Regularization Using a Parameterized Trust Region Subproblem

Oleg Grodzevich* Henry Wolkowicz†

May 9, 2005

University of Waterloo
Department of Combinatorics & Optimization
Waterloo, Ontario N2L 3G1, Canada
Research Report CORR 2005-11

Keywords: regularization, trust region subproblem, ill-conditioned problems, L-curve, image restoration

Abstract

We present a new method for regularization of ill-conditioned problems, such as those that arise in image restoration or mathematical processing of medical data. The method extends the traditional *trust-region subproblem*, TRS , approach that makes use of the *L-curve* maximum curvature criterion, a strategy recently proposed to find a good regularization parameter. We use derivative information, and properties of an algorithm for solving the TRS, to efficiently move along points on the L-curve and reach the point of maximum curvature. We do not find a complete characterization of the L-curve. A MATLAB code for the algorithm is tested and a comparison to the conjugate gradient least squares, CGLS, approach is given and analyzed.

*University of Waterloo, Department of Combinatorics and Optimization, Waterloo, Ontario, Canada, N2L 3G1. E-mail: ogrodzev@uwaterloo.ca

†Research supported by The Natural Sciences and Engineering Research Council of Canada. Email hwolkowica@uwaterloo.ca

Contents

List of Tables

List of Figures

List of Algorithms

1 Introduction

Regularization centers on finding approximate solutions for least-squares problems such as

$$\min_x \|Gx - d\|_2 , \tag{1}$$

where G is a singular or ill-conditioned *forward operator* and d is a vector of *observed data*. (Here we restrict G to being a square $n \times n$ matrix.) This problem arises from mathematical models $Gx = d$, where the data contains noise η,

$$Gx = Gx_{\text{true}} + \eta = d = d_{\text{true}} + \eta.$$

It is remarkable that, for many applications, a small amount of noise η can result in a solution x that has no relation to x_{true}, i.e. we can make the size of the error $\|\eta\|_2$ arbitrarily small, while the size of the error in the solution $\|x - x_{\text{true}}\|_2$ is arbitrarily large. Moreover, in the G singular case, there can be no solution or an infinite number of solutions x_{true}. (See e.g. the survey article [26] or the book [1].) The least-squares problem (1) typically arises from discretizations of linear equations in infinite dimensional spaces, e.g. $Tx = d$, where T is typically a compact operator and so has an unbounded inverse. This means that x is not a continuous function of the data d. Such problems are called *ill-posed* [14, 15].

To obtain meaningful solutions to the mathematical model one often uses various methods of *regularization*. The aim is to find algorithms for constructing *generalized solutions* that are stable under small changes in the data d. One method uses the solution, $x(\varepsilon)$, of the constrained least-squares problem:

$$r_\varepsilon := r(A, a, \varepsilon) := \begin{array}{ll} \min & \|Gx - d\|_2 \\ \text{subject to} & \|x\|_2 \le \varepsilon. \end{array} \tag{2}$$

The restriction on $\|x\|_2$ results in a larger residual error $\|Gx - d\|_2$ but reduces the propagated data error in $\|x\|_2$. As ε increases we reduce $\|Gx(\varepsilon) - d\|_2$ and expect $x(\varepsilon)$ to approximate the best least-squares solution $x_{\text{true}} = G^\dagger d_{\text{true}}$, where $G^\dagger$ denotes the Moore-Penrose generalized inverse of G. However, in practice the error propagation in $x(\varepsilon)$ stays small for small ε but then eventually causes divergence of the iterates $x(\varepsilon)$ from x_{true}. (See *semiconvergence* in [24].) Regularization depends on choosing the *correct* parameter ε.

By squaring the objective and the constraint, (2) can be reformulated as the so-called *trust region subproblem*, TRS , e.g. [7]:

$$\text{(TRS)} \qquad \mu_\varepsilon := \mu(A, a, \varepsilon) := \begin{array}{ll} \min & q(x) := x^T Ax - 2a^T x \\ \text{subject to} & \|x\|_2^2 \le \varepsilon^2, \end{array}$$

where $A := G^T G$ is $n \times n$ symmetric (we assume $n \geq 2$ and G is nonsingular, though probably ill-conditioned), $a := G^T d$ is an n-vector, ε is a *positive* scalar, and x is the n-vector of unknowns. All matrix and vector entries are real. We let $x^* = x(\varepsilon)$ denote the optimal solution and $\lambda^* = \lambda(\varepsilon)$ denote the corresponding optimal Lagrange multiplier. The relation between optimal values is $r_\varepsilon = \mu_\varepsilon + d^T d$. The TRS can be used to form the so-called *L-curve*,

$$\mathcal{L}(G, d) := \{(\log(\varepsilon), \log \|Gx(\varepsilon) - d\|_2) : \varepsilon > 0,\ x(\varepsilon) \text{ is optimal for TRS } \}, \tag{3}$$

see e.g. Figure 5 below. A strategy introduced recently to find a good (correct) regularization parameter uses the point of maximum curvature on the *L-curve*, e.g. [17].

In this paper, we apply known results for TRS to efficiently control the parameter ε and find regularized solutions of (1). This extends the traditional trust-region approach for regularization of ill-conditioned problems. We show that this is an effective tool that can be used in conjunction with the L-curve maximum curvature criterion. We also compare our approach to the Conjugate Gradient Least Squares method, CGLS, see e.g. [25].

1.1 Outline

In Section 2 we present the basic regularization theory that we need. This includes the analytic description of the L-curve and its curvature. In Section 3 we present the TRS approach for regularization and show how it can be used to efficiently control the regularization parameter using the trust region radius ε. The main result, i.e. the regularization algorithm using differential information for finding the *correct* regularization parameter, is presented in Section 4.

Numerical results are presented in Section 5. In Section 5.2 we consider an image restoration example. Concluding remarks are given in Section 6.

2 Basic Regularization Theory

2.1 Tikhonov Regularization

Regularization dates back to work by Tikhonov [35]. (See also [36].) For the linear equation $Tx = d$, one solves the damped normal equation

$$(T^*T + \alpha^2 I)x_\alpha = T^*d, \tag{4}$$

where T^* denotes the *adjoint* of T and $d = d_{\text{true}} + \eta$. In this paper we restrict our analysis to a finite-dimensional discretization of an operator T, represented by a matrix G. We

replace (4) by

$$(G^TG + \alpha^2 I)x_\alpha = G^T d. \tag{5}$$

Regularization involves choosing the correct value for the parameter α, when given some information on the size of the error η. A Lagrange multiplier argument shows the equivalence between choosing the correct value for α in (4) and choosing the correct value for ε in (2).

2.2 Using Singular Values

The singular value decomposition (SVD) of the matrix G simplifies the understanding of the L-curve analysis (see Section 2.3). We will write the SVD as $G = USV^T$, where matrix S is a diagonal $n \times n$ matrix consisting of singular values σ_i of G, $\sigma_1 \leq \ldots \leq \sigma_n$, and U, V are orthogonal matrices. We can characterize the Tikhonov regularized solution x_α using the SVD in the following way. Substitute the SVD of the matrix G into (5):

$$\begin{aligned} (VSU^TUSV^T + \alpha^2 I)x_\alpha &= VSU^T d \\ V^T x_\alpha &= (S^2 + \alpha^2 I)^{-1} SU^T d. \end{aligned}$$

Using the orthogonality of the matrices U and V with the so-called *Tikhonov filter factors* $f_i = \dfrac{\sigma_i^2}{\sigma_i^2 + \alpha^2}$, we get

$$\begin{aligned} x_\alpha = V(S^2 + \alpha^2 I)^{-1} SU^T d = \textstyle\sum_{i=1}^n f_i \dfrac{U_{:i}^T d}{\sigma_i} V_{:i} \\ d - Gx_\alpha = d - USV^T x_\alpha = U(I - S(S^2 + \alpha^2 I)^{-1} S)U^T d, \end{aligned} \tag{6}$$

where $U_{:i}$ denotes the i^{th} column of U. This implies

$$\begin{aligned} \|x_\alpha\|_2^2 &= \textstyle\sum_{i=1}^n f_i^2 \Big(\dfrac{U_{:i}^T d}{\sigma_i}\Big)^2 \\ \|Gx_\alpha - d\|_2^2 &= \textstyle\sum_{i=1}^n (1 - f_i)^2 \Big(U_{:i}^T d\Big)^2. \end{aligned} \tag{7}$$

We note that $\|x_\alpha\|_2 < \|x_0\|_2$, $\forall \alpha > 0$, with $\|x_\alpha\|_2 \to 0$ as $\alpha \to \infty$. If G is invertible, then setting $\alpha = 0$ gives the unique solution x_0, i.e. $\|Gx_0 - d\|_2 = 0$ as all filter factors are equal to one. Moreover, adding uncorrelated noise η results in

$$\|x_0\|_2^2 = \sum_{i=1}^n \Big(\frac{U_{:i}^T d_{\text{true}}}{\sigma_i} + \frac{U_{:i}^T \eta}{\sigma_i}\Big)^2.$$

These error contributions can be large when we have small singular values and the noise vector is not orthogonal to the corresponding singular vectors, $U_{:i}$'s.

In the case of small singular values, the situation continues to be problematic even if the noise component is absent, i.e. the ratio $\frac{U_{:i}^T d_{\text{true}}}{\sigma_i}$ in (7) implies that we require:

$$\boxed{\text{the Fourier coefficients } |U_{:i}^T d_{\text{true}}| \text{ decay faster than the } \sigma_i} \tag{8}$$

This condition, known as *the Discrete Picard Condition*, e.g. [20], guarantees that the least-squares solution has a reasonable norm and thus is physically meaningful. Example 2.1 and Figure 1 illustrates the difficulties that arise when the Picard condition fails.

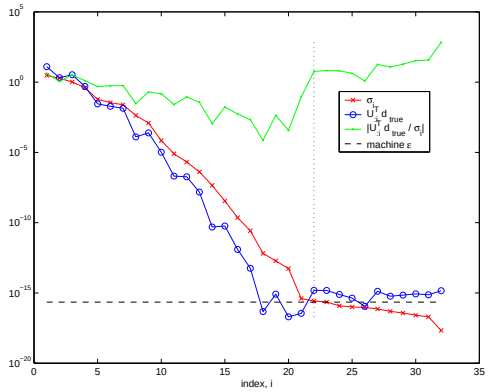

Figure 1: Picard plot for a Shaw problem

Example 2.1. *We consider a Shaw problem from the Hansen MATLAB package (see [18]) with $n = 32$. This is a one-dimensional image restoration problem which is constructed via discretization of a Fredholm integral equation of the first kind (see [31]). The MATLAB* ***shaw*** *command produces the matrix G and the right-hand side vector d_{true}, as well as the true solution vector x_{true}.*

We plot the Fourier coefficients $|U_{:i}^T d_{\text{true}}|$, the singular values σ_i and the ratio $\frac{|U_{:i}^T d_{\text{true}}|}{\sigma_i}$ in Figure 1, marked with o, $\times$, and $-$, respectively. The Picard condition holds until the singular values (line marked with $\times$) reach the machine epsilon level (horizontal dashed line). the Picard condition fails for the larger indices due to round-off error. The norm of the least-squares solution computed via SVD, i.e. by using (7), is $\sim 10^5$; while the true solution has norm ~ 10. A good approximation of the true solution is still recoverable via a truncated SVD, i.e. by setting to 0 all the singular values less than machine epsilon.

Remark 2.1. *Suppose that the singular values can be divided into two sets: the large singular values* $S_L = \{\sigma_i, i = 1 \dots r\}$ *and the small singular values* $S_S = \{\sigma_i, i = r+1 \dots n\}$. *And suppose that the Fourier coefficients* $\{|U_{:,i}^T d_{true}|, i = r+1 \dots n\}$ *are small, essentially zero. Then, by the above argument, we see that the least-squares solution is physically meaningful. It is interesting, as we shall see below, that this corresponds to the so-called* hard case *for TRS.*

2.3 The L-curve analysis

We now study the relationship between the norm of the solution $\|x_\alpha\|_2$ and the norm of the residual $\|Gx_\alpha - d\|_2$. The relationship is plotted as the log-log L-curve described in (3), see e.g. [20]. The curve usually features a strong L-shaped form with almost linear vertical and horizontal parts and a well distinguishable *elbow* or *corner*. (In this paper we use the nonstandard L-curve with the abscissa as $\log(\|x_\alpha\|_2)$.)

Recall the expressions (7) for the norms of the residual and the solution. In the presence of noise η, the latter is

$$\|x_\alpha\|_2^2 = \sum_{i=1}^{n} f_i^2 \Big(\frac{U_{:i}^T d_{\text{true}}}{\sigma_i} + \frac{U_{:i}^T \eta}{\sigma_i} \Big)^2 .$$

If we assume uncorrelated noise, then the expected value of the Fourier coefficients of η are independent of i, $\mathcal{E}\Big(|U_{:i}^T \eta|\Big) \approx \|\eta\|_2, \forall i$. Therefore the Picard condition fails in the presence of noise when there are small singular values.

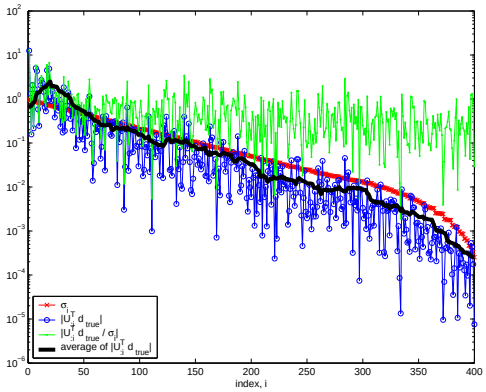

Figure 2: Picard plot for the unperturbed right-hand side

For illustration, we consider a deblurring of a 20×20 image. (See Section 5.2 for problem details.) Figure 2 shows the Picard plot for the unperturbed right-hand side. On average

the Fourier coefficients corresponding to the unperturbed data vector decay faster than the singular values. Hence, the Picard condition holds and the least-squares solution recovers the true solution in the absence of noise. The Fourier coefficients with noise η are plotted

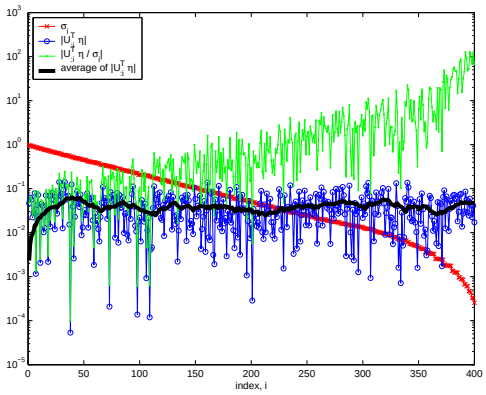

Figure 3: Picard plot for the noise vector

in Figure 3. Now on average they stay on the same level and hence fail to satisfy the Picard condition. As expected, the Picard plot for the perturbed (noisy) right-hand side levels off at approximately $\|\eta\|_2$ as shown in Figure 4.

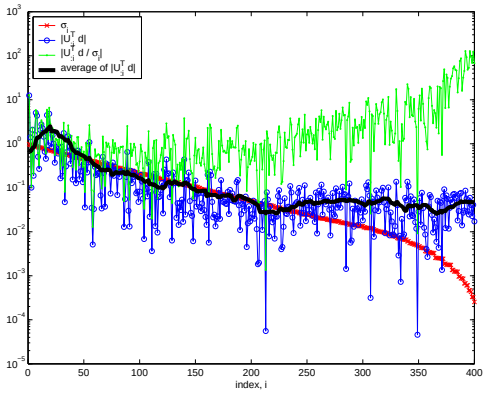

Figure 4: Picard plot for the perturbed right-hand side

Now we no longer restrict $f_i = 1$ and start looking at solutions x_α corresponding to the different values of the regularization parameter α. Since

$$f_i \simeq \begin{cases} 1 & \text{if } \sigma_i \gg \alpha \\ 0 & \text{if } \sigma_i \ll \alpha, \end{cases}$$

The filter factors control which terms in the summation contribute to the norm of the residual and the solution. Figure 5 demonstrates that when the regularization parameter corresponds to the larger singular values (equivalently ε is small), the norm of the residual varies greatly with α, but the norm of the solution is almost unaffected, since all the terms corresponding to the smaller singular values are filtered (this is also known as *oversmoothing* a solution). This situation gives rise to the vertical part of the L-curve. On the other hand, when α is small (or equivalently the trust region radius ε is large), then small changes in α results in small changes in the norm of the residual, but can cause large changes in the norm of the solution. This corresponds to the horizontal part of the L-curve. Depending on the particular Picard plot, the smoothness of the transition between the vertical and horizontal parts can vary in a broad range. For example, the L-curve for the deblurring problem is presented in Figure 5. It is not strongly L-shaped, but it is still possible to locate a distinguishable *elbow*. This discussion is relevant only when the log-log scale is used. In a linear scale the plot is always convex, see e.g. [19].

This behaviour leads to the *L-curve criterion* for choosing the regularization parameter proposed in [17, 21], i.e. one chooses the value of the parameter that corresponds to a point on the L-curve with maximum curvature (details on curvature calculation are given in Sections 2.3.1 and 2.3.2). A point of maximum curvature coincides with an *elbow* that separates the regions where the solution is dominated by regularization errors (oversmoothing) and perturbation errors.

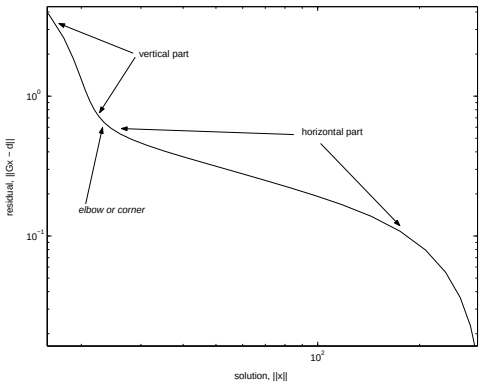

Figure 5: The L-curve for the deblurring problem

2.3.1 Curvature of the L-curve

Following [20], see also [16, 21, 17, 16], let

$$\eta := \|x_\varepsilon\|_2^2, \quad \hat{\eta} := \log \eta; \qquad \rho := \|Gx_\varepsilon - d\|_2^2 = \mu_\varepsilon + d^T d, \quad \hat{\rho} := \log \rho.$$

And, recall the $\lambda^* = \lambda(\varepsilon)$ is the optimal Lagrange multiplier for TRS. The L-curve is a plot of $\hat{\eta}/2$ versus $\hat{\rho}/2$. Then the curvature κ of the L-curve, as a function of ε, is given by

$$\kappa_\varepsilon = 2\frac{\hat{\rho}'\hat{\eta}'' - \hat{\rho}''\hat{\eta}'}{((\hat{\rho}')^2 + (\hat{\eta}')^2)^{3/2}}. \tag{9}$$

Under the assumptions made in Section 3.1, $\eta = \varepsilon^2$, and therefore,

$$\hat{\eta}' = \frac{\eta'}{\eta} = \frac{2}{\varepsilon} \quad \text{and} \quad \hat{\eta}'' = -\frac{2}{\varepsilon^2}.$$

Furthermore,

$$\hat{\rho}' = \frac{\rho'}{\rho} = \frac{\mu_\varepsilon'}{\mu_\varepsilon} \quad \text{and} \quad \hat{\rho}'' = \frac{\mu_\varepsilon''\mu_\varepsilon - (\mu_\varepsilon')^2}{\mu_\varepsilon^2}.$$

Substituting these expressions into (9) we get

$$\begin{aligned} \kappa_\varepsilon &= 2\Big(-\tfrac{\mu_\varepsilon'}{\mu_\varepsilon}\tfrac{2}{\varepsilon^2} - \tfrac{\mu_\varepsilon''\mu_\varepsilon - (\mu_\varepsilon')^2}{\mu_\varepsilon^2}\tfrac{2}{\varepsilon}\Big)\Big(\big(\tfrac{\mu_\varepsilon'}{\mu_\varepsilon}\big)^2 + \big(\tfrac{2}{\varepsilon}\big)^2\Big)^{-3/2} \\ &= 4\varepsilon\mu_\varepsilon\Big(\varepsilon(\mu_\varepsilon')^2 - \mu_\varepsilon\mu_\varepsilon' - \varepsilon\mu_\varepsilon\mu_\varepsilon''\Big)\Big(\varepsilon^2(\mu_\varepsilon')^2 + 4\mu_\varepsilon^2\Big)^{-3/2} \\ &= \varepsilon^2\mu_\varepsilon\Big(2\varepsilon^2\lambda^{*2} - 2\mu_\varepsilon\lambda^* - \varepsilon\mu_\varepsilon\big(\tfrac{\partial\lambda^*}{\partial\varepsilon}\big)\Big)\Big(\varepsilon^4\lambda^{*2} + \mu_\varepsilon^2\Big)^{-3/2}. \end{aligned} \tag{10}$$

The last equality follows from (15) and (16) derived below.

2.3.2 Curvature Estimation and Gauss Quadrature

The numerical evaluation of the curvature (10) requires the (expensive) derivative $\frac{\partial\lambda^*}{\partial\varepsilon} = \varepsilon/\left(a^T(A-\lambda^* I)^{-3}a\right)$, see (14) below. This issue is addressed in [10, 11, 2, 12]. One approach lies in obtaining both upper and lower bounds

$$l_p(\alpha) \le \nu_p(\alpha) = d^T G(G^T G + \alpha I)^p G^T d \le u_p(\alpha),$$

where $\alpha = -\lambda^*$ is a positive scalar, p is a negative integer ($p = -3, G^T G = A$ and $G^T d = a$ in (14)). These bounds are obtained using an iterative procedure and become tighter as the number of iterations, k, increases. We briefly outline the idea.

After k iterations applied to G, the Lanczos Bidiagonalization algorithm (e.g. [13]) produces a $(k+1)$-by-k lower bidiagonal matrix

$$B_k = \begin{bmatrix} \gamma_1 & & & \\ \delta_1 & \ddots & & \\ & \ddots & \ddots & \\ & & \ddots & \gamma_k \\ & & & \delta_k \end{bmatrix}, \text{ with } GV = VB_k,\ V = \begin{bmatrix} v_1 & \dots & v_k \end{bmatrix},$$

such that the Gauss and Gauss-Radau quadrature rules for $\nu_p(\alpha)$ are defined as

$$l_p(\alpha) = \left\|G^T d\right\|_2^2 e_1^T (B_k^T B_k + \alpha I)^p e_1 = \|d\|_2^2 e_1^T B_k (B_k^T B_k + \alpha I)^p B_k^T e_1, \tag{11}$$

$$u_p(\alpha) = \left\|G^T d\right\|_2^2 e_1^T (\tilde{U_k}^T \tilde{U_k} + \alpha I)^p e_1. \tag{12}$$

Here $\tilde{U_k}$ is $(k+1)$-by-k upper bidiagonal matrix obtained from B_k by a sequence of Givens rotations and by setting the main diagonal to zero.

Our implementation of the Lanczos Bidiagonalization algorithm allows restarting from the specified iteration (with v_k) if optional parameters are supplied. This enables one to increase the precision when necessary. This feature is exploited by the main algorithm that iterates by gradually decreasing α. Since for $p < 0$ and non-singular B_k we have that:

$$\lim_{\alpha \searrow 0} l_p(\alpha) < \infty, \quad \lim_{\alpha \searrow 0} u_p(\alpha) = \infty,$$

it is natural that the bounds weaken as $\alpha \searrow 0$, i.e. we need to increase the precision.

Note that evaluating the expressions $G_p(\alpha)$ and $R_p(\alpha)$ implies solving linear systems:

$$(\tilde{U_k}^T \tilde{U_k} + \alpha I)x = e_1,$$
$$(B_k^T B_k + \alpha I)x = B_k^T e_1.$$

The above equations are the normal equations for the linear least-squares problem, LLS ,

$$\min \left\| \begin{bmatrix} \tilde{U_k} \\ \sqrt{\alpha} I \end{bmatrix} x - \begin{bmatrix} 0 \\ e_1/\sqrt{\alpha} \end{bmatrix} \right\|_2$$
$$\min \left\| \begin{bmatrix} B_k \\ \sqrt{\alpha} I \end{bmatrix} x - \begin{bmatrix} e_1 \\ 0 \end{bmatrix} \right\|_2.$$

This means that the solution x for the linear least-squares problem satisfies the original linear system as well. We may, however, exploit the structure of LLS problems and solve them efficiently by a sequence of Givens rotations that produces the QR factorization. This approach is described in [6, 8, 38].

3 Regularization Using TRS

In this section, we recall some of the details in the Rendl-Wolkowicz TRS algorithm, [28, 7], denoted RW , and apply them to the regularization problem. We show that the RW algorithm visits a point on the L-curve at each iteration and that the curvature of the L-curve can be efficiently computed for each such point. Therefore, we can modify the radius, ε, of the trust region to steer the algorithm to the elbow of the L-curve.

3.1 The Optimality Conditions

It is known ([9, 32]) that x^* is a solution to TRS if and only if:

$$\begin{array}{cc} \left.\begin{array}{c}(A-\lambda^* I)x^* = a, \\ A-\lambda^* I \succeq 0, \lambda^* \leq 0\end{array}\right\} & \text{dual feasibility} \\ \|x^*\|_2^2 \leq \varepsilon^2 & \text{primal feasibility} \\ \lambda^*(\|x^*\|_2^2 - \varepsilon^2) = 0 & \text{complementary slackness} \end{array} \tag{13}$$

for some (Lagrange multiplier) λ^*. The above conditions connect Tikhonov regularization with TRS , i.e. solving (5) with a particular value of the regularization parameter α is equivalent to solving (2) with a corresponding value of ε. Also, for our applications $\lambda^* < 0 \leq \lambda_1(A)$. Therefore, the optimal solution always lies on the boundary, $\|x^*\|_2 = \varepsilon$, and the so-called *easy case* holds for TRS .

3.2 Derivatives of μ and λ^*

We keep the data A, a fixed and consider the optimal value as a function of $\varepsilon > 0$. By abuse of notation, we write $\mu_\varepsilon = \mu(A, a, \varepsilon)$. We assume (as discussed in Section 3.1) that the *easy case* holds, and that the optimum point lies on the boundary of the feasible region, i.e. $\|x^*\|_2 = \varepsilon$.

The derivative $\frac{\partial \lambda^*}{\partial \varepsilon}$ can be found using implicit differentiation on the equation $\|(A-\lambda^* I)^{-1}a\|_2^2 - \varepsilon^2 = 0$, obtained after the substitution $x^* = (A-\lambda^* I)^{-1}a$, i.e.

$$\begin{array}{rcl} a^T(A-\lambda^* I)^{-2}a & = & \varepsilon^2 \\ 2\left(\frac{\partial \lambda^*}{\partial \varepsilon}\right) a^T(A-\lambda^* I)^{-3}a & = & 2\varepsilon \end{array}$$

and

$$\frac{\partial \lambda^*}{\partial \varepsilon} = \frac{\varepsilon}{a^T(A-\lambda^* I)^{-3}a}. \tag{14}$$

Moreover,

$$\begin{aligned}
\mu_\varepsilon &= (x^*)^T A x^* - 2a^T x^* \\
&= (x^*)^T A x^* - 2a^T x^* - \lambda^*(\|x^*\|_2^2 - \varepsilon^2) \\
&= (x^*)^T (A - \lambda^* I) x^* - 2a^T x^* + \lambda^* \varepsilon^2 \\
&= a^T (A - \lambda^* I)^{-1} a - 2a^T (A - \lambda^* I)^{-1} a + \lambda^* \varepsilon^2 \\
&= -a^T (A - \lambda^* I)^{-1} a + \lambda^* \varepsilon^2.
\end{aligned}$$

Then, using $a^T (A - \lambda^* I)^{-2} a - \varepsilon^2 = \|x^*\|_2^2 - \varepsilon^2 = 0$, we get

$$\begin{aligned}
\frac{\partial \mu_\varepsilon}{\partial \varepsilon} &= a^T (A - \lambda^* I)^{-2} a \left(-\tfrac{\partial \lambda^*}{\partial \varepsilon}\right) + \left(\tfrac{\partial \lambda^*}{\partial \varepsilon}\right) \varepsilon^2 + 2\lambda^* \varepsilon \\
&= \left(-\tfrac{\partial \lambda^*}{\partial \varepsilon}\right) \left(a^T (A - \lambda^* I)^{-2} a - \varepsilon^2\right) + 2\lambda^* \varepsilon \\
&= 2\lambda^* \varepsilon
\end{aligned} \tag{15}$$

and

$$\frac{\partial^2 \mu_\varepsilon}{\partial \varepsilon^2} = 2\left(\lambda^* + \varepsilon \frac{\partial \lambda^*}{\partial \varepsilon}\right). \tag{16}$$

More details on these and other perturbation results can be found in [34].

3.3 Building the L-curve using TRS

If the optimum of TRS is on the boundary, then the objective function of TRS and ε correspond to a unique point on the L-curve, and vice-versa. Moreover, each step of the

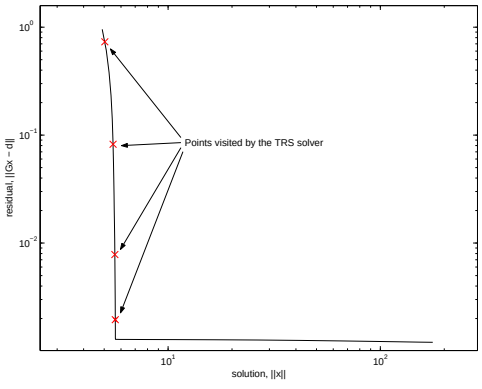

Figure 6: Points encountered while solving TRS

TRS algorithm produces a solution x that is optimal to TRS with a different, but close,

trust region radius ε, i.e. the TRS algorithm finds points on the L-curve. E.g. Figure 6 presents an L-curve for a sample Shaw problem created using the Hansen MATLAB package (see [18]). The TRS algorithm was then executed with the generated data and a fixed trust region radius $\varepsilon = 6$. (The TRS optimum yields a point near the *elbow*). It took 8 iterations to solve the trust region subproblem with a desired optimality tolerance $\delta = 10^{-8}$. We obtained four points on the L-curve. (The other four are located outside the interval of uncertainty.) These four points give enough information to approximate the vertical part of the L-curve to the left of the *elbow*.

We will modify the TRS solver to control the trust region radius as we iterate towards the solution so that each point that we find on the L-curve helps in locating the *elbow*.

We continue with results from the RW algorithm. By exploiting the strong Lagrangian duality of TRS (see [33]), TRS can be reformulated as an unconstrained concave maximization problem, i.e.

$$\mu_\varepsilon = \min_x \max_{\lambda \leq 0} L(x, \lambda) = \max_{\lambda \leq 0} \min_x L(x, \lambda),$$

where $L(x, \lambda) = x^T A x - 2a^T x + \lambda(\varepsilon^2 - \|x\|_2^2)$ denotes the *Lagrangian* of TRS . Define the symmetric $(n+1) \times (n+1)$ matrix

$$D(t) = \begin{bmatrix} t & -a^T \\ -a & A \end{bmatrix}, \tag{17}$$

and let $\lambda_1(D(t))$ denotes its smallest eigenvalue. Further define the concave function

$$k(t) = (\varepsilon^2 + 1)\lambda_1(D(t)) - t, \ t \in \mathbb{R}, \tag{18}$$

Then an unconstrained dual problem is given by

$$\mu_\varepsilon = \max_t k(t). \tag{19}$$

Furthermore, under assumptions of the *easy case*, $\lambda_1(D(t))$ is a singleton eigenvalue, and the derivative of $k(t)$ satisfies

$$k'(t) = (\varepsilon^2 + 1)y_0^2 - 1, \tag{20}$$

where $\binom{y_0}{x}$ is the normalized eigenvector for $\lambda_1(D(t))$, scaled so that $y_0 \geq 0$. (Note $y_0 > 0$ in the easy case.)

3.4 Regularization as a one-dimensional parameterized problem

The L-curve is formed using ε in TRS as a parameter and finding the residual for the corresponding optimal $x(\varepsilon)$. We now see that the L-curve can be formed using any of the following parameters:

t – control parameter in $k(t), D(t)$
ε – trust-region radius, norm of the solution $\|x(\varepsilon)\|_2$
α – Tikhonov regularization parameter
λ – optimal Lagrange multiplier for TRS

From Section 2.1, we have $\lambda = -\alpha^2$. However, changing between λ, t and ε is computationally expensive. the following lemmas describe some of the relationships.

Lemma 3.1. *Given the parameter $\lambda < 0$, the corresponding values of t and ε are given by*

$$\begin{array}{rcll} t & = & \lambda & + \quad d^T G(G^T G - \lambda I)^{-1} G^T d \\ \lambda_1(D(t)) & = & \lambda & \\ \varepsilon^2 & = & & d^T G(G^T G - \lambda I)^{-2} G^T d \end{array} \tag{21}$$

Proof: The formula for t follows from Proposition 3.1 and Corollary 3.4 in [28]. The formula for ε follows from the optimality conditions (13), since the optimal solution x^* to TRS , that corresponds to the Lagrange multiplier $\lambda^* = \lambda$, lies on the boundary. ■

Lemma 3.2. *Given the parameter $t < d^T d$, the corresponding values of λ and ε are given by*

$$\begin{array}{rcl} \lambda & = & \lambda_1(D(t)) \\ \varepsilon^2 & = & \dfrac{1 - y_0(t)^2}{y_0(t)^2}, \end{array} \tag{22}$$

where $y(t)$ is the eigenvector corresponding to $\lambda_1(D(t))$ and $y_0(t)$ is its first component.

Proof: The results follow from Theorem 3.7 in [28], e.g. we use the normalized eigenvector $\begin{pmatrix} y_0 \\ x \end{pmatrix}$ and find that $\frac{1}{y_0} x$ is a solution for TRS. This implies

$$y_0^2 + \|x\|_2^2 = 1, \quad \|x\|_2 = y_0 \varepsilon.$$

■

Corollary 3.1. *The derivatives*

$$\lambda' = \frac{d\lambda}{dt} = (1+\varepsilon^2)^{-1} > 0 \tag{23}$$

$$\varepsilon' = \frac{d\varepsilon}{dt} = \frac{-(1+\varepsilon^2)}{2\varepsilon}\frac{\lambda''}{\lambda'} = \frac{1}{\varepsilon}\sum_{s\neq 1}\frac{y_s^2}{\lambda_s - \lambda} > 0, \tag{24}$$

where y_s, λ_s and the formula for λ'' are given in (25) *below.*

Proof: We use (20) and (22). The derivative of the smallest eigenvalue (see e.g. [27])

$$\begin{aligned}
\tfrac{d\lambda}{dt} &= \tfrac{d\lambda_1(D(t))}{dt} \\
&= \left(\begin{smallmatrix} y_0 \\ x \end{smallmatrix}\right)^T \tfrac{dD(t)}{dt} \left(\begin{smallmatrix} y_0 \\ x \end{smallmatrix}\right) \\
&= \left(\begin{smallmatrix} y_0 \\ x \end{smallmatrix}\right)^T \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \left(\begin{smallmatrix} y_0 \\ x \end{smallmatrix}\right) \\
&= y_0^2.
\end{aligned}$$

This yields (23).

For a given ε, we have $k'(t) = (\varepsilon^2+1)\frac{d\lambda}{dt} - 1 = 0$ at the optimum t. We can differentiate both sides with respect to t and solve for $\frac{d\varepsilon}{dt}$ to get (24). A formula for λ'' can be found in [22, 27, 23], i.e. with $v_s = \left(\begin{smallmatrix} y_s \\ x_s \end{smallmatrix}\right)$, denoting the other normalized eigenvectors of $D(t)$ for eigenvalues $\lambda_s,\ s \neq 1$, and using $\frac{d^2D(t)}{dt^2} = 0$, we get

$$\lambda'' = 2\sum_{s\neq 1}\frac{(y_s y_0)^2}{\lambda - \lambda_s} = -2\sum_{s\neq 1}\frac{(y_s)^2}{(1+\varepsilon^2)(\lambda_s - \lambda)}. \tag{25}$$

■

Lemma 3.3. *Given the parameter $\varepsilon < \|G^{-1}d\|_2$, the corresponding values of t and λ can be obtained by solving* TRS *using the* RW *algorithm. The corresponding optimal solution satisfies $\|x(\varepsilon)\|_2 = \varepsilon$.*

Proof: The RW algorithm solves TRS with a fixed trust region radius ε producing the optimal solution $x(\varepsilon)$, the optimal Lagrange multiplier λ^*, and the corresponding parameter t. And, $\|x(\varepsilon)\|_2 = \varepsilon$, since the norm of the unconstrained minimum $\|x^*\|_2 = \|A^{-1}a\|_2 = \|G^{-1}d\|_2$ ■

Combining the above lemmas we conclude that each of t, λ, ε, α can be interchangeably used to parameterize the regularization problem.

3.5 Intervals of interest for t, λ and ε

The interval of uncertainty for the Tikhonov regularization parameter is $0 \leq \alpha^2 < \infty$. Using the results in Section 3.4, this corresponds to the following.

Corollary 3.2. *The intervals of uncertainty for the parameters are:*

$$\begin{array}{rcll} -\infty & < & \lambda = \lambda_1(D(t)) = -\alpha^2 & \leq & 0 \\ 0 & < & t = \lambda + d^T G(G^T G - \lambda I)^{-1} G^T d & \leq & \|d\|_2^2 \\ 0 & < & \varepsilon = \left\|(G^T G - \lambda I)^{-1} G^T d\right\|_2 & \leq & \|G^{-1} d\|_2 \end{array}$$

The upper bounds correspond to the linear least squares solution.

Proof: Follows directly from Lemmas 3.1 and 3.2. ■

Note that if the largest singular value σ_n of the matrix G is known, the results of Section 2.2 imply that $-\sigma_n^2$ is a lower bound on λ.

4 Regularization Algorithm

Before presenting the details of the algorithm we state our assumptions and present more geometry and relations among the various parameters. The key assumption is that the values of the parameters are in known bounded intervals of uncertainty, as described in Section 3.5.

First, we observe that the norm of the regularized solution is a monotonic function of both t and λ.

Lemma 4.1. $\|x(t)\|_2$ *and* $\|x(\lambda)\|_2$ *are monotonically increasing functions of* t *and* λ*, respectively.*

Proof: The lemma follows from Theorem 3.7 in [28]. ■

Throughout the rest of the paper we will interchangeably use parameters ε, t and λ when describing points on the L-curve. Hence, if a statement is true for *smaller or larger values of* ε, it is also true for, respectively, smaller or larger values of t and λ.

From a given t, we can calculate the corresponding ε and the value of the objective function $\mu_\varepsilon = k(t)$, thus obtaining a point on the L-curve. To analyze the location of a given pair

$(\varepsilon,\ \mu_\varepsilon)$ on the L-curve, we need the derivative of $l_r(\varepsilon) := \log(\|Gx(\varepsilon) - d\|_2)$ with respect to $l_x(\varepsilon) := \log(\|x(\varepsilon)\|_2)$, i.e.

$$\begin{aligned}\frac{\partial(l_r(\varepsilon))/\partial(\varepsilon)}{\partial(l_x(\varepsilon))/\partial(\varepsilon)} = \frac{\partial(\log(\|Gx-d\|_2))/\partial(\varepsilon)}{\partial(\log(\|x\|_2))/\partial(\varepsilon)} &= \frac{1}{2}\frac{\partial(\log(\mu_\varepsilon + d^T d))/\partial(\varepsilon)}{\partial(\log(\varepsilon))/\partial(\varepsilon)} \\ &= \frac{1}{2}\frac{\mu'_\varepsilon \varepsilon}{\mu_\varepsilon + d^T d} \\ &= \frac{\varepsilon^2 \lambda_\varepsilon}{\mu_\varepsilon + d^T d},\end{aligned} \tag{26}$$

where μ'_ε is found using (15).

To distinguish whether a point lies before (left) or after (right) the elbow, one can test the value of the derivative. It should be (negative) close to zero if we are at the plateau after the elbow. Alternatively, the value tends to a large negative number as we approach the elbow from the left.

4.1 Initial L-curve point

Each iterate of our algorithm increases the value of t, i.e. subsequent points are located to the right of previous ones. Hence, locating the elbow of the L-curve means we start to the left of it. One way is to start with the point corresponding to $\lambda = -\sigma_n(G)^2$, see Sections 2.2 and 3.5.

In the case we do not have the *largest* singular value of the matrix G, we start with a point associated with small enough value of $t = \frac{d^T d}{2}$. As discussed in Section 2.3, a "well-shaped" L-curve plot can be viewed as a linear plateau to the right of the *elbow* and a linear vertical part to the left of the *elbow*. For well shaped L-curve plots, small changes in t would result in large changes in ε when we are on the horizontal part. Conversely, large changes in t result in smalll changes in ε when we are on the vertical part. This is explained by the structure of the singular value decomposition of the matrix G (see Section 2.2). The behaviour remains true for less well-behaved L-shaped plots. This tells us that points that lie on the plateau region correspond to the values of t that are very close to $d^T d$. Thus, taking half of this value will put us onto the vertical part to the left of the *elbow*.

4.2 Outline of the algorithm

We now describe the details behind the implementation. It can be divided into three parts: *initialization*, *main loop* and *final solution refinement*.

Algorithm 1: Trust-Region Based Regularization [**initialization**]

1 compute the *largest* singular value σ_n of the matrix G
2 compute the initial bidiagonalization (γ, δ) of the matrix G using Lanczos Bidiagonalization algorithm; use d as the starting vector.
3 $t_{low} = 0$
4 $t_{up} = d^T d$
5 $\lambda_{low} = -\sigma_n^2$
6 $\lambda_{up} = 0$
7 $\varepsilon_{up} = -1$
8 $\kappa_{low}^{previous} = \infty$
9 $\kappa_{up}^{previous} = \infty$
10 $\lambda = \lambda_{low}$
11 find starting L-curve point parameters $[t, x, k]$ using (21)

The initialization starts with the computation of the *largest* singular value of G. This step is not essential, but is inexpensive and yields a lower bound on the eigenvalue λ. If this step is omitted, then a reasonable value for the parameter t, e.g. $\frac{d^T d}{2}$, can be used. This also yields a lower bound on the eigenvalue, see Lemma 3.2.

The more important step is to compute the initial bidiagonalization of the matrix G. This data is used to estimate the curvature of the L-curve every time a point is obtained. The details are covered in Section 2.3.2.

We then proceed by getting an initial point on the L-curve. The discussion on getting a good estimate is in Section 4.1. We assume that we know the *largest* singular value of G and thus start with a value on a parameter λ. Hence, to locate a point on the L-curve, we solve for values t, x and k.

Algorithm 2: Trust-Region Based Regularization [**main loop**]

while $\lambda < \lambda_{up} - 10^{-10}$ **do**
  ♯ *calculate the slope of the L-curve* (26) *and* $\frac{d\lambda}{dt}$ (23)
    $\varepsilon^2 = x^T x$, $res^2 = k + d^T d$
    $L_{slope} = \lambda\varepsilon^2 / res^2$
    $\frac{d\lambda}{dt} = (1+\varepsilon^2)^{-1}$
    save current point to the solutions history
    $t_{low} = t$, $\lambda_{low} = \lambda$
    $[\kappa_{low}, \kappa_{up}] =$ **curvature**$(\varepsilon, res^2, \lambda)$
  ♯ *termination criteria*
  while *curvature value is not certain* **do**
    if $\kappa_{low} > \kappa_{up}^{previous}$ **then**
      DONE, proceed to the final solution refinement
    end
    if $\kappa_{up} < \kappa_{low}^{previous}$ **then**
      $\kappa_{low}^{previous} = \kappa_{low}$
      $\kappa_{up}^{previous} = \kappa_{up}$
      curvature value is now specified, **break**
    else
      update bidiagonalization (γ, δ) of G to improve precision
      $[\kappa_{low}, \kappa_{up}] =$ **curvature**$(\varepsilon, res^2, \lambda)$
      recalculate bounds on $\kappa^{previous}$
    end
  end
  ♯ *update t*
    $\varepsilon_{target} = \varepsilon$
    perform triangle interpolation on the $k(t)$ to get an estimated t for ε_{target}
    find next L-curve parameters $[t, x, k]$ using (21)
end

The *curvature function* $[\kappa_{low}, \kappa_{up}] =$ **curvature** $(\varepsilon, res, \lambda)$ is found by computing lower and upper bounds on the curvature using the current Lanczos bidiagonalized approximation, see Section 2.3.2.

At each iteration the algorithm takes the current point and produces the next one strictly to the right on the L-curve. There are several possible strategies to achieve this goal. As

Lemma 4.1 suggests, increasing either one of the parameters t, λ or ε will move us further to the right. We can compute a step in any of the spaces associated with these parameters. The hard part, though, is the strategy on choosing the step length. Even a tiny change in t will lead to the huge change in ε. Changing ε alone involves solving TRS , which we try to avoid. Instead of relying just on a single parameter, we employ their combination to compute the step length in the following manner.

The key idea lies in the properties of the function $k(t)$. Recall from Section 3.3, that $k(t) = (\varepsilon^2 + 1)\lambda - t$ and $\mu_\varepsilon = \max_t k(t)$. The function $k(t)$ is also concave. Given the current iterate t_c and the corresponding ε_c we consider the function $k(t)$ with $\varepsilon = \varepsilon_c$, which attains its maximum at the point $t = t_c$. Consider also the point $t_{up} = d^T d$. From Section 3.5 we know that $k(t_{up}) = -t_{up} = -d^T d$. Moreover, the derivative is $k'(t_{up}) = -1$ and does not depend on ε. At every iteration we have $t_c < t_{up}$.

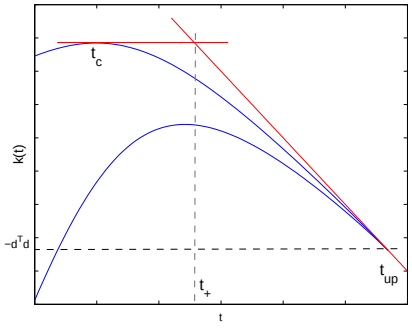

Figure 7: k(t) and triangle interpolation

In Figure 7 we see the main step of the algorithm. First, we use the tangent lines to the current curve $k(t)$ at the current point t_c and at the upper bound point t_{up}. The intersection of these two tangent lines yields a new point t_+. This new point t_+ corresponds to an eigenvalue $\lambda_1(D(t_+))$ and a trust region radius $\varepsilon = \|x(\varepsilon)\|_2$. This new trust region radius ε gives us the new curve $k(t)$ lying below the previous curve. This technique gives quite good results due to the fact that $k(t)$ is almost linear on both sides of the maximum. Moreover, as λ gets closer to $\lambda_1(A)$, the curve becomes linear to the right of t_c with slope -1.

Suppose t_+ is found using the triangle interpolation above. Since the slope at t_{up} is -1,

we have

$$\begin{aligned} t_{up} - t_+ &= k(t_c) - k(t_{up}) \\ t_+ &= t_{up} - k(t_c) + k(t_{up}) \\ t_+ &= -k(t_c). \end{aligned}$$

This gives an explicit expression for the iteration

$$t_+ = t_c - (\varepsilon^2 + 1)\lambda_1(D(t_c)). \tag{27}$$

Suppose that we have convergence in t. If we take the limit on both sides of (27), then t cancel on both sides and we get $0 = (\varepsilon^2 + 1)\lambda_1(D(t_*))$, i.e. the only limit point t^* satisfies $\lambda_1(D(t_*)) = 0$ yielding $t^* = t_{up} = d^T d$. Therefore, the iteration cannot terminate prematurely and $t_+ > t_c$ at each iteration. So we only have to worry about stepping too far, in which case we backtrack appropriately.

With the new t_+ we can compute both λ and ε as Lemma 3.2 suggests. Particular details on the eigenvalue computation are given in Section 5.1.

The termination condition is based on the L-curve maximum curvature criterion, i.e. we look for a point on the L-curve that has the maximum negative curvature. To locate such a point we compute the curvature at each step. This computation uses the Gauss Quadrature approach which is described in Section 2.3.2. Since we are only getting lower and upper bounds on the real value of the curvature, it can be difficult to compare values for two consecutive points. If such a situation is detected, we improve the bidiagonalization of the matrix G. This increases the precision in the curvature estimation and, eventually, allows safe comparison between the curvature at these two points. Since the L-curve is a convex function near the *elbow*, we can determine the area of interest by keeping track of the curvature. Once we get the point with a smaller curvature than the previous one, we know we have gone too far. The main loop of the algorithm terminates once we have 3 points, such that the middle one has a larger curvature value than the other two. From the convexity we deduce that the *elbow* should lie somewhere between the endpoints. The final refinement step is then performed to estimate the *elbow* location.

Algorithm 3: TRS Based Regularization [**final solution refinement**]

```
♯ observe interval of last three points left, center, right
while point of maximum curvature still can be improved do
    set λ as bisection of either left or right interval
    find corresponding L-curve point, [t, x, k] = l2t(λ)
    ♯ calculate norm of the residual and ε
    ε² = xᵀx
    res² = k + dᵀd
    [κ_low, κ_up] = curvature(ε, res², λ)
    if located point has larger curvature then
        set current point as a solution
        DONE
    end
    shrink interval of uncertainty
end
```

To estimate the *elbow* location, we proceed with a simple bisection of the left and right intervals trying to find a point with the maximum curvature value. We stop once we have got a point with a curvature value larger than the one we have seen so far.

5 Numerics

5.1 Eigensolver issues

As shown above, obtaining a new L-curve point means solving for the smallest eigenpair of the matrix $D(t)$. In the case G is large and sparse, the same is true for $D(t)$, so one should use matrix-free iterative algorithms to compute the eigenpairs, e.g. Lanczos methods. As t increases, the smallest eigenvalue may become numerically closer to the second one. This can substantially slow down the eigensolver.

Under such numerical degeneracy an algorithm may converge to a wrong eigenpair, giving an incorrect eigenvector and an incorrect regularized solution. One way to control the eigensolution is to start with an initial eigenvalue smaller than the estimated one and, at the same time, relatively close to it. For iterative algorithms, it is possible to store previous eigenvalue results to re-use on the next step as an initial guess. This works only if the eigenvalue is about to increase at every subsequent iteration.

We have employed this method in our Regularization Algorithm and it proved to be very efficient. We have used the MATLAB **eigs** routine which uses a Lanczos-type matrix-free algorithm. With a good initial eigenvalue guess, this method computes eigenvalues in time independent of the gap between the first and second eigenvalues.

Another approach that can be used is to apply a spectral transformation to separate the first and second eigenvalues, i.e. preconditioning. In particular, a Tchebyshev polynomial transformation is discussed in [29] and [30].

5.2 Image deblurring example

We demonstrate how the algorithm works by considering a sample problem of deblurring an image. Problems of this nature occur often. For instance, one might need to deblur a photo taken by a space telescope or a satellite, see e.g. the forthcoming book [37].

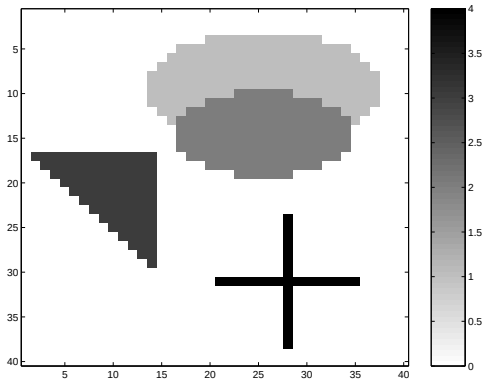

Figure 8: Image deblurring example: original picture

For this particular example we take an image generated by the Hansen MATLAB package ([18]). This package provides an excellent set of regularization tools that can be used for demonstration purposes. Figure 8 shows the image generated by the **blur** command. This command also produces the blurring matrix G and the right-hand side d, i.e. observed data, computed as $d = Gx_{\text{true}} + \eta$. Where η represents the noise. Figure 9 shows the observed image.

The generated image is a 40-by-40 grayscale picture, which is stored as a vector x_{true} of size 1600. This vector is formed by stretching the image matrix into a single column. Every component x^i_{true} represents the brightness of the pixel, measured from 0 for the white to 3 for the black (see the colorbar). The matrix G stands for the operator that represents

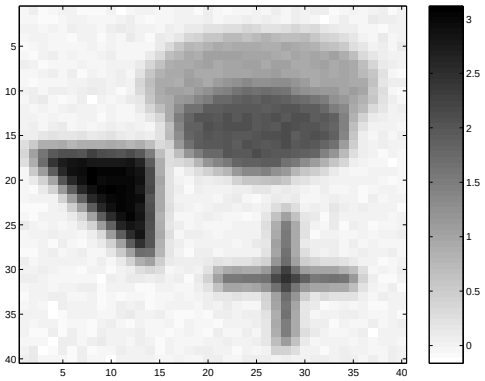

Figure 9: Image deblurring example: observed data, blurred with added noise

degradation of the image caused by atmospheric turbulence blur, modelled by a Gaussian point-spread function,

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

Taking a symmetric banded Toeplitz matrix T with the first row:

$$z_i = \begin{cases} e^{-(i-1)^2/2\sigma^2} & 1 \le i \le b, \\ 0 & b < i \le 40 \end{cases}$$

$$T = \begin{bmatrix} z_1 & z_2 & \cdots\cdots & z_{40} \\ z_2 & z_1 & z_2 & \cdots & z_{39} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ z_{40} & z_{39} & z_{38} & \cdots & z_1 \end{bmatrix},$$

matrix G is constructed as $G = (2\pi\sigma^2)^{-1}T \otimes T$, where $\otimes$ denotes the Kronecker product.

Here parameter σ controls the smoothness (by defining the shape of the Gaussian point spread), and b stands for the bandwidth. Since only non-zero elements are within a distance $b-1$ from the diagonal of the matrix T, it can be stored in a sparse format. It also follows that matrix G is sparse. Hence, we have an example of a large sparse problem.

For our example we fix the parameters to be $\sigma = 1,\ b = 5$. Noise η has a normal distribution with the mean of 0 and the standard deviation of 0.05.

Before running the algorithm, we construct the L-curve to get an idea where the solution is located. We can see that the curve is not strongly L-shaped, but we can still distinguish both vertical and horizontal parts. (See Figure 10.) We also build a plot (dashed line) that

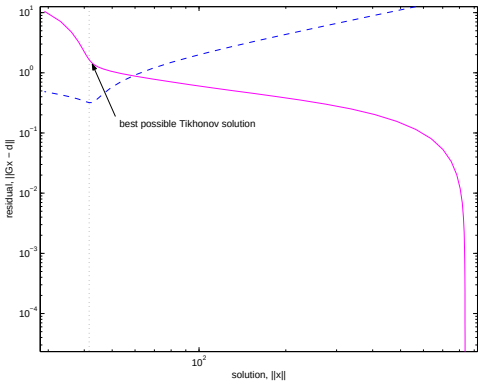

Figure 10: Image deblurring example: corresponding L-curve

shows how well the points on the L-curve approximates the true solution, i.e. for every point x we determine the quantity $\frac{\|x_{\text{true}} - x\|_2}{\|x_{\text{true}}\|_2}$ which we treat as the relative accuracy; the smaller the value, the better the approximation we obtain. The minimum corresponds to the best possible solution that can be obtained using the Tikhonov regularization approach.

Then we run the RPTRS algorithm, see Figure 11. For each point it visits, we present the associated solution image. See Figures 16, 17, 18, 19, 20, 21. We can follow how the solution transforms as we go along the curve. For smaller values of the paramter t the solution appears to be very smooth. The noise components are almost eliminated for these solutions. However, as we increase the regularization parameter, the noise starts to evolve. At the same time, pictures become sharper and represent a better approximation to the true solution. This behaviour continues until we hit the point #5 (see Figure 20). Suddenly, the noise components overcome the real signal and the solution becomes less distinguishable. Finally, the situation becomes even worse at the last point. The least-squares solution consists mostly of the noise components and contains practically no signal information.

The algorithm, however, observes the changes in the curvature value and backtracks, trying to locate the *elbow*. Figure 11 demonstrates the steps that are taken. Points marked with x (cross) are those visited during the main loop, and circles denote the final refinement steps. The algorithm terminates after locating the point closest to the point of the largest curvature (on the convex part). This point is returned as the solution. Note the proximity to the best possible Tikhonov solution. The final RPTRS solution is shown on Figure 12.

For more information and techniques on image de-blurring problems, we note the ongoing research based on wavelets (see e.g. [3, 4, 5]). We do not perform any comparison with

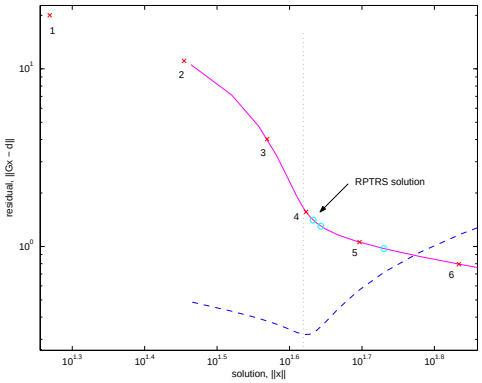

Figure 11: Image deblurring example: corresponding L-curve with RPTRS points

these techniques in this paper.

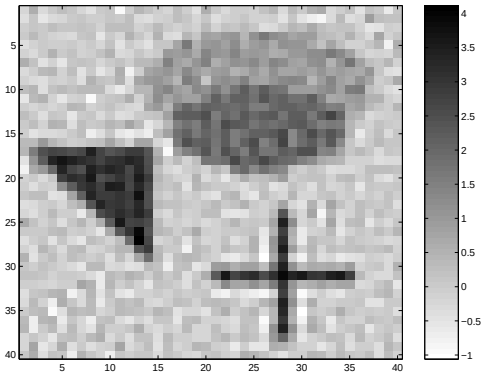

Figure 12: Image deblurring example: RPTRS solution picture

5.3 Comparison with CGLS

We compare our approach to the conjugate gradients based method for solving the least-squares problems CGLS. CGLS is one of the most robust regularization techniques that can handle very large problem instances. This method, described in [25] (see also [16]), applies conjugate gradients (CG) to the normal equations $T^*Tx = T^*d$ along with an early termination criteria to obtain the regularized solution. The stopping condition is based on the discrepancy principle, i.e. the method terminates once the residual is smaller than some prescribed bound δ. Typically, the value of δ is based on the norm of the noise.

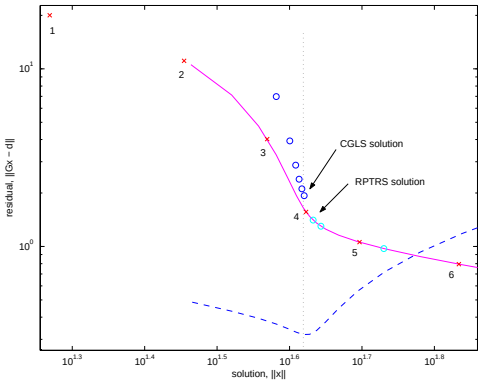

Figure 13: Image deblurring example: corresponding L-curve with CGLS points

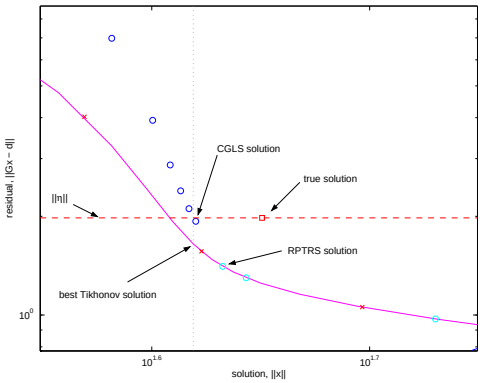

Figure 14: Image deblurring example: CGLS, RPTRS, x_{true}, best Tikhonov solutions

We applied the CGLS algorithm on the data from the previous example supplying δ to be precisely the norm of the noise, i.e. $\delta = \|\eta\|_2$. In some sense this corresponds to the best case for CGLS. The results are presented in Table 5.2 and Figure 13. The CGLS points are shown as circles above the L-curve. The CGLS solution is almost as good as the best Tikhonov solution. This result is not unusual and emphasizes the fact that the method was applied with exact knowledge of the noise. However, comparing both CGLS and RPTRS solutions to the true one (see Figure 14), we see that both methods achieve practically the same accuracy.

The RPTRS algorithm, though, does not require a specific value of the norm of the noise. This is a big advantage in a sense that CGLS might perform very poorly if supplied with slightly smaller (or larger) value of δ. Figure 15 illustrates this situation. Running CGLS with $\delta = 0.6\,\|\eta\|_2$ results in a larger number of iterations (31 comparing to 6 with

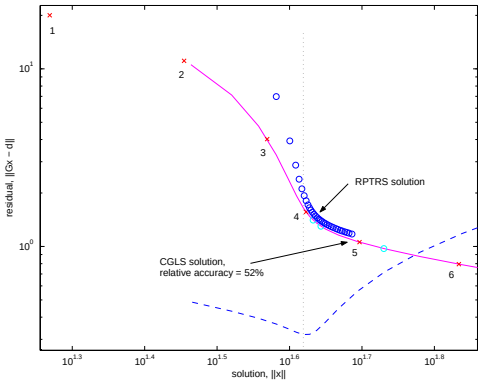

Figure 15: Image deblurring example: CGLS with $\delta = 0.6 \|\eta\|_2$, rel.acc. $= 52\%$

##	$\|x\|_2$	$\|Gx - d\|_2$	accuracy [%]
1	3.8162e+001	6.9804e+000	47.59
2	3.9849e+001	3.9256e+000	41.83
3	4.0593e+001	2.8676e+000	38.99
4	4.1045e+001	2.3920e+000	36.97
5	4.1406e+001	2.1105e+000	35.51
6	4.1706e+001	1.9309e+000	34.41

Table 1: Data for points visited by the CGLS algorithm with $\delta = \|\eta\|_2$

$\delta = \|\eta\|_2$) and the computed solution is much worse now. This shows the importance of a robust stopping criteria that does not rely on the possibly uncertain data.

The main advantage of the CGLS method is its speed. Each iteration of the algorithm requires only several matrix-vector multiplications, where only the original matrix G is used. This allows for solutions of problems that involve large sparse matrices which are never formed explicitly. At the same time, the RPTRS algorithm can be viewed as a matrix-free iterative algorithm based on the Lanczos method that features conjugate gradients steps as well. This leads to a conclusion that combining both approaches may result in a better algorithm that can provide a reliable and a fast way to locate a regularized solution in the absence of any certain knowledge about the noise.

##	$\|x\|_2$	$\|Gx-d\|_2$	accuracy [%]	time	t	λ
1	1.8573e+001	2.0010e+001	65.39	2.794	652.166	-9.8851e-001
2	2.8472e+001	1.1095e+001	49.63	3.054	994.155	-3.4166e-001
3	3.7079e+001	4.0222e+000	38.07	3.014	1271.46	-7.7717e-002
4	4.1957e+001	1.5642e+000	31.82	3.695	1378.38	-7.7959e-003
5	4.9732e+001	1.0570e+000	57.14	6.509	1392.12	-5.3731e-004
6	6.8218e+001	7.9497e-001	116.29	5.558	1393.45	-1.0426e-004
+1	4.2910e+001	1.4078e+000	32.63	2.834	1384.90	-4.1666e-003
+2	5.3732e+001	9.7305e-001	71.49	2.794	1392.69	-3.2078e-004
+3	4.3991e+001	1.2993e+000	35.36	2.824	1388.32	-2.3520e-003

Table 2: Data for points visited by the RPTRS algorithm

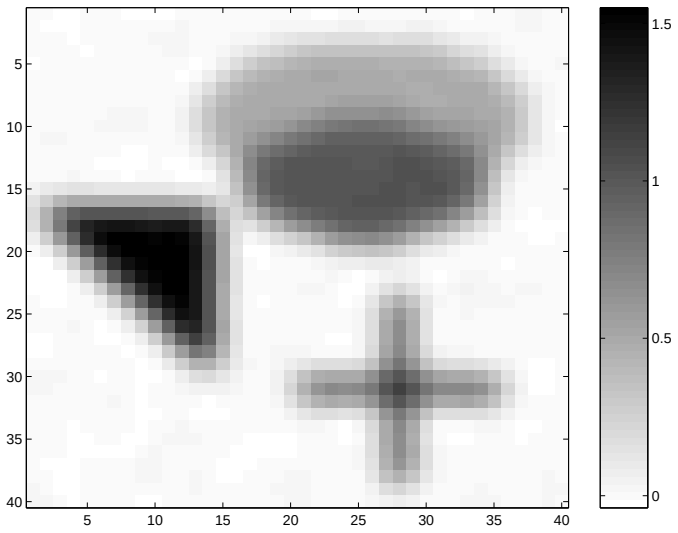

Figure 16: Image deblurring example: point #1, t = 652.166, rel.acc. = 65.39%

6 Conclusion

We have applied ideas from the RW algorithm for TRS to efficiently find the point of maximum curvature on the L-curve. This provides a regularization procedure for ill-conditioned problems $Gx = d$. We have taken advantage of the fact that each iteration of the RW algorithm corresponds to a point on the L-curve. We implicitly change the trust region radius while applying the RW algorithm. The changes drive the algorithm to the correct radius

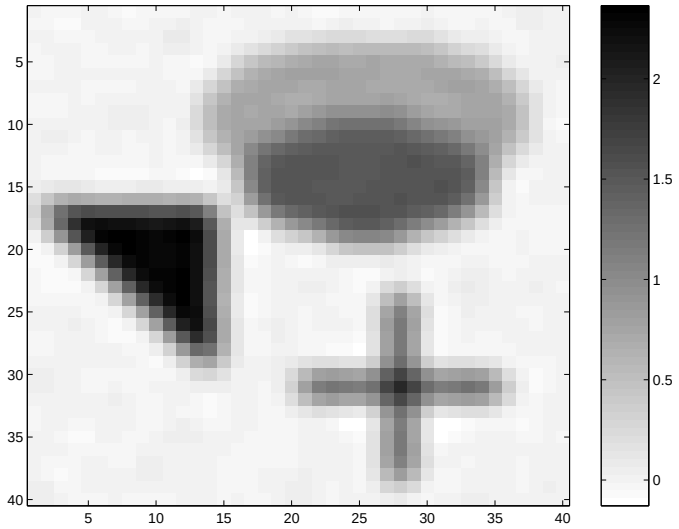

Figure 17: Image deblurring example: point #2, t = 994.155, rel.acc. = 49.63%

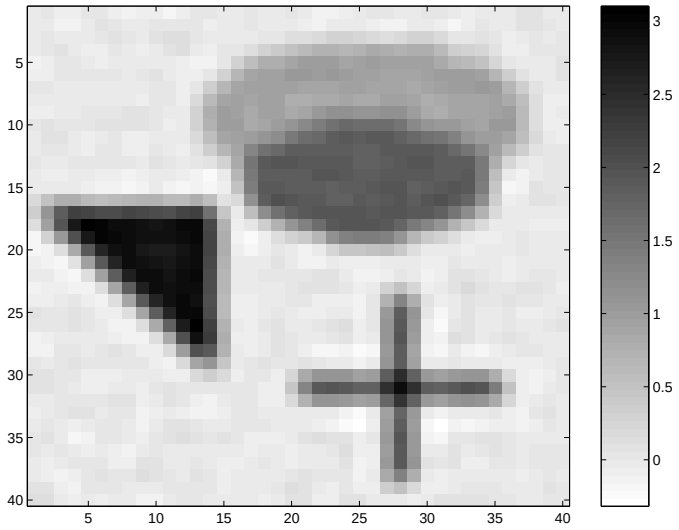

Figure 18: Image deblurring example: point #3, t = 1271.46, rel.acc. = 38.07%

that corresponds to the elbow, the point of maximum curvature on the L-curve.

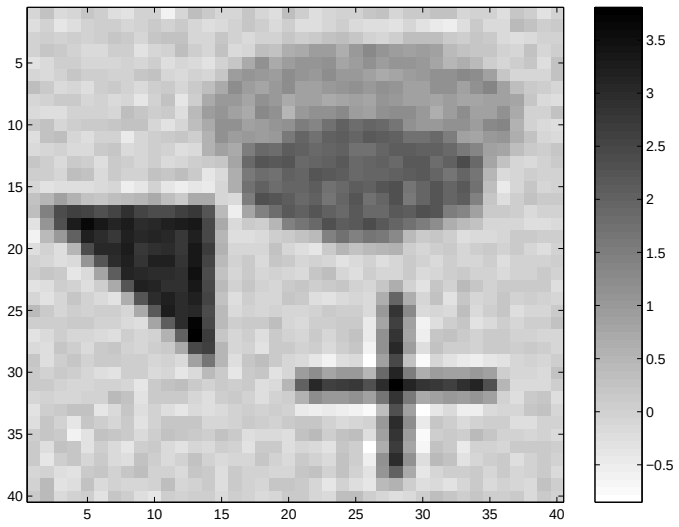

Figure 19: Image deblurring example: point #4, t = 1378.38, rel.acc. = 31.82%

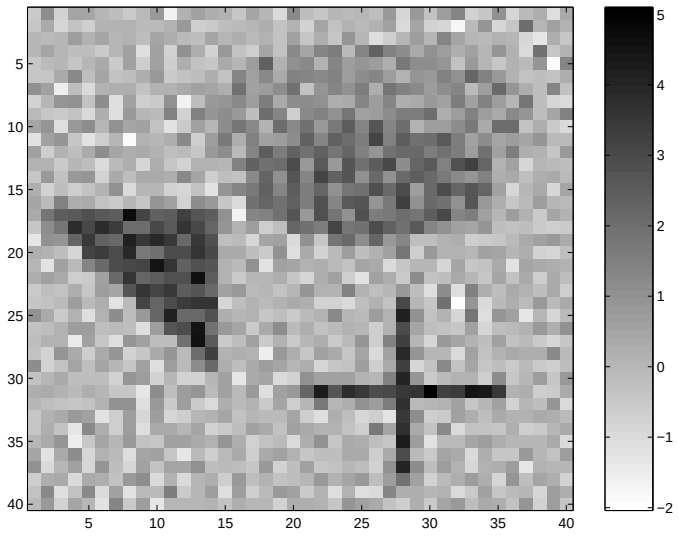

Figure 20: Image deblurring example: point #5, t = 1392.12, rel.acc. = 57.14%

Acknowledgement: The authors thank Professor Arkadi Nemirovski from Technion - Israel Institute of Technology, for helpful talks related to the CGLS method.

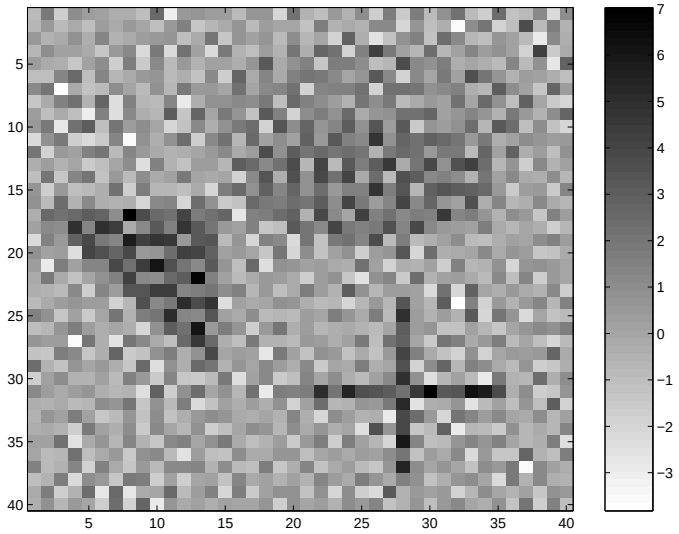

Figure 21: Image deblurring example: point #6, t = 1393.45, rel.acc. = 116.29%

References

[1] R. ASTER, B. BORCHERS, and C. THURBER. *Parameter Estimation and Inverse Problems*. Academic Press, 2004.

[2] D. CALVETTI, P.C. HANSEN, and L. REICHEL. L-curve curvature bounds via Lanczos bidiagonalization. *Electron. Trans. Numer. Anal.*, 14:135–150 (electronic), 2002. Orthogonal polynomials, approximation theory, and harmonic analysis (Inzel, 2000).

[3] D. L. DONOHO and I. M. JOHNSTONE. Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, 81(3):425–455, 1994.

[4] D. L. DONOHO and I. M. JOHNSTONE. Adapting to unknown smoothness via wavelet shrinkage. *Journal of the American Statistical Association*, 90(432):1200–1224, 1995.

[5] D. L. DONOHO, I. M. JOHNSTONE, G. KERKYACHARIAN, and D. PICARD. Wavelet shrinkage: asymptopia? *Journal of the Royal Statistical Society, Ser. B*, 57:301–337, 1995.

[6] L. ELDÉN. Algorithms for the regularization of ill-conditioned least squares problems. *BIT*, 17:134–145, 1977.

[7] C. FORTIN and H. WOLKOWICZ. The trust region subproblem and semidefinite programming. *Optimization Methods and Software*, 19(1):41–67, 2004. special issue dedicated to Jochem Zowes 60th birthday, Guest Editors: Florian Jarre and Michal Kocvara.

[8] W. GANDER. On the linear least squares problem with a quadratic constraint. Technical Report STAN-CS-78-697, Department of Computer Science, Stanford University, Stanford, CA, 1978.

[9] D.M. GAY. Computing optimal locally constrained steps. *SIAM J. Sci. Statist. Comput.*, 2:186–197, 1981.

[10] G. H. GOLUB and U. von MATT. Generalized cross-validation for large-scale problems. *Journal of Computational and Graphical Statistics*, 6(1):1–34, March 1997.

[11] G. H. GOLUB and U. von MATT. Tikhonov regularization for large scale problems. Technical Report SCCM-97-03, Stanford University, 1997.

[12] G. H. GOLUB and U. von MATT. Tikhonov regularization for large scale problems. In *Scientific computing (Hong Kong, 1997)*, pages 3–26. Springer, Singapore, 1997.

[13] G.H. GOLUB and C.F. VAN LOAN. *Matrix Computations*. Johns Hopkins University Press, Baltimore, Maryland, 3^{nd} edition, 1996.

[14] J. HADAMARD. Sur les problhmes aux dirivies partielles et leur signification physique. *Princeton University Bulletin*, pages 49–52, 1902.

[15] J. HADAMARD. *Lectures on Cauchy's problem in linear partial differential equations*. Dover Publications, New York, 1953.

[16] M. HANKE and P.C. HANSEN. Regularization methods for large-scale problems. *Surveys Math. Indust.*, 3(4):253–315, 1993.

[17] P.C. HANSEN. Analysis of discrete ill-posed problems by means of the L-curve. *SIAM Rev.*, 34(4):561–580, 1992.

[18] P.C. HANSEN. Regularization tools: a Matlab package for analysis and solution of discrete ill-posed problems. *Numer. Algorithms*, 6(1-2):1–35, 1994.

[19] P.C. HANSEN. *Rank-deficient and discrete ill-posed problems*. SIAM Monographs on Mathematical Modeling and Computation. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1998. Numerical aspects of linear inversion.

[20] P.C. HANSEN. The l-curve and its use in the numerical treatment of inverse problems. Technical report, Technical University of Denmark, 1999.

[21] P.C. HANSEN and D.P. O'LEARY. The use of the L-curve in the regularization of discrete ill-posed problems. *SIAM J. Sci. Comput.*, 14(6):1487–1503, 1993.

[22] P. LANCASTER. On eigenvalues of matrices dependent on a parameter. *Numer. Math.*, 6:377–387, 1964.

[23] P. LANCASTER and M. TISMENETSKY. *The theory of matrices.* Computer Science and Applied Mathematics. Academic Press Inc., Orlando, Fla., second edition, 1985.

[24] F. NATTERER. *The Mathematics of Computerized Tomography.* Wiley, New York, 1986.

[25] A. NEMIROVSKII. The regularization properties of the adjoint gradient method in ill-posed problems. *USSR Comput. Math. and Math. Phys.*, 26(2):7–16, 1986.

[26] A. NEUMAIER. Solving ill-conditioned and singular linear systems: a tutorial on regularization. *SIAM Rev.*, 40(3):636–666 (electronic), 1998.

[27] M.L. OVERTON and R.S. WOMERSLEY. Second derivatives for optimizing eigenvalues of symmetric matrices. *SIAM J. Matrix Anal. Appl.*, 16(3):697–718, 1995.

[28] F. RENDL and H. WOLKOWICZ. A semidefinite framework for trust region subproblems with applications to large scale minimization. *Math. Programming*, 77(2, Ser. B):273–299, 1997.

[29] M. ROJAS, S.A. SANTOS, and D.C. SORENSEN. A new matrix-free algorithm for the large-scale trust-region subproblem. *SIAM J. Optim.*, 11(3):611–646 (electronic), 2000/01.

[30] M. ROJAS and D.C. SORENSEN. A trust-region approach to the regularization of large-scale discrete forms of ill-posed problems. *SIAM J. Sci. Comput.*, 23(6):1842–1860 (electronic), 2002.

[31] C.B. SHAW, Jr. Improvement of the resolution of an instrument by numerical solution of an integral equation. *J. Math. Anal. Appl.*, 37:83–112, 1972.

[32] D.C. SORENSEN. Minimization of a large-scale quadratic function subject to a spherical constraint. *SIAM Journal on Optimization*, 7(1):141–161, 1997.

[33] R. STERN and H. WOLKOWICZ. Indefinite trust region subproblems and nonsymmetric eigenvalue perturbations. *SIAM J. Optim.*, 5(2):286–313, 1995.

[34] R.J. STERN and J.J. YE. Variational analysis of an extended eigenvalue problem. *Linear Algebra Appl.*, 220:391–417, 1995.

[35] A.N. TIKHONOV. Regularization of incorrectly posed problems. *Soviet Math.*, 4:1624–1627, 1963.

[36] A.N. TIKHONOV and V.Y. ARSENIN. *Solutions of Ill-Posed Problems*. V.H. Winston & Sons, John Wiley & Sons, Washington D.C., 1977. Translation editor Fritz John.

[37] R.J. VANDERBEI. *The Amateur Astrophotographer*. Forthcoming.

[38] U. von MATT. *Large Constrained Quadratic Problems*. Ph. D. thesis, Institute for Scientific Computing, ETH, Zürich, Switzerland, 1993.