ ,$$

where L_p is the pseudoinverse of the matrix L, the matrix K has columns in the null space of L, and $\mathsf{M} = T_o^{-1} H_o^T$, cf. (2.25). In method 2, described in [2,3,4], the transformation takes the form

$$\mathsf{x\_s} = \mathsf{L\_p}\,\mathsf{x\_s} + \mathsf{x\_0}\ ,$$

where now L_p is the A-weighted generalized inverse of L, and x_0 is the component of x in the null space of L (notice that this component is independent of x_s); see (2.32).

Usually, the transformation from general form into standard form is performed by means of routine std_form.

Notice that both gen_form and std_form are available for pedagogical reasons only—usually it is more efficient to build the transformations into the solution process, such as in the iterative methods pcgls, plsqr, and pnu.

Examples:

Transform a general-form problem into standard form, produce 10 TSVD solutions, and transform back again, using method 1; then compare with the mathematically identical TGSVD solutions:

```
[A_s,b_s,L_p,K,M] = std_form (A,L,b); [U,s,V] = csvd (A_s);
X_s = tsvd (U,s,V,b_s,1:10); X = gen_form (L_p,X_s,A,b,K,M);
[U1,V1,sm,X1] = gsvd (A,L); XX = tgsvd (U1,sm,X1,b,1:10); norm (X−XX)
```

Algorithm:

The algorithms are described in details in refs. [1,2,3,4].

See also:

std_form

References:

1. L. Eldén, *Algorithms for regularization of ill-conditioned least-squares problems*, BIT **17** (1977), 134–145.
2. L. Eldén, *A weighted pseudoinverse, generalized singular values, and constrained least squares problems*, BIT **22** (1982), 487–502.
3. M. Hanke, *Regularization with differential operators. An iterative approach*, J. Numer. Funct. Anal. Optim. **13** (1992), 523–540.
4. P. C. Hansen, *Rank-Deficient and Discrete Ill-Posed Problems. Numerical Aspects of Linear Inversion,* SIAM, Philadelphia, 1997.

gen_hh

Purpose:

Generate a Householder transformation.

Synopsis:

[x1,beta,v] = gen_hh (x)

Description:

Given a vector x, gen_hh computes the scalar beta and the vector v determining a Householder transformation

$$H = I_n - \mathsf{beta}\,\mathsf{v}\,\mathsf{v}^T$$

such that $H\,\mathsf{x} = \pm\,\|\mathsf{x}\|_2\,\mathbf{e}_1$, where $\mathbf{e}_1$ is the first unit vector. The quantity x1 returned by gen_hh is the first element of $H\,\mathsf{x}$.

Examples:

The very first step of the bidiagonalization of a matrix A looks as follows:

[A(1,1),beta,v] = gen_hh (A(1:m,1)); A(2:m,1) = zeros (m-1,1);

A(1:m,2:n) = app_hh (A(1:m,2:n),beta,v);

See also:

app_hh

get_l

Purpose:

Compute discrete derivative operators.

Synopsis:

[L,W] = get_l (n,d)

Description:

Computes the discrete approximation L to the derivative operator of order d on a regular grid with n points, i.e. the matrix L is $(n - d) \times n$. In particular, for $d = 1$ and $d = 2$, L has the form

$$L = \begin{pmatrix} 1 & -1 & & & & \\ & 1 & -1 & & & \\ & & \ddots & \ddots & & \\ & & & & 1 & -1 \end{pmatrix} \quad \text{and} \quad L = \begin{pmatrix} 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -2 & 1 \end{pmatrix},$$

respectively. The matrix L is stored as a sparse matrix.

If required, an orthonormal basis represented by the columns of the matrix W is also computed. The matrix W is used in the "preconditioned" iterative methods.

Algorithm:

The matrix W is computed by first generating the "trivial" basis vectors $(1\ 1 \ldots 1)^T$, $(1\ 2 \ldots n)^T$, etc., and then orthonormalizing these by means of MGS.

heat

Purpose:

Test problem: inverse heat equation.

Synopsis:

[A,b,x] = heat (n,kappa)

Description:

The inverse heat equation used here is a Volterra integral equation of the first kind with $[0,1]$ as integration interval. The kernel is $K(s,t) = k(s-t)$ with

$$k(t) = \frac{t^{-3/2}}{2\,\mathsf{kappa}\sqrt{\pi}} \exp\left(-\frac{1}{4\,\mathsf{kappa}^2\,t^2}\right).$$

Here, the parameter kappa controls the ill-conditioning of the matrix A:

kappa = 5 gives a well-conditioned matrix,
kappa = 1 gives an ill-conditioned matrix.

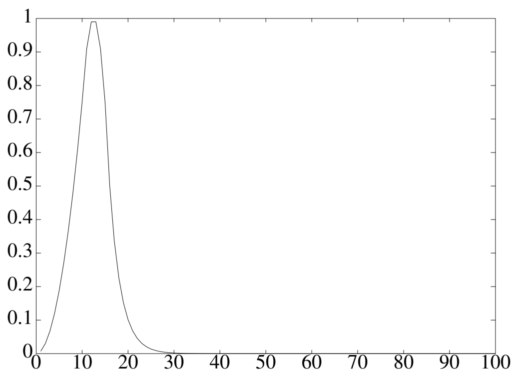
The default is kappa = 1.

Algorithm:

The integral equation is discretized by means of simple collocation and the midpoint rule with n points, cf. [1,2]. An exact solution x is constructed, and then the right-hand side b is produced as b = A x.

References:

1. A. S. Carasso, *Determining surface temperatures from interior observations*, SIAM J. Appl. Math. **42** (1982), 558–574.
2. L. Eldén, *The numerical solution of a non-characteristic Cauchy problem for a parabolic equation*; in P. Deuflhart & E. Hairer (Eds.), *Numerical Treatment of Inverse Problems in Differential and Integral Equations*, Birkhäuser, Boston, 1983.



heb_new

Purpose:

Newton iteration with Hebden model (utility routine for lsqi).

Synopsis:

lambda = heb_new (lambda_0,alpha,s,beta,omega)

Description:

heb_new uses Newton iteration with a Hebden (rational) model to find the solution to the secular equation

$$\|L\,(\mathbf{x}_\lambda - \mathbf{x}_0)\|_2 = \mathsf{alpha}\ ,$$

where $\mathbf{x}_\lambda$ is the solution defined by Tikhonov regularization.

The initial guess of λ is lambda_0. The norm $\|L\,(\mathbf{x}_\lambda - \mathbf{x}_0)\|_2$ is computed via the quantities s, beta, and omega. Here, s holds either the singular values of A, if $L = I_n$, or the c, s-pairs of the GSVD of (A, L) if $L \neq I_n$. Moreover, beta $= U^T\mathbf{b}$ and omega is either $V^T\mathbf{x}_0$ or the first p elements of $X^{-1}\mathbf{x}_0$.

Algorithm:

The original algorithm is described in [1], and the extension to the case $\mathbf{x}_0 \neq \mathbf{0}$ is by P. C. Hansen. The Hebden model, to which Newton's method is applied, and which was found experimentally to be superior in connection with lsqi, is $\|L\,(\mathbf{x}_\lambda - \mathbf{x}_0)\|_2^{-2} - \mathsf{alpha}^{-2}$.

Diagnostics:

The performance of heb_new relies to some extent on a fairly good starting guess lambda_0 of λ. A maximum number of 50 iterations is default.

See also:

lsqi

References:

1. T. F. Chan, J. Olkin & D. W. Cooley, *Solving quadratically constrained least squares using black box unconstrained solvers*, BIT **32** (1992), 481–495.

ilaplace

Purpose:

Test problem: inverse Laplace transformation.

Synopsis:

[A,b,x] = ilaplace (n,example)

Description:

Discretization of the inverse Laplace transformation (a Fredholm integral equation of the first kind (2.1)) by means of Gauss-Laguerre quadrature. The kernel K is given by

$$K(s,t) = \exp(-s\,t),$$

and both integration intervals are $[0,\infty)$. The following examples are implemented, where f denotes the solution and g denotes the corresponding right-hand side:

example = 1: $f(t) = \exp(-t/2)$, $\quad g(s) = \frac{1}{s+1/2}$

example = 2: $f(t) = 1 - \exp(-t/2)$, $\quad g(s) = \frac{1}{s} - \frac{1}{s+1/2}$

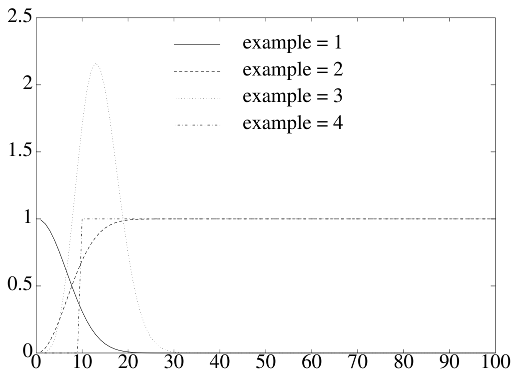
example = 3: $f(t) = t^2 \exp(-t/2)$, $\quad g(s) = \frac{2}{(s+1/2)^3}$

example = 4: $f(t) = \begin{cases} 0, & t \le 2 \\ 1, & t > 2 \end{cases}$, $\quad g(s) = \frac{\exp(-2s)}{s}$

All four examples are from [1]. The size of the matrix A is n $\times$ n.

References:

1. J. M. Varah, *Pitfalls in the numerical solution of linear ill-posed problems*, SIAM J. Sci. Stat. Comput. **4** (1983), 164–176.



lagrange

Purpose:

Plot the Lagrange function for Tikhonov regularization.

Synopsis:

[La,dLa,lambda0] = lagrange (U,s,b,more)

[La,dLa,lambda0] = lagrange (U,sm,b,more) , sm = [sigma,mu]

Description:

Plots the Lagrange function for Tikhonov regularization,

$$\mathsf{La} = \|A\,\mathbf{x}_\lambda - \mathsf{b}\|_2^2 + \lambda^2\,\|L\,\mathbf{x}_\lambda\|_2^2 \ ,$$

and its first derivative dLa $= d\mathsf{La}/d\lambda$, versus λ. Here, $\mathbf{x}_\lambda$ is the Tikhonov regularized solution. If nargin = 4, then the norms $\|A\,\mathbf{x}_\lambda - \mathsf{b}\|_2$ and $\|L\,\mathbf{x}_\lambda\|_2$ are also plotted. The routine returns La, dLa, and an approximate value lambda0 of λ for which dLa has its minimum.

See also:

l_curve, tikhonov

lanc_b

Purpose:

Lanczos bidiagonalization.

Synopsis:

B_k = lanc_b (A,p,k,reorth)

[U,B_k,V] = lanc_b (A,p,k,reorth)

Description:

Performs k steps of the Lanczos bidiagonalization process with starting vector p, producing a lower bidiagonal $(k+1) \times k$ matrix B_k,

$$\mathsf{B\_k} = \begin{pmatrix} b_{11} & & & \\ b_{21} & b_{22} & & \\ & b_{32} & \ddots & \\ & & \ddots & b_{kk} \\ & & & b_{k+1,k} \end{pmatrix}, \qquad \text{such that} \qquad \mathsf{A\,V} = \mathsf{U\,B\_k}\ .$$

The matrix B_k is stored as a sparse matrix. The matrices $\mathsf{U} \in \mathbb{R}^{m\times(k+1)}$ and $\mathsf{V} \in \mathbb{R}^{n\times k}$ consist of the left and right Lanczos vectors (which are orthonormal in exact arithmetic). Reorthogonalization is controlled by means of reorth as follows:

reorth = 0 : no reorthogonalization
reorth = 1 : reorthogonalization by means of MGS
reorth = 2 : Householder reorthogonalization.

No reorthogonalization is assumed if reorth is not specified.

Examples:

Perform 10 steps of Lanczos bidiagonalization without reorthogonalization, then compute the singular values of the 11×10 bidiagonal matrix and compare with the 10 largest singular values of the coefficient matrix A:

```
B_k = lanc_b (A,b,10); s_k = csvd (B_k); s = svd (A); [s_k,s(1:10)]
```

Algorithm:

The algorithm is a straight-forward implementation of the Lanczos bidiagonalization process as described in, e.g., [1].

See also:

bidiag, bsvd, lsqr, plsqr

References:

1. G. H. Golub & C. F. Van Loan, *Matrix Computations*, 2. Ed., Johns Hopkins, Baltimore, 1989; Section 9.3.4.

lsolve

Purpose:

Utility routine for "preconditioned" iterative methods.

Synopsis:

x = lsolve (L,y,W,T)

Description:

Computes the vector

$$\mathsf{x} = \mathsf{L}_A^{\dagger}\,\mathsf{y}\ ,$$

where $\mathsf{L}_A^{\dagger}$ is the A-weighted generalized inverse of L. Here, L is a $p \times n$ matrix, W holds a basis for the null space of L, and T is a utility matrix which should be computed by routine pinit. Alternatively, L is square and dense, and W and T are not needed.

Notice that x and y may be matrices, in which case

$$\mathsf{x}(:,i) = \mathsf{L}_A^{\dagger}\,\mathsf{y}(:,i)\ .$$

Algorithm:

The vector x is computed by means of the following algorithm from [2, §2.3.2], based on [1]:

$$\begin{aligned} \hat{\mathsf{x}} &\leftarrow \mathsf{L}(:,1\!:\!p)^{-1}\mathsf{y} \\ \mathsf{x} &\leftarrow \begin{pmatrix} \hat{\mathsf{x}} \\ 0 \end{pmatrix} - \mathsf{W}\,\mathsf{T}(:,1\!:\!\mathsf{p})\,\hat{\mathsf{x}}\ . \end{aligned}$$

See also:

ltsolve, pcgls, pinit, plsqr, pnu

References:

1. M. Hanke, *Regularization with differential operators. An iterative approach*, J. Numer. Funct. Anal. Optim. **13** (1992), 523–540.
2. P. C. Hansen, *Rank-Deficient and Discrete Ill-Posed Problems. Numerical Aspects of Linear Inversion*, SIAM, Philadelphia, 1997.

lsqi

Purpose:

Least squares minimization with a quadratic inequality constraint.

Synopsis:

[x_alpha,lambda] = lsqi (U,s,V,b,alpha,x_0)

[x_alpha,lambda] = lsqi (U,sm,X,b,alpha,x_0) , sm = [sigma,mu]

Description:

Solves the following least squares minimization problems with a quadratic inequality constraint:

$$\min \|A\,\mathbf{x} - \mathsf{b}\|_2 \quad \text{subject to} \quad \|\mathbf{x} - \mathsf{x\_0}\|_2 \leq \mathsf{alpha}$$
$$\min \|A\,\mathbf{x} - \mathsf{b}\|_2 \quad \text{subject to} \quad \|L\,(\mathbf{x} - \mathsf{x\_0})\|_2 \leq \mathsf{alpha}$$

where x_0 is an initial guess of the solution, and alpha is a positive constant. The routine requires either the compact SVD of A saved as U, s, and V, or part of the GSVD of (A, L) saved as U, sm, and X. The regularization parameter lambda corresponding to the solution x_alpha is also returned.

If alpha is a vector, then x_alpha is a matrix such that

$$\mathsf{x\_alpha} = [\mathsf{x\_alpha}(1),\, \mathsf{x\_alpha}(2),\, \ldots]\ .$$

If x_0 is not specified, x_0 = $\mathbf{0}$ is used.

The "opposite" problem, namely that of minimizing the solution (semi)norm subject to an upper bound on the residual norm, is treated by routine discrep.

Examples:

Generate a "noisy" test problem with a know exact solution x. Then compute two regularized solutions x1 and x2 with $\|\mathsf{x1}\|_2 = 1.1\,\|\mathsf{x}\|_2$ and $\|L\,\mathsf{x1}\|_2 = 1.1\,\|L\,\mathsf{x}\|_2$, where L is an approximation to the second derivative operator:

```
[A,b,x] = shaw (32); b = b + 1e-3*randn (size(b)); [U,s,V] = csvd (A);
L = get_l (32,2); [UU,sm,XX] = cgsvd (A,L);
x1 = lsqi (U,s,V,b,1.05*norm(x));
x2 = lsqi (UU,sm,XX,b,1.05*norm(L*x));
plot ([x,x1,x2])
```

Algorithm:

The algorithm uses Newton iteration with a Hebden rational model (implemented as routine heb_new) to solve the secular equation $\|L\,(\mathbf{x}_\lambda - \mathsf{x\_0})\|_2 = \mathsf{alpha}$, cf. [1]. The initial guess of λ is computed as described in [1]. Although it is derived for the case x_0 = $\mathbf{0}$, the idea is also applicable for x_0 $\neq \mathbf{0}$ because the initial λ is almost unaffected by x_0 (since $\|\mathbf{x}_0\|_2 \gg \|\mathsf{x\_0}\|_2$, where $\mathbf{x}_0$ is the unregularized solution).

Diagnostics:

In rare cases the iteration in heb_new converges very slowly; then try to increase the maximum number of allowed iterations in heb_new or a slightly different alpha.

See also:

hebnew, discrep

References:

1. T. F. Chan, J. Olkin & D. W. Cooley, *Solving quadratically constrained least squares using black box unconstrained solvers*, BIT **32** (1992), 481–495.

lsqr

Purpose:

Solution of least squares problems by the LSQR algorithm based on Lanczos bidiagonalization.

Synopsis:

[X,rho,eta,F] = lsqr (A,b,k,reorth,s)

Description:

Performs k steps of the LSQR Lanczos bidiagonalization algorithm (due to Paige & Saunders) applied to the system

$$\min \|\mathsf{A}\,\mathbf{x} - \mathsf{b}\|_2 \ .$$

The routine returns all k solutions, stored as columns of the matrix X. The solution norms and residual norms are returned in eta and rho, respectively.

Reorthogonalization of the Lanczos vectors is controlled by means of reorth as follows:

reorth = 0 : no reorthogonalization
reorth = 1 : reorthogonalization by means of MGS
reorth = 2 : Householder reorthogonalization.

No reorthogonalization is assumed if reorth is not specified.

If the singular values s of A are also provided, then lsqr computes the filter factors associated with each step and stores them columnwise in the array F.

A "preconditioned" version of LSQR for the general-form problem where one minimizes $\|L\,\mathbf{x}\|_2$ instead of $\|\mathbf{x}\|_2$ in each step is implemented in routine plsqr.

Examples:

Perform 25 LSQR iterations without reorthogonalization, and plot the corresponding L-curve:

[X,rho,eta] = lsqr (A,b,25); plot_lc (rho,eta,'o');

Algorithm:

lsqr is a straightforward implementation of the algorithm described in [1]. The original algorithm also includes the possibility for adding Tikhonov regularization with a fixed regularization parameter to the least squares problem, but for simplicity this feature is not included in Regularization Tools.

See also:

bidiag, lanc_b, plot_lc, plsqr

References:

1. C. C. Paige & M. A. Saunders, *LSQR: an algorithm for sparse linear equations and sparse least squares*, ACM Trans. Math. Software **8** (1982), 43–71.

ltsolve

Purpose:

Utility routine for "preconditioned" iterative methods.

Synopsis:

x = ltsolve (L,y,W,T)

Description:

Computes the vector

$$\mathsf{x} = (\mathsf{L}_A^\dagger)^T \mathsf{y} ,$$

where $\mathsf{L}_A^\dagger$ is the A-weighted generalized inverse of L. Here, L is a $p \times n$ matrix, W holds a basis for the null space of L, and T is a utility matrix which should be computed by routine pinit. Alternatively, L is square and dense, and W and T are not needed.

If W and T are not specified, then instead the vector

$$\mathsf{x} = (\mathsf{L}(:,1\colon p)^{-1})^T \mathsf{y}(1\colon p)$$

is computed.

Notice that x and y may be matrices, in which case

$$\mathsf{x}(:,i) = (\mathsf{L}_A^\dagger)^T \mathsf{y}(:,i) \qquad \text{or} \qquad \mathsf{x}(:,i) = (\mathsf{L}(:,1\colon p)^{-1})^T \mathsf{y}(1\colon p, i) .$$

Algorithm:

The following algorithm from [2, §2 3 2] (based on [1]) is used. If W and T are specified, then y is first updated by means of the relation

$$\mathsf{y} \leftarrow \mathsf{y}(1\colon p) - \mathsf{T}(:,1\colon p)^T \mathsf{W}^T \mathsf{y} ,$$

otherwise the update $\mathsf{y} \leftarrow \mathsf{y}(1\colon p)$ is computed. The latter option is used in the "preconditioned" iterative methods. Then x is computed as $(\mathsf{L}(:,1\colon p)^{-1})^T \mathbf{y}$.

See also:

lsolve, pcgls, pinit, plsqr, pnu

References:

1. M. Hanke, *Regularization with differential operators. An iterative approach*, J. Numer. Funct. Anal. Optim. **13** (1992), 523–540.
2. P. C. Hansen, *Rank-Deficient and Discrete Ill-Posed Problems. Numerical Aspects of Linear Inversion*, SIAM, Philadelphia, 1997.

l_corner

Purpose:

Locate the "corner" of the L-curve.

Synopsis:

[reg_c,rho_c,eta_c] = l_corner (rho,eta,reg_param)

[reg_c,rho_c,eta_c] = l_corner (rho,eta,reg_param,U,s,b,method,M)

[reg_c,rho_c,eta_c] = l_corner (rho,eta,reg_param,U,sm,b,method,M) , sm = [sigma,mu]

Description:

l_corner locates the corner of the L-curve, given by rho and eta, in *log-log scale*. It is assumed that corresponding values of $\|A\,\mathbf{x}-\mathbf{b}\|_2$, $\|L\,\mathbf{x}\|_2$, and the regularization parameter are stored in the arrays rho, eta, and reg_param, respectively (such as the output from routine l_curve). If nargin = 3 then no particular method is assumed, and if nargin = 2 then it is assumed that reg_param = 1:length (rho).

If nargin $\geq$ 6, then the following regularization methods are allowed:

method = 'dsvd'	:	damped SVD or GSVD
method = 'mtsvd'	:	modified TSVD
method = 'Tikh'	:	Tikhonov regularization
method = 'tsvd'	:	truncated SVD or GSVD.

If no method is specified, 'Tikh' is default.

An eighth argument M specifies an upper bound for eta, below which the corner should be found.

Examples:

l_corner is invoked by l_curve if any output arguments are specified to the latter routine. However, l_corner can also be use as a stand-alone routine. In the following example, rho and eta come from the LSQR algorithm with 15 iterations—so the regularization parameters in reg_param are 1:15—and l_corner is used to compute the "corner" of the L-curve associated with rho and eta:

```
[A,b,x] = shaw (32); b = b + 1e-3*randn (size(b)); [X,rho,eta] = lsqr (A,b,15,1);
[reg_c,rho_c,eta_c] = l_corner (rho,eta,1:15);
```

Algorithm:

If method is Tikhonov regularization or DSVD/DGSVD, then the L-curve is differentiable, and it is straightforward to compute the curvature and find the point on the L-curve with maximum curvature, which is then defined as the "corner". For the remaining methods, the L-curve is discrete and the procedure is then to fit a 2D spline curve to the data points, compute the point on the spline curve with maximum curvature, and then define the "corner" of the discrete L-curve as the data point closest to the point on the spline curve with maximum curvature. For more details, see [2].

If the curvature of the L-curve is negative everywhere, then the leftmost point on the L-curve is taken as the "corner". This choice is the most natural in connection with the L-curve criterion for choosing the regularization parameter.

Diagnostics:

For discrete L-curves, the number of data points in rho and eta must be greater than four, the order of the fitting 2D spline curve.

The algorithm may sometimes mistake a fine-grained local "corner" for the desired corner. In this case, the user may wish to experiment with the default parameters deg (the degree of a local smoothing polynomial applied before fitting the 2D spline curve), q (the half-width of the local smoothing interval in which polynomial smoothing is applied), and order (the order of the fitting 2D spline curve).

The user may also wish to increase the parameter s_thr: for TSVD, TGSVD and MTSVD, values of rho and eta corresponding the singular values s below s_thr are not used in the search for the "corner".

If the Spline Toolbox is not available, and the routine is called with nargin ≥ 6 and method equal to 'tsvd' or 'mtsvd', then the routine returns with an error message.

See also:

l_curve

External routines required:

The following ten routines from the Spline Toolbox [1] are required by l_corner:

fnder, ppbrk, ppcut, ppmak, ppual, sp2pp, sorted, spbrk, spmak, sprpp.

References:

1. C. de Boor, *Spline Toolbox*, Version 1.1, The Mathworks, MA, 1992.
2. P. C. Hansen & D. P. O'Leary, *The use of the L-curve in the regularization of discrete ill-posed problems*, SIAM J. Sci. Comput. **14** (1993), 1487–1503.

l_curve

Purpose:

Plot the L-curve and find its "corner".

Synopsis:

[reg_corner,rho,eta,reg_param] = l_curve (U,s,b,method)

[reg_corner,rho,eta,reg_param] = l_curve (U,sm,b,method) , sm = [sigma,mu]

[reg_corner,rho,eta,reg_param] = l_curve (U,s,b,method,L,V)

Description:

Plots the L-shaped curve of eta, the solution norm $\|\mathbf{x}\|_2$ or seminorm $\|L\,\mathbf{x}\|_2$, versus rho, the residual norm $\|A\,\mathbf{x}-\mathbf{b}\|_2$, for the following methods:

method = 'Tikh'	:	Tikhonov regularization	(solid line)
method = 'tsvd'	:	truncated SVD or GSVD	(o markers)
method = 'dsvd'	:	damped SVD or GSVD	(dotted line)
method = 'mtsvd'	:	modified TSVD	(x markers)

The corresponding regularization parameters are returned in reg_param. If no method is specified, 'Tikh' is default. For other methods, use routine plot_lc.

If any output arguments are specified, then the corner of the L-curve is identified (by means of routine l_corner) and the corresponding regularization parameter reg_corner is returned as well as printed on the plot. This is the *L-curve criterion* for choosing the regularization parameter. Use l_corner if an upper bound on eta is required when finding the L-curve's "corner".

Examples:

Plot the L-curve for Tikhonov regularization and compute the regularization parameter reg_corner corresponding to the "corner":

[A,b,x] = shaw (32); b = b + 1e-3*randn (size(b)); [U,s,V] = csvd (A);

reg_corner = l_curve (U,s,b);

Algorithm:

The algorithm for computing the "corner" of the L-curve in log-log scale is described in [1], where the existence of the "corner" is also discussed.

Diagnostics:

See the documentation for l_corner for diagnostics. If the Spline Toolbox is not available and reg_corner is requested, then the routine returns reg_corner = NaN if method equals 'tsvd' or 'mtsvd'.

See also:

discrep, gcv, l_corner, plot_lc, quasiopt

References:

1. P. C. Hansen & D. P. O'Leary, *The use of the L-curve in the regularization of discrete ill-posed problems*, SIAM J. Sci. Comput. **14** (1993), 1487–1503.

maxent

Purpose:

Maximum entropy regularization.

Synopsis:

[x_lambda,rho,eta] = maxent (A,b,lambda,w,x0)

Description:

Computes the regularized maximum entropy solution x_lambda which minimizes

$$\|\mathsf{A}\,\mathbf{x} - \mathsf{b}\|_2^2 + \mathsf{lambda}^2 \sum_{i=1}^{n} x_i \, \log(w_i \, x_i) \ ,$$

where $-\sum_{i=1}^{n} x_i \, \log(w_i \, x_i)$ is the entropy of the vector $\mathbf{x}$, and $\mathsf{w} = (w_1, \ldots, w_n)^T$ is a vector of weights. If no weights are specified, unit weights are used. A starting vector x0 for the iterative process can be specified, otherwise a constant starting vector is used.

If lambda is a vector, then x_lambda is a matrix such that

$$\mathsf{x\_lambda} = [\, \mathsf{x\_lambda}(1), \, \mathsf{x\_lambda}(2), \, \ldots] \ .$$

Examples:

Compare maximum entropy regularization with Tikhonov regularization:

```
[A,b,x] = shaw (32); b = b + 1e-3*randn (size(b)); [U,s,V] = csvd (A);
lambda = logspace (-1,-4,12);
X_tikh = tikhonov (U,s,V,b,lambda);
X_ment = maxent (A,b,lambda);
for i=1:12
  dif(i,1) = norm(x−X_tikh(:,i));
  dif(i,2) = norm(x−X_ment(:,i));
  dif(i,3) = norm(X_tikh−X_ment(:,i));
end
loglog (lambda,dif/norm(x))
```

Algorithm:

Routine maxent uses a nonlinear conjugate gradient algorithm [1, §4.1] with inexact line search and a step-length control which ensures that all elements of the iteration vector are positive. For more details, see §2.7.5.

Diagnostics:

Slow convergence may occur for very small values of the regularization parameter lambda.

References:

1. R. Fletcher, *Practical Methods of Optimization. Vol. 1, Unconstrained Optimization*, Wiley, Chichester, 1980.

mtsvd

Purpose:

Regularization by means of modified TSVD.

Synopsis:

[x_k,rho,eta] = mtsvd (U,s,V,b,k,L)

Description:

The modified TSVD solution is defined as the solution to the problem

$$\min \|\mathsf{L}\,\mathbf{x}\|_2 \qquad \text{subject to} \qquad \|A_{\mathsf{k}}\,\mathbf{x} - \mathbf{b}\|_2 = \min \ ,$$

where A_{k} is the truncated SVD of the matrix A. The MTSVD solution is then given by

$$\mathsf{x\_k} = \mathsf{V} \begin{pmatrix} \xi_{\mathsf{k}} \\ \xi_0 \end{pmatrix} .$$

Here, ξ_{k} defines the usual TSVD solution $\xi_{\mathsf{k}} = (\mathrm{diag}(\mathsf{s}(1:\mathsf{k})))^{-1}\mathsf{U}(:,1:\mathsf{k})^T\mathsf{b}$, and ξ_0 is chosen so as to minimize the seminorm $\|\mathsf{L}\,\mathsf{x}_{\mathsf{k}}\|_2$. This leads to choosing ξ_0 as the solution to the following least squares problem:

$$\min \|(\mathsf{L}\,\mathsf{V}(:,\mathsf{k}+1:\mathsf{n}))\,\xi - \mathsf{L}\,\mathsf{V}(:,1:\mathsf{k})\,\xi_{\mathsf{k}}\|_2 \ .$$

The truncation parameter must satisfy $\mathsf{k} > n - p$, where $\mathsf{L} \in \mathbb{R}^{p\times n}$ The solution and residual norms are returned in eta and rho.

If k is a vector, then x_k is a matrix such that

$$\mathsf{x\_k} = [\,\mathsf{x\_k}(1),\, \mathsf{x\_k}(2),\, \ldots\,] \ .$$

Examples:

Compute the MTSVD solutions corresponding to k = 5:12:

X = mtsvd (U,s,V,b,5:12,L);

Algorithm:

The algorithm computes one QR factorization of the matrix $\mathsf{L}\,\mathsf{V}(:,\mathsf{k}_{\min}+1:\mathsf{n})$, where $\mathsf{k}_{\min} = \min(\mathsf{k})$. For more details, cf. [1].

See also:

discrep, lsqi, tgsvd, tikhonov, tsvd

References:

1. P. C. Hansen, T. Sekii & H. Shibahashi, *The modified truncated-SVD method for regularization in general form*, SIAM J. Sci. Stat. Comput. **13** (1992), 1142-1150.

newton

Purpose:

Newton iteration (utility routine for discrep).

Synopsis:

lambda = newton (lambda_0,delta,s,beta,omega,delta_0)

Description:

newton uses Newton iteration to find the solution to the equation

$$\|A\,\mathbf{x}_\lambda - \mathbf{b}\|_2 = \mathsf{delta}\ ,$$

where $\mathbf{x}_\lambda$ is the solution defined by Tikhonov regularization.

The initial guess of λ is lambda_0. The norm $\|A\,\mathbf{x}_\lambda - \mathbf{b}\|_2$ is computed via the quantities s, beta, omega, and delta_0. Here, s holds either the singular values of A, if $L = I_n$, or the c,s-pairs of the GSVD of (A,L), if $L \neq I_n$. Moreover, beta $= U^T\mathbf{b}$ and omega is either $V^T\mathbf{x}_0$ or the first p elements of $X^{-1}\mathbf{x}_0$. Finally, delta_0 is the incompatibility measure $\|(I_m - U\,U^T)\mathbf{b}\|_2$.

Algorithm:

It was experimentally found that with the initial guess lambda_0 from discrep, Newton's method applied to $\|A\,\mathbf{x}_\lambda - \mathbf{b}\|_2^2 - \mathsf{delta}^2$ is faster than when applied to $\|A\,\mathbf{x}_\lambda - \mathbf{b}\|_2^{-2} - \mathsf{delta}^{-2}$, as suggested in [1].

Diagnostics:

The performance of newton relies to some extent on a fairly good starting guess lambda_0 of λ. A maximum number of 50 iterations is default.

See also:

discrep

References:

1. V. A. Morozov, *Methods for Solving Incorrectly Posed Problems*, Springer, New York, 1984; Chapter 26.

nu

Purpose:

Brakhage's ν-method.

Synopsis:

[X,rho,eta,F] = nu (A,b,k,nu,s)

Description:

Performs k steps of Brakhage's ν-method for the problem

$$\min \|\mathsf{A}\,\mathbf{x} - \mathsf{b}\|_2 \ .$$

The routine returns all k solutions, stored as columns of the matrix X. The solution norms and residual norms are returned in eta and rho, respectively.

If nu is not specified, nu = .5 is the default value which gives the Chebychev method of Nemirovskii and Polyak.

If the singular values s are also provided, nu computes the filter factors associated with each step and stores them columnwise in the array F.

A "preconditioned" version of the ν-method for the general-form problem where one minimizes $\|L\,\mathbf{x}\|_2$ instead of $\|\mathbf{x}\|_2$ in each step is implemented in routine pnu.

Examples:

Perform 50 iterations of the ν-method and plot the corresponding L-curve:

[X,rho,eta] = nu (A,b,50); plot_lc (rho,eta,'o');

Algorithm:

nu is a straightforward implementation of the algorithm described in [1]. The iteration converges only if $\|\mathsf{A}\|_2 < 1$. Therefore, A and b are scaled before the iteration begins with a scaling factor given by $0.99/\|B\|_2$, where B is a bidiagonal matrix obtained from a few steps of the Lanczos bidiagonalization process applied to A with starting vector b. Hence, $\|B\|_2$ is a good approximation to $\|\mathsf{A}\|_2$.

See also:

cgls, pnu

References:

1. H. Brakhage, *On ill-posed problems and the method of conjugate gradients*; in H. W. Engl & G. W. Groetsch (Eds.), *Inverse and Ill-Posed Problems*, Academic Press, New York, 1987.

parallax

Purpose:

Stellar parallax problem with 28 fixed, real observations.

Synopsis:

[A,b] = parallax (n)

Description:

The underlying problem is a Fredholm integral equation of the first kind with kernel

$$K(s,t) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{s-t}{\sigma}\right)^2\right)$$

with $\sigma = 0.014234$, and it is discretized by means of a Galerkin method with n orthonormal basis functions. The right-hand side b consists of a measured distribution function of stellar parallaxes from [1], and its length is fixed at 26; i.e., the matrix A is $26 \times$ n. The exact solution, which represents the true distribution of stellar parallaxes, is unknown.

References:

1. W. M. Smart, *Stellar Dynamics*, Cambridge University Press, Cambridge, 1938; p. 30.

pcgls

Purpose:

"Preconditioned" conjugate gradient algorithm applied implicitly to the normal equations.

Synopsis:

[X,rho,eta,F] = pcgls (A,L,W,b,k,sm)

Description:

Performs k steps of the "preconditioned" conjugate gradient algorithm applied implicitly to the normal equations $\mathsf{A}^T\mathsf{A}\,\mathbf{x} = \mathsf{A}^T\mathsf{b}$ with "preconditioner" $\mathsf{L}_\mathsf{A}^\dagger\,(\mathsf{L}_\mathsf{A}^\dagger)^T$ and with starting vector $\mathbf{x}_0$ (2.30) computed by means of Algorithm (2.34). Here, $\mathsf{L}_\mathsf{A}^\dagger$ is the A-weighted generalized inverse of L. Notice that the matrix W holding a basis for the null space of L must also be specified.

The routine returns all k solutions, stored as columns of the matrix X. The solution seminorms and the residual norms are returned in eta and rho, respectively.

If the c, s-pairs sm of the GSVD of (A, L) are also provided, then pcgls computes the filter factors associated with each step and stores them in the array F.

Reorthogonalization of the normal equation residual vectors $\mathsf{A}^T(\mathsf{A}\,\mathsf{X}(:,i) - \mathsf{b})$, $i = 1,\ldots,\mathsf{k}$ is controlled by means of reorth as follows:

reorth = 0 :   no reorthogonalization
reorth = 1 :   reorthogonalization by means of MGS.

No reorthogonalization is assumed if reorth is not specified.

A simpler version of pcgls for the case $\mathsf{L} = I_n$ is implemented in routine cgls.

Examples:

Choose minimization of the second derivative as side constraint, and perform 20 iterations of the "preconditioned" conjugate gradient method:

[L,W] = get_l (n,2); X = pcgls (A,L,W,b,20); mesh (X)

Algorithm:

pcgls is a straightforward implementation of the algorithm described in [1], with the necessary modifications to the case $\mathsf{L} \neq I_n$ from [2]. The computation of the filter factors is also described in [2].

See also:

cgls, lsqr, nu, plsqr, pnu

References:

1. Å. Björck, *Numerical Methods for Least Squares Problems*, SIAM, Philadelphia, 1996.
2. P. C. Hansen, *Rank-Deficient and Discrete Ill-Posed Problems. Numerical Aspects of Linear Inversion,* SIAM, Philadelphia, 1997.

phillips

Purpose:

Phillips's "famous" test problem.

Synopsis:

[A,b,x] = phillips (n)

Description:

Discretization of the "famous" Fredholm integral equation of the first kind (2.1) deviced by D. L. Phillips [1]. Define the function

$$\phi(x) = \begin{cases} 1 + \cos\left(\frac{\pi x}{3}\right), & |x| < 3 \\ 0, & |x| \geq 3 \end{cases} .$$

Then the kernel K, the solution f, and the right-hand side g are given by:

$$\begin{aligned} K(s,t) &= \phi(s-t) \\ f(t) &= \phi(t) \\ g(s) &= (6 - |s|)\left(1 + \frac{1}{2}\cos\left(\frac{\pi s}{3}\right)\right) + \frac{9}{2\pi}\sin\left(\frac{\pi |s|}{3}\right). \end{aligned}$$

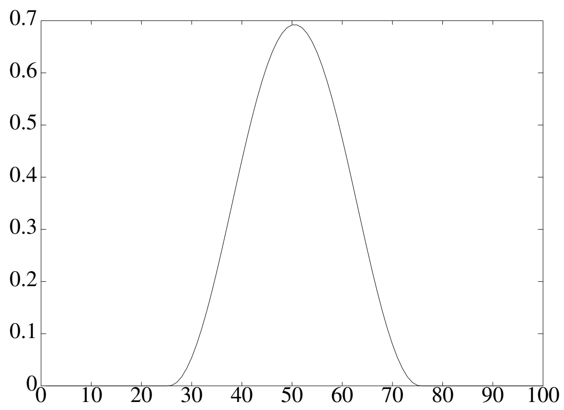
Both integration intervals are $[-6, 6]$. The size of the matrix A is n $\times$ n.

Limitations:

The order n must be a multiple of 4.

References:

1. D. L. Phillips, *A technique for the numerical solution of certain integral equations of the first kind*, J. ACM **9** (1962), 84–97.



picard

Purpose:

Visual inspection of the Picard condition.

Synopsis:

xi = picard (U,s,b,d)

xi = picard (U,sm,b,d) ,   sm = [sigma,mu]

Description:

picard plots the singular values s, the absolute values of the Fourier coefficients, $|\mathsf{U}^T\mathsf{b}|$, and a (possible smoothed) curve of the solution coefficients $|(\mathsf{U}^T\mathsf{b})./\mathsf{s}|$. If the c,s-values sm = [sigma,mu] are specified, where γ = sigma./mu are the generalized singular values, then the routine plots γ, $|\mathsf{U}^T\mathsf{b}|$, and (smoothed) $|(\mathsf{U}^T\mathsf{b})./\gamma|$.

The smoothing is a geometric mean over $2\mathsf{d}+1$ points. If nargin = 3 then d = 0 (i.e., no smoothing).

The quantities plotted by picard are useful for visually inspecting whether the discrete Picard condition is satisfied for the given problem: for the large singular values s, or the large generalized singular values γ, the Fourier coefficients $|\mathsf{U}^T\mathsf{b}|$ should decay at least as fast as the s and γ, respectively.

Examples:

Generate a test problem, add noise to the right-hand side, and use picard to check the discrete Picard condition visually:

```
[A,b,x] = shaw (32); [U,s,V] = csvd (A); b = b + 1e-3*randn (size(b));
picard (U,s,b);
```

See also:

l_curve

References:

1. P. C. Hansen, *The discrete Picard condition for discrete ill-posed problems*, BIT **30** (1990), 658–672.

pinit

Purpose:

Utility initialization-procedure for the "preconditioned" iterative regularization methods.

Synopsis:

T = pinit (W,A)

[T,x_0] = pinit (W,A,b)

Description:

pinit is a utility routine used for initialization inside the iterative regularization methods. Given a matrix W whose columns span the null space of L, pinit computes a matrix T which is needed in the routines for treating general-form regularization problems.

If b is also specified then x_0, the component of the regularized solution in the null space of L, is also computed.

Algorithm:

T and x_0 are computed by the following procedure:

$$S \leftarrow (A\,W)^{\dagger}\ , \quad \mathsf{T} \leftarrow S\,\mathsf{A}$$

$$\mathsf{x\_0} \leftarrow \mathsf{W}\,S\,\mathsf{b}\ .$$

The Matlab command pinv is used to compute the pseudoinverse $(A\,W)^{\dagger}$.

See also:

get_l

References:

1. M. Hanke, *Regularization with differential operators. An iterative approach*, J. Numer. Funct. Anal. Optim. **13** (1992), 523-540.
2. P. C. Hansen, *Rank-Deficient and Discrete Ill-Posed Problems. Numerical Aspects of Linear Inversion,* SIAM, Philadelphia, 1997.

plot_lc

Purpose:

Plot the L-curve.

Synopsis:

plot_lc (rho,eta,marker,ps,reg_param)

Description:

Plots the L-curve, i.e., the L-shaped curve of the solution norm

eta $= \|\mathbf{x}\|_2$ if ps $= 1$
eta $= \|L\,\mathbf{x}\|_2$ if ps $= 2$

versus the residual norm rho $= \|A\,\mathbf{x} - \mathbf{b}\|_2$. If ps is not specified, the value ps $= 1$ is assumed.

The text string marker is used as marker for the L-curve. If marker is not specified, the marker '–' is used, giving a solid line.

If a fifth argument reg_param is present, holding the regularization parameters corresponding to rho and eta, then some points on the L-curve are identified by their corresponding regularization parameter.

Diagnostics:

The default number of identified points on the L-curve is 10. It is controlled by the parameter np inside plot_lc.

See also:

l_curve

plsqr

Purpose:

"Preconditioned" version of the LSQR Lanczos bidiagonalization algorithm.

Synopsis:

[X,rho,eta,F] = plsqr (A,L,W,b,k,reorth,sm)

Description:

Performs k steps of the "preconditioned" LSQR Lanczos bidiagonalization algorithm applied to the system

$$\min \|\mathsf{A}\,\mathbf{x} - \mathsf{b}\|_2$$

with "preconditioner" $\mathsf{L}_\mathsf{A}^\dagger\,(\mathsf{L}_\mathsf{A}^\dagger)^T$ and with starting vector $\mathbf{x}_0$ (2.30). Here, $\mathsf{L}_\mathsf{A}^\dagger$ is the A-weighted generalized inverse of L. Notice that the matrix W holding a basis for the null space of L must also be specified. The routine returns all k solutions, stored as columns of the matrix X. The solution seminorms and the residual norms are returned in eta and rho, respectively.

Reorthogonalization of the Lanczos vectors is controlled by means of reorth as follows:

reorth = 0 : no reorthogonalization
reorth = 1 : reorthogonalization by means of MGS
reorth = 2 : Householder reorthogonalization.

No reorthogonalization is assumed if reorth is not specified.

If the c, s-values sm of the GSVD of (A, L) are also provided, then plsqr computes the filter factors associated with each step and stores them columnwise in the array F.

A simpler version of plsqr for the case $\mathsf{L} = I_n$ is implemented in routine lsqr.

Examples:

Choose minimization of the second derivative as side constraint, perform 20 iterations of the "preconditioned" LSQR algorithm with no reorthogonalization including computation of the filter factors, and display the filter factors:

```
[L,W] = get_l (n,2); sm = gsvd (A,L);
[X,rho,eta,F] = plsqr (A,L,W,b,25,0,s); mesh (F), axis ('ij')
```

Algorithm:

plsqr is an implementation of the LSQR algorithm [2] with the necessary modifications for "preconditioning" described in [2].

See also:

cgls, lsqr, pcgls

References:

1. P. C. Hansen, *Rank-Deficient and Discrete Ill-Posed Problems. Numerical Aspects of Linear Inversion,* SIAM, Philadelphia, 1997.
2. C. C. Paige & M. A. Saunders, *LSQR: an algorithm for sparse linear equations and sparse least squares*, ACM Trans. Math. Software **8** (1982), 43–71.

pnu

Purpose:

"Preconditioned" version of Brakhage's ν-method.

Synopsis:

[X,rho,eta,F] = pnu (A,L,W,b,k,nu,sm)

Description:

Performs k steps of the "preconditioned" version of Brakhage's ν-method applied to the system

$$\min \|\mathsf{A}\,\mathbf{x} - \mathsf{b}\|_2$$

with "preconditioner" $\mathsf{L}_\mathsf{A}^\dagger\,(\mathsf{L}_\mathsf{A}^\dagger)^T$ and with starting vector $\mathbf{x}_0$ (2.30). Here, $\mathsf{L}_\mathsf{A}^\dagger$ is the A-weighted generalized inverse of L. Notice that the matrix W holding a basis for the null space of L must also be specified. The routine returns all k solutions, stored as columns of the matrix X. The solution seminorms and the residual norms are returned in eta and rho, respectively.

If nu is not specified, nu = .5 is the default value which gives the "preconditioned" version of the Chebychev method of Nemirovskii and Polyak.

If the c,s-values sm of the GSVD of (A,L) are also provided, then pnu computes the filter factors associated with each step and stores them columnwise in the array F.

A simpler version of pnu for the case $\mathsf{L} = I_n$ is implemented in routine nu.

Examples:

Perform 50 iterations of the "preconditioned" ν-method with $\nu = .2$ and plot the corresponding L-curve:

[X,rho,eta] = pnu (A,L,W,b,50,.2); plot_lc (rho,eta,'o');

Algorithm:

pnu is a straightforward implementation of the algorithm described in [1], with the necessary modifications for "preconditioning" from [2]. The iteration converges only if $\|\mathsf{A}\,\mathsf{L}_\mathsf{A}^\dagger\|_2 < 1$. Therefore, A and b are scaled before the iteration begins with a scaling factor given by $0.99/\|B\|_2$, where B is a bidiagonal matrix obtained from a few steps of the Lanczos bidiagonalization process applied to $\mathsf{A}\,\mathsf{L}_\mathsf{A}^\dagger$ with starting vector b. Hence, $\|B\|_2$ is a good approximation to $\|\mathsf{A}\,\mathsf{L}_\mathsf{A}^\dagger\|_2$.

See also:

cgls, nu, pcgls

References:

1. H. Brakhage, *On ill-posed problems and the method of conjugate gradients*; in H. W. Engl & G. W. Groetsch, *Inverse and Ill-Posed Problems*, Academic Press, New York, 1987.
2. P. C. Hansen, *Rank-Deficient and Discrete Ill-Posed Problems. Numerical Aspects of Linear Inversion,* SIAM, Philadelphia, 1997.

pythag

Purpose:

Compute $x = \sqrt{y^2 + z^2}$.

Synopsis:

x = pythag (y,z)

Description:

pythag returns $x = \sqrt{y^2 + z^2}$ but is careful to scale to avoid overflow.

quasiopt

Purpose:

Quasi-optimality criterion for choosing the regularization parameter.

Synopsis:

[reg_min,Q,reg_param] = quasiopt (U,s,b,method)

[reg_min,Q,reg_param] = quasiopt (U,sm,b,method) , sm = [sigma,mu]

Description:

Plots the quasi-optimality function Q, cf. (2.64), for the following methods:

method = 'Tikh'	:	Tikhonov regularization	(solid line)
method = 'tsvd'	:	truncated SVD or GSVD	(o markers)
method = 'dsvd'	:	damped SVD or GSVD	(dotted line)

If no method is specified, 'Tikh' is default. Returns Q and the corresponding regularization parameters in reg_param.

If any output arguments are specified, then the approximate minimum of Q is identified and the corresponding regularization parameter reg_min is returned.

Examples:

Use the quasi-optimality criterion to find the optimal regularization parameter for Tikhonov regularization in general form:

```
[A,b,x] = shaw (n); b = b + 1e-3*randn (size(b)); [U,s,V] = svd (A);
lambda_opt = quasiopt (U,s,b);
```

Algorithm:

For Tikhonov regularization and damped SVD/GSVD, 200 logarithmically distributed regularization parameters are generated, and Q is plotted for these values. Then the minimizer of the quasi-optimality function is computed via fmin, using the minimizer of Q as initial guess. For truncated SVD and GSVD, the quasi-optimality function is discrete and simply becomes Q = (U′ ∗ b)./s and Q = (U′ ∗ b)./(sigma./mu), respectively, and the minimum is easily found.

Limitations:

The method is not implemented for MTSVD; for this and other discrete regularization methods use the relations $\mathsf{Q}(k) = \|\mathbf{x}_k - \mathbf{x}_{k-1}\|_2$ and $\mathsf{Q}(k) = \|L\,(\mathbf{x}_k - \mathbf{x}_{k-1})\|_2$.

See also:

gcv, discrep, l_curve

regudemo

Purpose:

Tutorial introduction to REGULARIZATION TOOLS.

Synopsis:

regudemo

Description:

This script contains all the Matlab commands used in the Tutorial in Section 3, with appropriate pause statements added.

Diagnostics:

The user may want to experiment with other seeds for the random number generator in the beginning of the script, as well as other noise levels.

regutm

Purpose:

Generates random test matrices for regularization methods.

Synopsis:

[A,U,V] = regutm (m,n,s)

Description:

Generates a random $m \times n$ matrix A such that $A A^T$ and $A^T A$ are oscillating. Hence, in the SVD of A,

$$A = U \operatorname{diag}(s) V^T ,$$

the number of sign changes in the column vectors $U(:,i)$ and $V(:,i)$ is exactly $i-1$.

The third argument s specifies the singular values of A. If not present, then s = logspace(0,round(log10(eps)),n).

Algorithm:

If $m = n$ then U and V are computed as the left and right singular matrices of a random bidiagonal $n \times n$ matrix with nonnegative elements. If $m \neq n$ then V is computed as above, and U is computed analogously via a random $m \times m$ bidiagonal matrix. Finally, A is computed as $A = U \operatorname{diag}(s) V^T$.

See also:

The test problem routines.

References:

1. P. C. Hansen, *Test matrices for regularization methods*, SIAM J. Sci. Comput. **16** (1995), 506–512.

shaw

Purpose:

Test problem: one-dimensional image restoration model.

Synopsis:

[A,b,x] = shaw (n)

Description:

Discretization of a Fredholm integral equation of the first kind (2.1) with $[-\pi/2, \pi/2]$ as both integration intervals. The integral equation is a one-dimensional model of an image reconstruction problem from [1]. The kernel K and the solution f are given by

$$\begin{aligned} K(s,t) &= (\cos(s) + \cos(t))^2 \left(\frac{\sin(u)}{u}\right)^2 \\ u &= \pi\ (\sin(s) + \sin(t)) \\ f(t) &= a_1 \exp\left(-c_1(t-t_1)^2\right) + a_2 \exp\left(-c_2(t-t_2)^2\right). \end{aligned}$$

The kernel K is the point spread function for an infinitely long slit. The parameters a_1, a_2, etc., are constants that determine the shape of the solution f; in this implementation we use $a_1 = 2$, $a_2 = 1$, $c_1 = 6$, $c_2 = 2$, $t_1 = .8$, $t_2 = -.5$ giving an f with two different "humps".

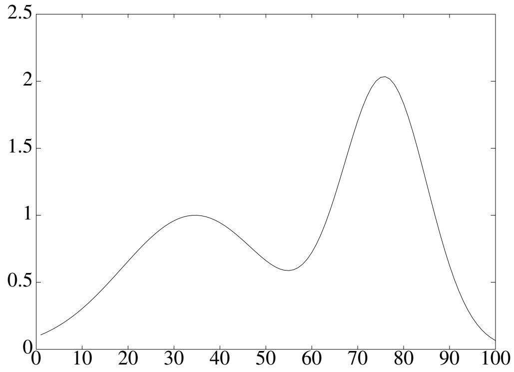
The kernel and the solution are discretized by simple collocation with n points to produce A and x. Then the discrete right-hand side is produced as b = A x.

Limitations:

The order n must be even.

Reference:

1. C. B. Shaw, Jr., *Improvements of the resolution of an instrument by numerical solution of an integral equation*, J. Math. Anal. Appl. **37** (1972), 83–112.



spikes

Purpose:

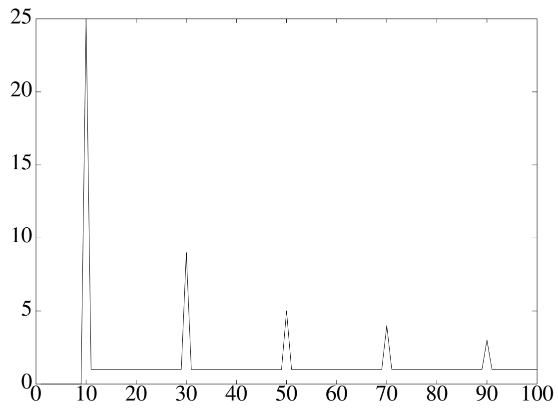
Test problem with a "spiky" solution.

Synopsis:

[A,b,x] = spikes (n,t_max)

Description:

Artificially generated discrete ill-posed problem. The size of the matrix A is $n \times n$. The solution x consists of a unit step at $t = .5$, and a pulse train of spikes of decreasing amplitude at $t = .5, 1.5, 2.5, \ldots$ The parameter t_max controls the length of the pulse train (i.e., the time unit of x is t_max/n); t_max is optional, and its default is 5 giving five spikes.



spleval

Purpose:

Evaluation of a spline or spline curve.

Synopsis:

points = spleval (f)

Description:

Computes points on the given spline or spline curve f between its extreme breaks. The representation f for the spline is assumed to be consistent with that used in the Spline Toolbox [1]. Routine spleval is used in l_corner only.

Algorithm:

Original routine fnplt from de Boor's Spline Toolbox; modified by P. C. Hansen to skip the plot. The number of points is set by npoints inside spleval; currently, npoints = 300.

References:

1. C. de Boor, *Spline Toolbox*, Version 1.1, The Mathworks, MA, 1992.

std_form

Purpose:

Transform a general-form regularization problem into one in standard form.

Synopsis:

[A_s,b_s,L_p,K,M] = std_form (A,L,b) (method 1)
[A_s,b_s,L_p,x_0] = std_form (A,L,b,W) (method 2)

Description:

Transforms a regularization problem in general form

$$\min \left\{ \|\mathsf{A}\,\mathbf{x} - \mathsf{b}\|_2^2 + \lambda^2 \, \|\mathsf{L}\,\mathbf{x}\|_2^2 \right\}$$

into one in standard form

$$\min \left\{ \|\mathsf{A\_s}\,\mathbf{x}_s - \mathsf{b\_s}\|_2^2 + \lambda^2 \, \|\mathbf{x}_s\|_2^2 \right\} \ .$$

Two methods are available, and the distinction between them is controlled by the number of input parameters to std_form.

In method 1, described in [1], A_s, L_p, K, M, and b_s are computed by Eqs. (2.22)–(2.24). In particular, L_p is the pseudoinverse of L, $\mathsf{K} = K_o$ has columns in the null space of L, and $\mathsf{M} = T_o^{-1} H_o^T$. Then the transformation back to the original general-form setting is given by

$$\mathsf{x} = \mathsf{L\_p}\,\mathsf{x\_s} + \mathsf{K}\,\mathsf{M}\,(\mathsf{b} - \mathsf{A}\,\mathsf{L\_p}\,\mathsf{x\_s}) \ .$$

In method 2, described in [2,3,4], a matrix W holding a basis for the null space of L is also required, and A_s, L_p, b_s, and x_0 are computed by Eqs. (2.29)–(2.31). Here L_p is the A-weighted generalized inverse of L, and x_0 is the component of the solution in the null space of L (which, in this method, is independent of $\mathbf{x}_s$). For method 2, the tranformation back to the general-form setting is simply

$$\mathsf{x\_s} = \mathsf{L\_p}\,\mathsf{x\_s} + \mathsf{x\_0} \ .$$

The transformations back to general form can be performed by means of routine gen_form.

Notice that both gen_form and std_form are available for pedagogical reasons only—usually it is more efficient to build the transformations into the solution process, such as in the iterative methods pcgls, plsqr, and pnu.

Examples:

Transform a general-form problem into standard form, produce 10 TSVD solutions, and transform back again, using method 1; then compare with the mathematically identical TGSVD solutions:

```
[A_s,b_s,L_p,K,M] = std_form (A,L,b); [U,s,V] = csvd (A_s);
X_s = tsvd (U,s,V,b_s,1:10); X = gen_form (L_p,X_s,A,b,K,M);
[U1,sm,X1] = cgsvd (A,L); XX = tgsvd (U1,sm,X1,b,1:10); norm (X-XX)
```

Algorithm:

The formulas used in method 1 are:

$$\mathsf{L}^T = (K_p \; K_o) \begin{pmatrix} R_p \\ 0 \end{pmatrix} \quad \text{(QR factorization)}, \qquad \mathsf{L\_p} = K_p \, R_p^{-T}$$

$$\mathsf{A}\, K_o = (H_o \, H_q) \begin{pmatrix} T_o \\ 0 \end{pmatrix} \quad \text{(QR factorization)}, \qquad \mathsf{A\_s} = H_q^T \mathsf{A}\, \mathsf{L\_p}$$

$$\mathsf{K} = K_o \;, \qquad \mathsf{M} = T_o^{-1} H_o^T \;, \qquad \mathsf{b\_s} = H_q^T \mathsf{b} \;.$$

The formulas used in method 2 are the following (see also pinit):

$$[\mathsf{T}, \mathsf{x\_0}] = \mathsf{pinit}\,(\mathsf{W}, \mathsf{A}, \mathsf{b}) \;;$$

$$\mathsf{L\_p} = \left(\begin{pmatrix} I_p \\ 0 \end{pmatrix} - \mathsf{W}\, \mathsf{T}(:\,, 1\!:p) \right) L(:\,, 1\!:p)^{-1}$$

$$\mathsf{A\_s} = \mathsf{A}\, \mathsf{L\_p} \;.$$

See also:

gen_form

References:

1. L. Eldén, *Algorithms for regularization of ill-conditioned least-squares problems*, BIT **17** (1977), 134–145.
2. L. Eldén, *A weighted pseudoinverse, generalized singular values, and constrained least squares problems*, BIT **22** (1982), 487–502.
3. M. Hanke, *Regularization with differential operators. An iterative approach*, J. Numer. Funct. Anal. Optim. **13** (1992), 523–540.
4. P. C. Hansen, *Rank-Deficient and Discrete Ill-Posed Problems. Numerical Aspects of Linear Inversion,* SIAM, Philadelphia, 1997.

tgsvd

Purpose:

Compute the truncated GSVD solution.

Synopsis:

[x_k,rho,eta] = tgsvd (U,sm,X,b,k) , sm = [sigma,mu]

Description:

Computes the truncated GSVD solution, defined as

$$\mathsf{x\_k} = \sum_{i=p-\mathsf{k}+1}^{p} \frac{\mathsf{U}(:,i)^T \mathsf{b}}{\mathsf{sigma}(i)} \mathsf{X}(:,i) + \sum_{i=p+1}^{n} \mathsf{U}(:,i)^T \mathsf{b}\, \mathsf{X}(:,i) \ .$$

If k is a vector, then x_k is a matrix such that

$$\mathsf{x\_k} = [\, \mathsf{x\_k}(1),\ \mathsf{x\_k}(2),\ \ldots \,] \ .$$

The solution and residual norms are returned in eta and rho.

Examples:

Compute the first 10 TGSVD solutions to an inverse Laplace transform model-problem:

[A,b,x] = ilaplace (n,2); L = get_l (n,1); b = b + 1e-4*randn (size(b));

[U,sm,X] = cgsvd (A,L); X_tgsvd = tgsvd (U,sm,X,b,1:10);

Algorithm:

Straightforward use of the above definition is used to compute x_k. For motivations for the TGSVD solution, cf. [1].

See also:

mtsvd, tsvd

References:

1. P. C. Hansen, *Regularization, GSVD and truncated GSVD*, BIT **29** (1989), 491–504.

tikhonov

Purpose:

Compute the Tikhonov regularized solution.

Synopsis:

[x_lambda,rho,eta] = tikhonov (U,s,V,b,lambda,x_0)

[x_lambda,rho,eta] = tikhonov (U,sm,X,b,lambda,x_0) ,   sm = [sigma,mu]

Description:

Compute the solution by means of Tikhonov regularization. If the SVD of A is used, i.e. if U, s, and V are given as input, then Tikhonov regularization in standard form is used, and x_lambda solves the problem

$$\min \left\{ \|A\,\mathsf{x} - \mathsf{b}\|_2^2 + \mathsf{lambda}^2 \|\mathsf{x} - \mathsf{x\_0}\|_2^2 \right\} \ .$$

On the other hand, if the GSVD of (A, L) is used, i.e. if U, sm, and X are given as input, then Tikhonov regularization in general form is used, and x_lambda then solves the problem

$$\min \left\{ \|A\,\mathsf{x} - \mathsf{b}\|_2^2 + \mathsf{lambda}^2 \|L\,(\mathsf{x} - \mathsf{x\_0})\|_2^2 \right\} \ .$$

If x_0 is not specified, then x_0 = **0** is used. If lambda is a vector, then x_lambda is a matrix such that

$$\mathsf{x\_lambda} = [\,\mathsf{x\_lambda}(1),\, \mathsf{x\_lambda}(2),\, \ldots\,] \ .$$

The solution and residual norms are returned in eta and rho.

Examples:

Solve a discrete ill-posed problem for three values of the regularization parameter, namely 10^{-1}, 10^{-3}, and 10^{-5}:

x_lambda = tikhonov (U,s,V,b,[1e-1,1e-3,1e-5]); plot (x_lambda)

Algorithm:

x_lambda is computed by straightforward use of the formulas from Section 2.4.

See also:

discrep, lsqi, mtsvd

References:

1. A. N. Tikhonov & V. Y. Arsenin, *Solutions of Ill-Posed Problems*, Winston & Sons, Washington, D.C., 1977.

tsvd

Purpose:

Compute the truncated SVD solution.

Synopsis:

[x_k,rho,eta] = tsvd (U,s,V,b,k)

Description:

Computes the truncated SVD solution, defined as

$$\mathsf{x\_k} = \sum_{i=1}^{\mathsf{k}} \frac{\mathsf{U}(:,i)^T \mathsf{b}}{\mathsf{s}(i)} \mathsf{V}(:,i) \ .$$

If k is a vector, then x_k is a matrix such that

$$\mathsf{x\_k} = [\, \mathsf{x\_k}(1), \, \mathsf{x\_k}(2), \, \ldots] \ .$$

The solution and residual norms are returned in eta and rho.

Examples:

Compute the TSVD solutions for the truncation parameters 3, 6, 9, 12, and 15:

[A,b,x] = shaw (n); b = b + 1e-3*rand (size(b)); [U,s,V] = csvd (A);

X = tsvd (U,s,V,b,[3,6,9,12,15]); plot ([x,X])

Algorithm:

Straightforward use of the definition is used to compute x_k.

See also:

mtsvd, tgsvd

ttls

Purpose:

Compute the truncated TLS solution.

Synopsis:

[x_k,rho,eta] = ttls (V1,k,s1)

Description:

Computes the truncated TLS solution, given by

$$\mathsf{x\_k} = -\mathsf{V1}(1:n, \mathsf{k}+1:n+1)\,\mathsf{V1}(n+1, \mathsf{k}+1:n+1))^{\dagger}\ ,$$

where V1 is the right singular matrix in the SVD of the *compound matrix*

$$(A\ \mathbf{b}) = U_1\,\mathrm{diag}(\mathsf{s1})\,\mathsf{V1}^T\ .$$

If k is a vector, then x_k is a matrix such that

$$\mathsf{x\_k} = [\,\mathsf{x\_k}(1),\ \mathsf{x\_k}(2),\ \ldots]\ .$$

If k is not specified, the default value is $\mathsf{k} = n$ and x_k is then the standard TLS solution.

The solution norms $\|\mathsf{x\_k}\|_2$ and the corresponding residual norms $\|(A\ \mathbf{b}) - (\tilde{A}\ \tilde{\mathbf{b}})_k\|_F$, where $(\tilde{A}\ \tilde{\mathbf{b}})_k = U_1(:,1:\mathsf{k})\,\mathrm{diag}(\mathsf{s1}(1:\mathsf{k}))\,\mathsf{V1}(:,1:\mathsf{k})^T$, are returned in eta and rho, respectively. The singular values s1 of $(A\ \mathbf{b})$ are required to compute rho.

Examples:

Compute the truncated TLS solutions for $\mathsf{k} = n-5$, $n-10$, and $n-15$:

[U1,s1,V1] = csvd ([A,b],'full'); X = ttls (V1,n−[5,10,15]);

Algorithm:

x_k is computed by means of the definition, using the following relation for the pseudoinverse of a row vector $\mathbf{v}^T$:

$$(\mathbf{v}^T)^{\dagger} = \|\mathbf{v}\|_2^{-2}\,\mathbf{v}\ .$$

The norms are given by

$$\|\mathsf{x\_k}\|_2 = \sqrt{\|\mathsf{V1}(n+1, \mathsf{k}+1:n+1)\|_2^{-2} - 1}$$

$$\|(A\ \mathbf{b}) - (\tilde{A}\ \tilde{\mathbf{b}})_k\|_F = \|\mathsf{s1}(\mathsf{k}+1:n+1)\|_2\ .$$

References:

1. R. D. Fierro, G. H. Golub, P. C. Hansen & D. P. O'Leary, *Regularization by truncated total least squares*, SIAM J. Sci. Comput. **18** (1997), 1223–1241.

ursell

Purpose:

Test problem: integral equation with no square integrable solution.

Synopsis:

[A,b] = ursell (n)

Description:

Discretization of a Fredholm integral equation of the first kind (2.1) from [1] with kernel K and right-hand side g given by

$$K(s,t) = \frac{1}{s+t+1} \ , \qquad g(s) = 1 \ ,$$

where both integration intervals are $[0,1]$. This integral equation does not satisfy the Picard condition and has *no* square integrable solution (hence, no x is produced). The size of the matrix A is n $\times$ n.

Examples:

Generate A and b and check the discrete Picard condition:

[A,b] = ursell (16); [U,s,V] = csvd (A); picard (U,s,b);

References:

1. F. Ursell, *Introduction to the theory of linear integral equations*, Chapter 1 in L. M. Delves & J. Walsh (Eds.), *Numerical Solution of Integral Equations*, Clarendon Press, Oxford, 1974.

wing

Purpose:

Test problem with a discontinuous solution.

Synopsis:

[A,b,x] = wing (n,t1,t2)

Description:

Discretization of a Fredholm integral equation of the first kind (2.1) from [1, p. 109] with both integration intervals equal to $[0,1]$, with kernel K and right-hand side g given by

$$K(s,t) = t\,\exp(-\,s\,t^2)\ , \qquad g(s) = \frac{\exp(-\,s\,\mathsf{t1}^2) - \exp(-\,s\,\mathsf{t2}^2)}{2\,s}\ ,$$

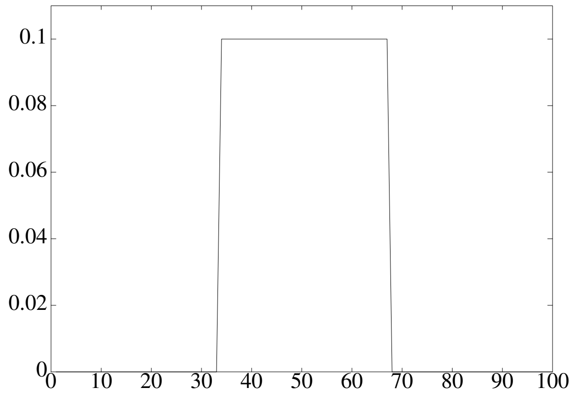
and with the solution f given by

$$f(t) = \begin{cases} 1, & \text{for} \quad \mathsf{t1} < t < \mathsf{t2} \\ 0, & \text{elsewhere.} \end{cases}$$

Here, t1 and t2 are constants satisfying $0 < \mathsf{t1} < \mathsf{t2} < 1$. If they are not specified, the values $\mathsf{t1} = 1/3$ and $\mathsf{t2} = 2/3$ are used. The size of the matrix A is $\mathsf{n} \times \mathsf{n}$.

References:

1. G. M. Wing & J. D. Zahrt, *A Primer on Integral Equations of the First Kind*, SIAM, Philadelphia, 1991; p. 109.



Bibliography

[1] R. C. Allen, Jr., W. R. Boland, V. Faber & G. M. Wing, *Singular values and condition numbers of Galerkin matrices arising from linear integral equations of the first kind*, J. Math. Anal. Appl. **109** (1985), 564–590.

[2] R. S. Anderssen & P. M. Prenter, *A formal comparison of methods proposed for the numerical solution of first kind integral equations*, J. Austral. Math. Soc. (Series B) **22** (1981), 488–500.

[3] C. T. H. Baker, *Expansion methods*, Chapter 7 in [18].

[4] C. T. H. Baker, *The Numerical Treatment of Integral Equations*, Clarendon Press, Oxford, 1977.

[5] C. T. H. Baker & G. F. Miller (Eds.), *Treatment of Integral Equations by Numerical Methods*, Academic Press, New York, 1982.

[6] D. M. Bates, M. J. Lindstrom, G. Wahba & B. S. Yandell, *GCVPACK – routines for generalized cross validation*, Commun. Statist.-Simula. **16** (1987), 263–297.

[7] M. Bertero, C. De Mol & E. R. Pike, *Linear inverse problems with discrete data: II. Stability and regularization*, Inverse Problems **4** (1988), 573–594.

[8] Å. Björck, *A bidiagonalization algorithm for solving large and sparse ill-posed systems of linear equations*, BIT **28** (1988), 659–670.

[9] Å. Björck, *Numerical Methods for Least Squares Problems*, SIAM, Philadelphia, 1996.

[10] Å. Björck & L. Eldén, *Methods in numerical algebra for ill-posed problems*, Report LiTH-MAT-R33-1979, Dept. of Mathematics, Linköping University, 1979.

[11] H. Brakhage, *On ill-posed problems and the method of conjugate gradients*; in H. W. Engl & C. W. Groetsch (Eds.), *Inverse and Ill-Posed Problems*, Academic Press, Boston, 1987.

[12] T. F. Chan & P. C. Hansen, *Some applications of the rank revealing QR factorization*, SIAM J. Sci. Stat. Comput. **13** (1992), 727–741.

[13] J. A. Cochran, *The Analysis of Linear Integral Equations*, McGraw-Hill, New York, 1972.

[14] D. Colton & R. Kress, *Integral Equation Methods for Scattering Theory*, John Wiley, New York, 1983.

[15] I. J. D. Craig & J. C. Brown, *Inverse Problems in Astronomy*, Adam Hilger, Bristol, 1986.

[16] J. J. M. Cuppen, *A Numerical Solution of the Inverse Problem of Electrocardiography*, Ph. D. Thesis, Dept. of Mathematics, Univ. of Amsterdam, 1983.

[17] L. M. Delves & J. L. Mohamed, *Computational Methods for Integral Equations*, Cambridge University Press, Cambridge, 1985.

[18] L. M. Delves & J. Walsh (Eds.), *Numerical Solution of Integral Equations*, Clarendon Press, Oxford, 1974.

[19] J. B. Drake, *ARIES: a computer program for the solution of first kind integral equations with noisy data*, Report K/CSD/TM-43, Dept. of Computer Science, Oak Ridge National Laboratory, October 1983.

[20] M. P. Ekstrom & R. L. Rhodes, *On the application of eigenvector expansions to numerical deconvolution*, J. Comp. Phys. **14** (1974), 319-340.

[21] L. Eldén, *Algorithms for regularization of ill-conditioned least-squares problems*, BIT **17** (1977), 134–145.

[22] L. Eldén, *A program for interactive regularization*, Report LiTH-MAT-R-79-25, Dept. of Mathematics, Linköping University, Sweden, 1979.

[23] L. Eldén, *A weighted pseudoinverse, generalized singular values, and constrained least squares problems*, BIT **22** (1982), 487–501.

[24] L. Eldén, *A note on the computation of the generalized cross-validation function for ill-conditioned least squares problems*, BIT **24** (1984), 467–472.

[25] H. W. Engl & J. Gfrerer, *A posteriori parameter choice for general regularization methods for solving linear ill-posed problems*, App. Numerical Math. **4** (1988), 395–417.

[26] R. D. Fierro & J. R. Bunch, *Collinearity and total least squares*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 1167–1181.

[27] R. D. Fierro, G. H. Golub, P. C. Hansen & D. P. O'Leary, *Regularization by truncated total least squares*, SIAM J. Sci. Comput. **18** (1997), 1223–1241.

[28] R. Fletcher, *Practical Optimization Methods. Vol. 1, Unconstrained Optimization*, Wiley, Chichester, 1980.

[29] G. H. Golub & C. F. Van Loan, *Matrix Computations*, 3. Ed., Johns Hopkins, Baltimore, 1996.

[30] G. H. Golub & U. von Matt, *Quadratically constrained least squares and quadratic problems*, Numer. Math. **59** (1991), 561–580.

[31] C. W. Groetsch, *The Theory of Tikhonov Regularization for Fredholm Equations of the First Kind*, Pitman, Boston, 1984.

[32] C. W. Groetsch, *Inverse Problems in the Mathematical Sciences*, Vieweg Verlag, Wiesbaden, 1993.

[33] C. W. Groetsch & C. R. Vogel, *Asymptotic theory of filtering for linear operator equations with discrete noisy data*, Math. Comp. **49** (1987), 499–506.

[34] J. Hadamard, *Lectures on Cauchy's Problem in Linear Partial Differential Equations*, Yale University Press, New Haven, 1923.

[35] M. Hanke, *Regularization with differential operators. An iterative approach*, J. Numer. Funct. Anal. Optim. **13** (1992), 523–540.

[36] M. Hanke, *Iterative solution of underdetermined linear systems by transformation to standard form;* in *Numerical Mathematics in Theory and Practice*, Dept. of Mathematics, University of West Bohemia, Plzeň, pp. 55–63 (1993).

[37] M. Hanke & P. C. Hansen, *Regularization Methods for Large-Scale Problems*, Surv. Math. Ind. **3** (1993), 253–315.

[38] P. C. Hansen, *The truncated SVD as a method for regularization*, BIT **27** (1987), 543–553.

[39] P. C. Hansen, *Computation of the singular value expansion*, Computing **40** (1988), 185–199.

[40] P. C. Hansen, *Perturbation bounds for discrete Tikhonov regularization*, Inverse Problems **5** (1989), L41–L44.

[41] P. C. Hansen, *Regularization, GSVD and truncated GSVD*, BIT **29** (1989), 491–504.

[42] P. C. Hansen, *Truncated SVD solutions to discrete ill-posed problems with ill-determined numerical rank*, SIAM J. Sci. Stat. Comput. **11** (1990), 503–518.

[43] P. C. Hansen, *Relations between SVD and GSVD of discrete regularization problems in standard and general form*, Lin. Alg. Appl. **141** (1990), 165–176.

[44] P. C. Hansen, *The discrete Picard condition for discrete ill-posed problems*, BIT **30** (1990), 658–672.

[45] P. C. Hansen, *Analysis of discrete ill-posed problems by means of the L-curve*, SIAM Review **34** (1992), 561–580.

[46] P. C. Hansen, *Numerical tools for analysis and solution of Fredholm integral equations of the first kind*, Inverse Problems **8** (1992), 849–872.

[47] P. C. Hansen, *Rank-Deficient and Discrete Ill-Posed Problems. Numerical Aspects of Linear Inversion*, SIAM, Philadelphia, 1997.

[48] P. C. Hansen & D. P. O'Leary, *The use of the L-curve in the regularization of discrete ill-posed problems*, SIAM J. Sci. Comput. **14** (1993), 1487–1503.

[49] P. C. Hansen, T. Sekii & H. Shibahashi, *The modified truncated SVD method for regularization in general form*, SIAM J. Sci. Stat. Comput. **13** (1992), 1142-1150.

[50] B. Hofmann, *Regularization for Applied Inverse and Ill-Posed Problems*, Teubner, Leipzig, 1986.

[51] R. Kress, *Linear Integral Equations*, Springer, Berlin, 1989.

[52] C. L. Lawson & R. J. Hanson, *Solving Least Squares Problems*, Prentice-Hall, Englewood Cliffs, 1974.

[53] P. Linz, *Uncertainty in the solution of linear operator equations*, BIT **24** (1984), 92–101.

[54] *Matlab Reference Guide*, The MathWorks, Mass., 1996.

[55] K. Miller, *Least squares methods for ill-posed problems with a prescribed bound*, SIAM J. Math. Anal. **1** (1970), 52–74.

[56] V. A. Morozov, *Methods for Solving Incorrectly Posed Problems*, Springer Verlag, New York, 1984.

[57] F. Natterer, *The Mathematics of Computerized Tomography*, John Wiley, New York, 1986.

[58] F. Natterer, *Numerical treatment of ill-posed problems*, in [65].

[59] D. P. O'Leary & J. A. Simmons, *A bidiagonalization-regularization procedure for large scale discretizations of ill-posed problems*, SIAM J. Sci. Stat. Comput. **2** (1981), 474–489.

[60] C. C. Paige & M. A. Saunders, *LSQR: an algorithm for sparse linear equations and sparse least squares*, ACM Trans. Math. Software **8** (1982), 43–71.

[61] R. L. Parker, *Understanding inverse theory*, Ann. Rev. Earth Planet Sci. **5** (1977), 35–64.

[62] D. L. Phillips, *A technique for the numerical solution of certain integral equations of the first kind*, J. ACM **9** (1962), 84–97.

[63] F. Santosa, Y.-H. Pao, W. W. Symes & C. Holland (Eds.), *Inverse Problems of Acoustic and Elastic Waves*, SIAM, Philadelphia, 1984.

[64] C. Ray Smith & W. T. Grandy, Jr. (Eds.), *Maximum-Entropy and Bayesian Methods in Inverse Problems*, Reidel, Boston, 1985.

[65] G. Talenti (Ed.), *Inverse Problems*, Lecture Notes in Mathematics 1225, Springer Verlag, Berlin, 1986.

[66] H. J. J. te Riele, *A program for solving first kind Fredholm integral equations by means of regularization*, Computer Physics Comm. **36** (1985), 423–432.

[67] A. N. Tikhonov, *Solution of incorrectly formulated problems and the regularization method*, Dokl. Akad. Nauk. SSSR **151** (1963), 501–504 = Soviet Math. Dokl. **4** (1963), 1035–1038.

[68] A. N. Tikhonov & V. Y. Arsenin, *Solutions of Ill-Posed Problems*, Winston & Sons, Washington, D.C., 1977.

[69] A. N. Tikhonov & A. V. Goncharsky, *Ill-Posed Problems in the Natural Sciences*, MIR Publishers, Moscow, 1987.

[70] A. van der Sluis, *The convergence behavior of conjugate gradients and Ritz values in various circumstances*; in R. Beauwens & P. de Groen (Eds.), *Iterative Methods in Linear Algebra*, North-Holland, Amsterdam, 1992.

[71] A. van der Sluis & H. A. van der Vorst, *SIRT- and CG-type methods for iterative solution of sparse linear least-squares problems*, Lin. Alg. Appl. **130** (1990), 257–302.

[72] J. M. Varah, *On the numerical solution of ill-conditioned linear systems with applications to ill-posed problems*, SIAM J. Numer. Anal. **10** (1973), 257–267.

[73] J. M. Varah, *A practical examination of some numerical methods for linear discrete ill-posed problems*, SIAM Rev. **21** (1979), 100–111.

[74] J. M. Varah, *Pitfalls in the numerical solution of linear ill-posed problems*, SIAM J. Sci. Stat. Comput. **4** (1983), 164–176.

[75] C. R. Vogel, *Optimal choice of a truncation level for the truncated SVD solution of linear first kind integral equations when data are noisy*, SIAM J. Numer. Anal. **23** (1986), 109–117.

[76] C. R. Vogel, *Solving ill-conditioned linear systems using the conjugate gradient method*, Report, Dept. of Mathematical Sciences, Montana State University, 1987.

[77] G. Wahba, *Spline Models for Observational Data*, CBMS-NSF Regional Conference Series in Applied Mathematics, Vol. 59, SIAM, Philadelphia, 1990.

[78] G. M. Wing, *Condition numbers of matrices arising from the numerical solution of linear integral equations of the first kind*, J. Integral Equations **9** (Suppl.) (1985), 191–204.

[79] G. M. Wing & J. D. Zahrt, *A Primer on Integral Equations of the First Kind*, SIAM, Philadelphia, 1991.

[80] H. Zha & P. C. Hansen, *Regularization and the general Gauss-Markov linear model*, Math. Comp. **55** (1990), 613–624.