An Adaptive Pruning Algorithm for the Discrete L-Curve Criterion ⋆

Per Christian Hansen *, Toke Koldborg Jensen

Informatics and Mathematical Modelling, Building 321
Technical University of Denmark, DK-2800 Lyngby, Denmark

Giuseppe Rodriguez

Dipartimento di Matematica e Informatica, Università di Cagliari
viale Merello 92, I-09123 Cagliari, Italy

Abstract

We describe a robust and adaptive implementation of the L-curve criterion, i.e., for locating the corner of a discrete L-curve consisting of a log-log plot of corresponding residual and solution norms of regularized solutions from a method with a discrete regularization parameter (such as truncated SVD or regularizing CG iterations). Our algorithm needs no pre-defined parameters, and in order to capture the global features of the curve in an adaptive fashion, we use a sequence of pruned L-curves that correspond to considering the curves at different scales. We compare our new algorithm to existing algoritms and demonstrate its robustness by numerical examples.

Key words: Discrete ill-posed problems, L-curve criterion, regularization, parameter-choice method.

⋆ This work was supported in part by grant no. 21-03-0574 from the Danish Natural Science Research Foundation, and by COFIN grant no. 2002014121 from MIUR (Italy).

* Corresponding author

Email addresses: `pch@imm.dtu.dk` (Per Christian Hansen), `tkj@imm.dtu.dk` (Toke Koldborg Jensen), `rodriguez@unica.it` (Giuseppe Rodriguez).

URLs: `http://www.imm.dtu.dk/~pch` (Per Christian Hansen), `http://www.imm.dtu.dk/~tkj` (Toke Koldborg Jensen), `http://bugs.unica.it/~gppe/` (Giuseppe Rodriguez).

1 Introduction

We are concerned with discrete ill-posed problems, i.e., linear systems of equations $A\,x = b$ or linear least squares problems $\min \|A\,x - b\|_2$ with a very ill conditioned coefficient matrix A, obtained from the discretization of an ill-posed problem, such as a Fredholm integral equation of the first kind. To compute stable solutions to such systems under the influence of data noise, one must use some kind of regularization in which prior information is incorporated in the solution method. All regularization methods make use of a certain regularization parameter that controls the amount of stabilization imposed on the solution, and in most cases it is necessary to choose this parameter from the given problem and the given set of data.

In this paper we are concerned with regularization methods for which the regularization parameter takes discrete values k, e.g., when the stabilization is imposed as a requirement that the regularized solution lies in a k-dimensional subspace. Examples of such methods are truncated (G)SVD and regularizing CG iterations. These methods can be thought of as producing a sequence of n_p regularized solutions x_k for $k = 1, 2, \ldots, n_p$, and the key point is to choose the optimal value of the parameter k.

A variety of methods have been proposed for the parameter choice problem, such as the discrepancy principle, error-estimation methods, generalized cross-validation, and the L-curve criterion. For an overview, see Chapter 7 in [6].

For problems with a continuous regularization parameter, the L-curve criterion has proven to be useful in a number of problems. The L-curve is a plot in log-log scale of corresponding values of the residual and solution norms parameterized by the regularization parameter. For problems with a discrete regularization parameter k, the discrete L-curve consists of a plot of the set of points

$$(\,\log \|A\,x_k - b\|_2\,,\ \log \|x_k\|_2\,), \qquad k = 1, \ldots, n_p. \tag{1}$$

For many problems arising in a variety of applications, it is found that this curve – continuous or discrete – has a particular "L" shape, and that the optimal regularization parameter corresponds to a point on the curve near the "corner" of the L-shaped region; see, e.g., [6, §7.5] or [7] for an analysis of this phenomenon.

For continuous L-curves it was suggested in [8] to define the corner as the point with maximum curvature; this criterion is implemented in Regularization Tools [5] and has proven quite successful in many applications.

For discrete L-curves it is less obvious how to make an operational definition of a corner suited for computer implementation – in spite of the fact that it is often easy to "eyeball" the corner. While a few attempts have been made, cf. [3], [5] and [10], we feel that there is a need for a robust general-purpose algorithm for computing the corner of a discrete L-curve.

The goal of this work is therefore to describe such a robust algorithm which finds the optimal k via the discrete L-curve for a large class of problems – *provided* that the L-curve criterion makes sense for the particular problem, which means that the noise-free problem must satisfy the discrete Picard condition [6, §4.5], and the L-curve must exhibit a distinguishable corner. Our algorithm is partly based on an earlier version from [10].

Our paper is organized as follows. Section two introduces the basic regularization methods, and section three introduces the discrete L-curve, and previous algorithms for finding the corner. Also, the notation used in the present paper is introduced. Section four is the main contribution of the paper and describes in detail the proposed algorithm. The algorithm is tested thoroughly in section five where the performance is shown using a series of smaller test problems as well as a large-scale problem. Section six mentions possible applications of the proposed algorithm, and section seven concludes the paper.

2 Some Regularization Methods

For convenience assume that the coefficient matrix A is of dimensions $m \times n$ with $m \geq n$, and let the SVD of A be given by $A = \sum_{i=1}^{n} u_i \, \sigma_i \, v_i^T$. What characterizes a discrete ill-posed problem is that the singular values σ_i decay gradually to zero, and that the absolute value of the right-hand side coefficients $u_i^T b$ decay (perhaps slightly) faster.

One of the best known regularization methods with a continuous regularization parameter λ is Tikhonov's method, which amounts to computing the solution

$$x_\lambda = \operatorname{argmin} \left\{ \|A\,x - b\|_2^2 + \lambda^2 \|H\,x\|_2^2 \right\}, \tag{2}$$

in which the matrix H defines a smoothing norm suited for the given problem. The corresponding L-curve consists of a parametric log-log plot of the residual and solution norms, and for $H = I$ these norms are given by

$$\|A\,x_\lambda - b\|_2^2 = \sum_{i=1}^{n} \left(\frac{\lambda^2}{\sigma_i^2 + \lambda^2} \, u_i^T b \right)^2, \qquad \|x_\lambda\|_2^2 = \sum_{i=1}^{n} \left(\frac{\sigma_i}{\sigma_i^2 + \lambda^2} \, u_i^T b \right)^2 \tag{3}$$

with λ as the parameter. This L-curve is C^∞ with respect to λ, and the standard definition of the corner is the point for which the L-curve's curvature has a maximum. Efficient large-scale algorithms for computing the curvature, as well as lower and upper bounds for the curvature, are described in [2].

The best known regularization method with a discrete regularization parameter is probably the truncated SVD (TSVD) method. For any $k < n$ the TSVD solution x_k is defined by

$$x_k = \sum_{i=1}^{k} \frac{u_i^T b}{\sigma_i} \, v_i. \tag{4}$$

The TSVD solutions are similar to the Tikhonov solutions when the matrix H is the identity. The residual and solution norms for x_k are given by

$$\|A\, x_k - b\|_2^2 = \sum_{i=k+1}^{n} (u_i^T b)^2, \qquad \|x_k\|_2^2 = \sum_{i=1}^{k} \left(\frac{u_i^T b}{\sigma_i} \right)^2 , \tag{5}$$

and the L-curve for TSVD consists of the set of these points plotted in log-log scale.

There is also a truncated GSVD method, involving the matrix pair (A, H) and corresponding to a penalization term of the form $\|H\,x\|_2$ in the Tikhonov formulation. We shall not go deeper into this method, but instead refer to [6].

The CGLS algorithm is mathematically equivalent to applying the CG method to the normal equations, and when applied to ill-posed problems this method exhibits semi-convergence, i.e., initially the iterates approach the exact solution while at later stages they deviate from it again. Moreover, it is found that the number k of iterations plays the role of the regularization parameter. Hence, these so-called regularizing CG iterations also lead to a discrete L-curve. For more details, see, e.g., §§6.3–6.5 in [6].

3 The Discrete L-Curve Criterion

A standard tool for analyzing the discrete ill-posed problems is the discrete Picard plot, which is a plot of the quantities σ_i, $|u_i^T b|$ and $|u_i^T b|/\sigma_i$ that arise in (3), (4) and (5). For a discrete ill-posed problem to possess a meaningful regularized solution, it must satisfy the discrete Picard condition, i.e., the noise-free coefficients must decay faster than the singular values, on the average.

An example using the test problem baart from [5] of size $n = 16$ is shown in Fig. 1, together with the norms $\|A\,x_k - b\|_2$ and $\|x_k\|_2$. The figure illustrates that $\|A\,x_k - b\|_2$ decreases monotonically with k, and that $\|x_k\|$ increase monotonically with k. The dashed vertical line indicates the index of the optimal solution, and we notice that this solution lies in the transition region between a decaying residual norm and an increasing solution norm. In this example, with a fast decay of the singular values, the corner of the L-curve will lie near the point where the solution coefficients start to get dominated by the noise.

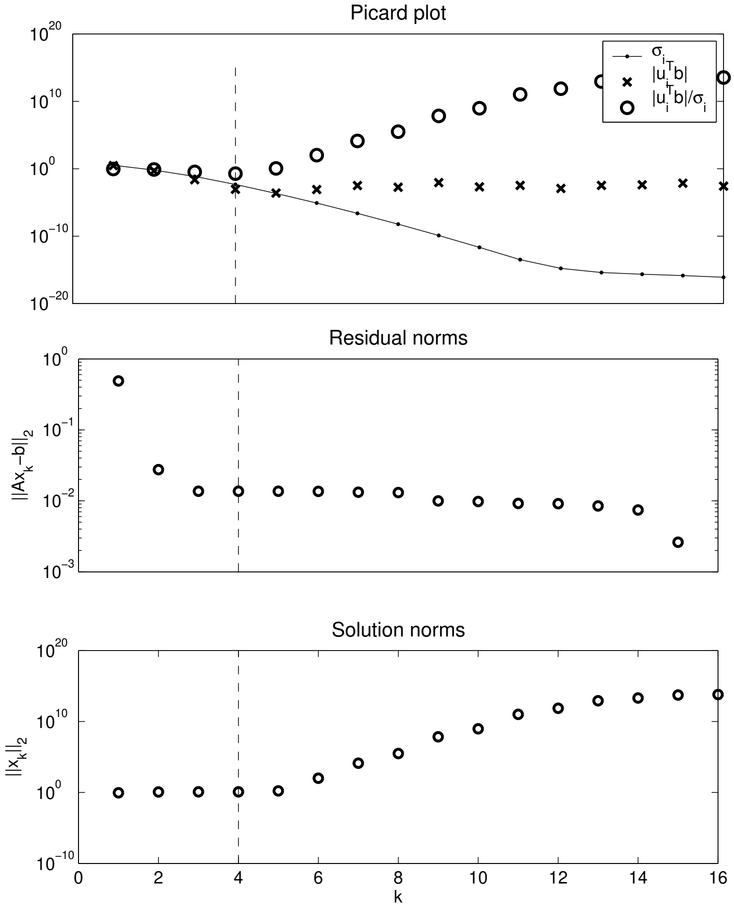


Fig. 1. Example of Picard plot (top) and illustration of decay of $\|Ax_k - b\|_2$ (middle) and increase of $\|x_k\|_2$ (bottom). A dotted line indicates the index $k = 4$ of the best attainable solution.

For problems with more slowly decaying singular values this transition area gets wider, and the corner becomes less distinct. This might cause a problem for the L-curve method because the optimal solution and the corner of the L-curve are less likely to coincide. A thorough analysis of these aspects are outside the scope of the present paper. Nevertheless, this fact must be kept in mind when testing the performance of the methods – especially for large-scale problems with many slowly decaying singular values.

In the rest of the paper we occasionally need to talk about the angle between the two line segments associated with a triple of L-curve points, with the usual convention of the sign of the angle. Specifically, let $\mathcal{P}_j$, $\mathcal{P}_k$ and $\mathcal{P}_\ell$ be three points satisfying $j < k < \ell$, and let $v_{r,s}$ denote the *normalized* vector from $\mathcal{P}_r$ to $\mathcal{P}_s$. Then we define the angle $\theta(j,k,\ell) \in [-\pi, \pi]$ associated with the triplet as the angle between the two vectors $v_{j,k}$ and $v_{k,\ell}$, i.e.,

$$\theta(j,k,\ell) = \angle(v_{j,k}, v_{k,\ell}). \tag{6}$$

With this definition, an angle $\theta(j,k,\ell) < 0$ corresponds to a point which is a

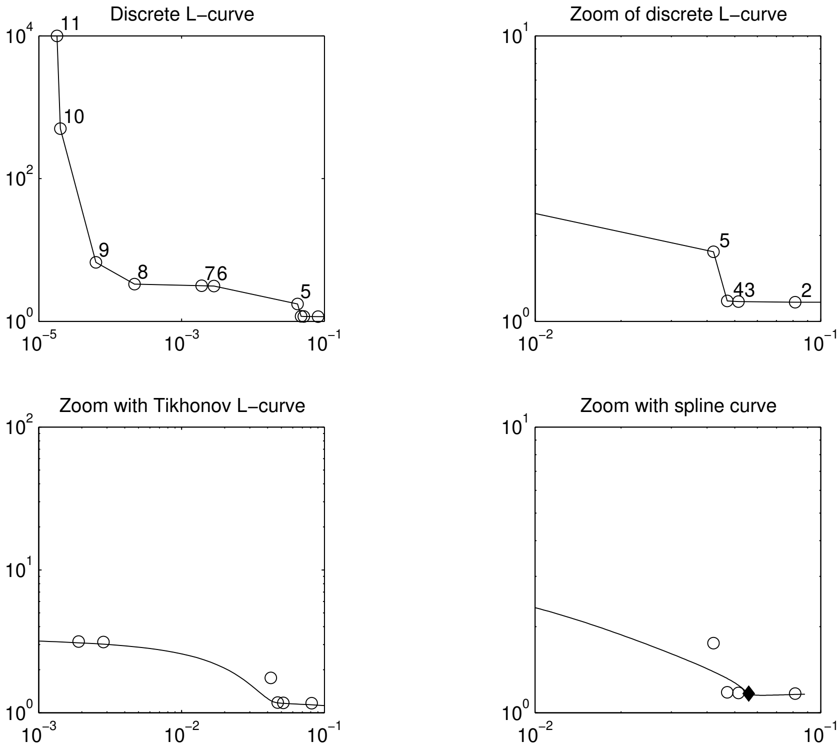


Fig. 2. Top: a discrete L-curve for TSVD with a global corner at $k = 9$ and a little "step" at $k = 4$; the smallest angle between neighboring triplets of points occurs at $k = 4$. Bottom left: part of the Tikhonov L-curve for the same problem. Bottom right: part of the 2D spline curve used by the Matlab function l_curve in REGULARIZATION TOOLS [5]; the point on the spline curve with maximum curvature is indicated by the diamond.

potential candidate for the corner point, while $\theta(j,k,\ell) \geq 0$ indicates a point of no interest.

In principle, it ought to be easy to find the corner of a discrete L-curve: compute the angle $\theta(k-1,k,k+1)$ for $k = 2,\ldots,n_p - 1$ and associate the corner point $\mathcal{P}_k$ with the angle closest to $-\pi/2$. Unfortunately, this simple approach is not guaranteed to work in practice as an individual algorithm because discrete L-curves often have several small local corners, occasionally associated with clusters of L-curve points. We remark that the continuous Tikhonov L-curves do usually not have this drawback, due to the inherent "smoothing" in the expressions (3) compared to the TSVD expressions (5).

A global point of view of the discrete L-curve is needed in order to find the desired corner of the overall curve. An alternative approach would therefore be to locate the vertical and horizontal parts of the curve, and let the corner be the point right between these parts. In principle this seems as an easy task, but in practice it turns out to be very difficult to implement such a robust corner-finding strategy.

Figure 2 illustrates these issues using the TSVD method on a tiny problem. The top left plot shows a discrete L-curve with 11 points, with the desired global corner at $k = 9$ and with a local corner at $k = 4$ (shown in more detail in the top right plot). For this particular L-curve, the smallest angle between neighboring line segments is attained at $k = 4$; but the L-curve's little "step" here is actually an insignificant part of the overall horizontal part of the curve in this region.

The bottom left plot in Fig. 2 shows a part of the continuous Tikhonov L-curve for the same problem, together with the points of the discrete TSVD L-curve. Clearly the Tikhonov L-curve is not affected very much by the little "step" of the discrete L-curve.

Two algorithms have been proposed for computing the corner of a discrete L-curve, taking into account the need to capture the overall behavior of the curve and avoiding the local corners.

The first algorithm was described in [8], and is used in the Matlab function l_curve in the REGULARIZATION TOOLS package [5]. The algorithm is summarized in Fig. 3. This algorithm fits a 2D spline curve to the points of the discrete L-curve. The advantage of this approach is that the curvature of the spline curve is well defined and independent of the parametrization, and the algorithm returns the point on the discrete L-curve closest to the corner of the spline curve.

However, the spline curve has a tendency to track the unwanted local corners of the discrete L-curve, and therefore a preprocessing stage is added where the L-curve points are first smoothed by means of a local low-degree polynomial. Unfortunately, this smoothing step depends on a few fixed parameters. Hence the overall algorithms is not adaptive, and often it is necessary to hand-tune the parameters of the smoothing process in order to remove the influence of the small local corners, without missing the global corner.

ALGORITHM l_corner FROM REGULARIZATION TOOLS

1. For $i = q + 1, \ldots, n_p - q - 1$
2. Fit two degree-d polynomials to coordinates of points $\mathcal{P}_{i-q}, \ldots, \mathcal{P}_{i+q}$.
3. Let $\widehat{\mathcal{P}}_i$ = values of fitting polynomials at $\mathcal{P}_i$.
4. Fit a 2D spline curve to the new points $\{\widehat{\mathcal{P}}_i\}$.
5. Compute derivatives of the spline curve at $\{\widehat{\mathcal{P}}_i\}$.
6. Compute the curvature κ_i at the points $\{\widehat{\mathcal{P}}_i\}$.
7. Let $k = \max_i(\kappa_i)$.

Fig. 3. The overall design of the algorithm l_corner from [5]. The two integers d and q that determine the fit are problem dependent.

If we use the default parameters in [5] then we obtain the spline curve shown in the bottom right plot of Fig. 2 whose corner (indicated by the diamond) is, incorrectly, located at the little "step."

A more recent algorithm, called the *triangle method,* was described in [3]. The key idea here is to consider the following triples of L-curve points:

$$\left(\mathcal{P}_j, \mathcal{P}_k, \mathcal{P}_{n_p}\right), \qquad j = 1, \ldots, n_p - 2, \quad k = j+1, \ldots, n_p - 1,$$

and identify as the corner the triple where the oriented angle $\theta(j, k, n_p)$ is minimum. If all angles $\theta(j, k, n_p)$ are greater than $-\pi/8$ then the L-curve is considered "flat" and the leftmost point is chosen. Note that the leftmost point $\mathcal{P}_{n_p}$ is always included in the calculations. The details of the implementation are shown in Fig. 4. Unfortunately, the authors of the algorithm [3] were not able to provide us with a working Matlab code (only a modified Fortran subroutine used inside a CG algorithm), and hence the tests in Section 5 are done using our own Matlab implementation.

Triangle Algorithm

Remove points with zero norm
Initialize $c_{\max} = -2$
for $k = 1, \ldots, n_p - 2$
    for $j = k+1, \ldots, n_p - 1$
        Compute vectors: $v_1 = \mathcal{P}_k - \mathcal{P}_j$ and $v_2 = \mathcal{P}_j - \mathcal{P}_{n_p}$
        Compute: $c = \frac{-v_1^T v_2}{\|v_1\|_2 \|v_2\|_2}$
        Compute: $w = \det([v_1,\ v_2])$
        if $c > \cos(7\pi/8)$ and $c > c_{\max}$ and $w < 0$
            Set: corner $= j$ and $c_{\max} = c$

Fig. 4. The overall design of the triangle algorithm [3] for finding the corner of a discrete L-curve. The two loops give a complexity of $\mathcal{O}(n_p^2)$.

For the tiny L-curve in Fig. 2 this algorithm returns $k = 8$ which is a good estimate of the optimal k. Unfortunately, there is one main concern with the triangle algorithm. The complexity is $\frac{1}{2}(n_p - 1)(n_p - 2)$ which is too high for large-scale problems where n_p can be large. The amount of computation can be reduced by working with a subsampled L-curve, but the subsampling must be done carefully by the user and is not part of the algorithm.

4 The Adaptive Pruning Algorithm

Common for the regularization methods mentioned above is that the discrete L-curve is monotonic in the sense that the solution norms increase monotonically with k and the residual norms decrease monotonically with k. While this property is crucial for the understanding of the L-curve criterion, it may fail to be satisfied in real computations due to the effect of rounding errors. Hence we also want our algorithm to be robust to these effects.

An implementation of a robust discrete L-curve criterion should have complexity of at most $\mathcal{O}(n_p \log n_p)$, and must include a means for adaptively filtering small local phenomena, including local corners. The process must be adaptive, because the size or scale of the local phenomena is problem dependent and usually unknown by the user. The algorithm must give a useful answer in all circumstances – also when the user supplies an L-curve without a corner (e.g., when n_p is not big enough to include solutions with a large norm, or when the problem is well conditioned). Finally, the algorithm should not make use of any pre-set parameters.

To achieve the required adaptivity and robustness, our new algorithm for locating the corner of a discrete L-curve consists of an initialization and two main stages. In the initialization we remove any points where the residual norm or solutions norm is zero. In the first main stage we compute the corner of the L-curve at different scales or resolutions (not knowing a priori which scale is optimal). In the second main stage we then compute the overall best corner from the candidates found in the first stage. Also, during the two stages we monitor the results, in order to identify L-curves that lack a corner (e.g., because the problem is well conditioned).

4.1 The Overall Algorithm

The key idea is that if we remove exactly the right amount of points from the discrete L-curve, then the corner can easily be found from the remaining set of points. However, the set of points to be removed is unknown. If too few points are removed we still maintain unwanted local features, and if too many points are removed the corner will be incorrectly located or may disappear.

In the first main stage of the overall algorithm, we therefore work with a sequence of pruned L-curves, that is, curves in which a varying number of points are removed. For each case we locate the corner of the pruned L-curve, making sure that a corner is always found if the L-curve is convex. This produces a short list of candidate points to the corner $\mathcal{P}_{k_1}, \ldots, \mathcal{P}_{k_r}$ and several (or possibly all) of the candidates may be identical. The candidate list is then sorted, so that the indices satisfy $k_i > k_{i-1}$, and duplicate entries are removed.

In the second stage we then pick the best corner from the list of candidates found in the first stage. If the candidate list includes only one point then we are done, otherwise we must choose a single point from the list. We cannot exclude that points on the vertical part of the L-curve are among the candidates, and as a safeguard we therefore seek to avoid any such point. If we traverse the sorted candidate list from k_1 to k_r, then the wanted corner is the last candidate point before reaching the vertical part, provided that the curvature in this point is acceptable. If no point lies on the vertical branch of the L-curve, then the leftmost point k_r is a good choice. To evaluate the feasibility of the first candidate point, the first point of the L-curve $\mathcal{P}_1$ is included in the candidate list as the first point k_0. The following two criteria are set up to check for feasible points:

Norm increase. The point $\mathcal{P}_{k_{i+1}}$, $i = 0, \ldots, r-1$ is considered lying on the vertical branch of the L-curve if going from $\mathcal{P}_{k_i}$ to $\mathcal{P}_{k_{i+1}}$ yields a larger increase in solution norm than decrease in residual norm. This is equivalent to the vector $v_{k_i,k_{i+1}}$ having a slope $\phi_i > \pi/4$.
Curvature. The curvature of a candidate point $\mathcal{P}_{k_i}$ is acceptable if the angle $\theta(k_{i-1}, k_i, k_{i+1})$ is negative.

Adaptive Pruning Algorithm

0. Remove points with zero norm and create empty list $\mathcal{L} = \emptyset$.
1. Initialize $p = \min(5,\, n_p - 1)$
2. **Stage one:** while $p < 2(n_p - 1)$
3. $\quad$ $p = \min(p,\, n_p - 1)$
4. $\quad$ Create a pruned L-curve consisting of the p largest line segments.
5. $\quad$ Locate the corner $\mathcal{P}_k$ of the pruned L-curve.
6. $\quad$ Add the corner to the list: $\mathcal{L} = \mathcal{L} \cup \{\mathcal{P}_k\}$.
7. $\quad$ $p = 2p$
8. **Stage two:** if $\#\mathcal{L} = 1$ then $k = k_1$; return.
9. Otherwise for $i = 1, \ldots, \#\mathcal{L} - 1$
10. $\quad$ Compute the slope ϕ_i associated with point $\mathcal{P}_{k_i}$ in $\mathcal{L}$.
11. If $\max\{\phi_i\} < \pi/4$ then $k = \max\{k_i\}$; return.
12. Otherwise let $k = \min\{k_i : \phi_i > \pi/4 \wedge \theta(k_{i-1},\, k_i,\, k_{i+1}) < 0\}$.

Fig. 5. The overall design of the adaptive pruning algorithm for locating the corner of a discrete L-curve. Here, $\mathcal{P}_k$ denotes a point on the original L-curve, and $\mathcal{P}_{k_i}$ denotes a candidate point in the list $\mathcal{L}$.

The complete algorithm is shown in Fig. 5. The computation of the corner of each pruned L-curve is done by two separate routines which we describe below, one relying on the angles between subsequent line segments and one aiming at tracking the global vertical and horizontal features of the L-curve. The routine based on angles additionally checks for correct curvature of the given pruned L-curve. No corner is returned from this routine if the pruned L-curve is flat or concave. The returned corner points are added to the candidate list and, after running through all pruned L-curves, the corner is found as described above. This algorithm will always return an index k to a single point which is considered as the corner of the discrete L-curve, unless all the pruned L-curves are found to be concave. In this case the algorithm will return an error message. Our experiments (see Section 5) indicate that the overall complexity of the algorithm is $\mathcal{O}(n_p \log n_p)$.

4.2 Corner Location Based on Angles

This corner selection strategy has already been proposed in [10] and is similar in spirit to the guideline of the triangle method described in [3] and summarized in Fig. 4.

The procedure consists of finding the angle $\theta(k-1,k,k+1)$ over the discrete L-curve which is closest to $-\pi/2$. To explain our method, we consider the angle $\theta_i = \theta(i-1,i,i+1)$ derived via Eq. (6), which we can write as

$$\theta_i = s_i|\theta_i|, \quad s_i = \mathrm{sign}(\theta_i), \quad i = 2, \ldots, n_p - 1$$

The two quantities s_i and $|\theta_i|$ are given by the following relations,

$$s_i = \mathrm{sign}\,(w_i)\,, \quad \text{where} \quad w_i = (v_{i-1,i})_1(v_{i,i+1})_2 - (v_{i-1,i})_2(v_{i,i+1})_1$$

$$|\theta_i| = \arccos\, v_{i-1,i}^T v_{i,i+1},$$

which follows from elementary geometry. Here, $(z)_l$ denotes coordinate l of the vector z. The corner is then defined by $k = \mathrm{argmin}_i\ |\theta_i + \pi/2|$. We note as an implementational detail that regarding the wanted minimum, w_i carries sufficient information such that $k = \mathrm{argmin}_i\,|w_i + 1|$.

If θ_k (or equivalently w_k) is negative, then the point $\mathcal{P}_k$ is accepted as a corner. Otherwise, the given pruned L-curve is considered flat or concave and no corner is found.

4.3 Corner Location Based on Global Behavior

The approach used here is similar to an idea from [1] in which the corner of the continuous Tikhonov L-curve is defined as the point with smallest Euclidian distance to the "origin" $\mathcal{O}$ of the coordinate system. With $\mathcal{O}$ chosen in a suitable way, it is shown in [1] that the point on the L-curve closest to $\mathcal{O}$ is near the point of maximum curvature.

The main issue is to locate a suitable "origin". In [1] it is defined as the point $(\log \|A\, x_{\sigma_n} - b\|_2\,,\ \log \|x_{\sigma_1}\|_2\,)$ where x_{σ_1} and x_{σ_n} are the Tikhonov solutions for $\lambda = \sigma_1$ and $\lambda = \sigma_n$, respectively. But given only points on a discrete L-curve,

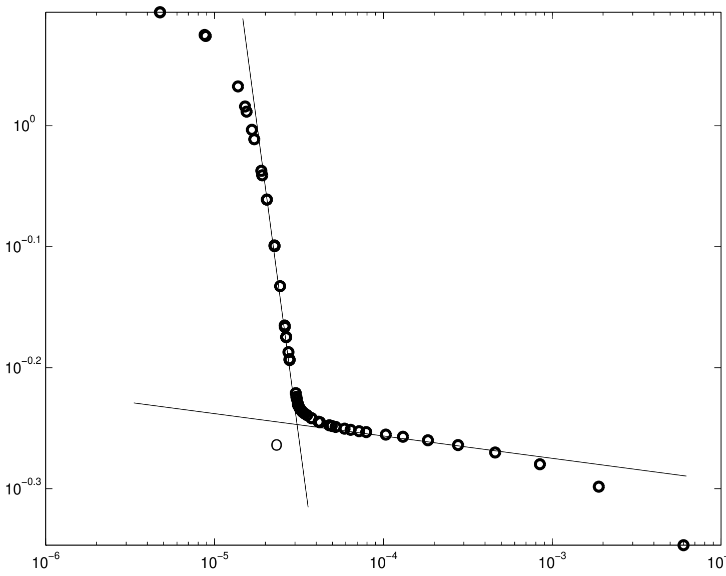


Fig. 6. Illustration of discrete L-curve with straight lines showing the global behavior. The “origin” $\mathcal{O}$ is chosen as the intersection between the two straight lines. The corner, in turn, is then chosen as the point with smallest Euclidian distance to $\mathcal{O}$.

neither the singular values nor estimates are necessarily known. Instead we seek to identify the “flat” and the “steep” parts of the L-curve, knowing that the corner must lie between these parts. More precisely, we seek to fit straight lines to the “flat” and “steep” parts, and then define $\mathcal{O}$ as the intersection between these lines. This idea is illustrated in Fig. 6 where the global behavior is indicated by straight lines and the point $\mathcal{O}$ is indeed located near the corner of the L-curve. Then the corner of the L-curve is chosen as the point with smallest Euclidean distance to $\mathcal{O}$.

Fitting straight lines to identify the overall horizontal and vertical parts of the L-curve is computationally nontrivial. Instead, we use the normalized vectors describing the pruned L-curve. Specifically, if we define the horizontal vector $v_H = (-1, 0)^T$ and p is the number of points on the pruned L-curve, then we first define the slopes ϕ_j as the angles between v_H and all the normalized vectors $v_{j-1,j}$ for $j = 2, \ldots, p$. Then the most horizontal line segment is identified by $\ell_h = \operatorname{argmin}_j |\phi_j|$ and the most vertical one by $\ell_v = \operatorname{argmin}_j |\phi_j + \pi/2|$. We note as an implementational detail that if we describe the slopes similar to the angles in Section 4.2 then $w_i = (v_H)_1(v_{i,i-1})_2 - (v_H)_2(v_{i,i-1})_1$ again carries sufficient information. With the above definition of v_H, we get $\ell_h = \operatorname{argmin}_j |(v_{j-1,1})_2|$, and $\ell_v = \operatorname{argmin}_j |1-(v_{j-1,1})_2|$, and the horizontal and vertical parts of the curve can in practice be found without computation.

Unfortunately, the most horizontal line segment might appear in the leftmost part of the L-curve, and the most vertical line segment might appear in the rightmost part. To ensure that the chosen line segments are good candidates for the global behavior of the L-curve, we add the constraint that the horizontal line segment must lie to the right of the vertical one. A worst-case scenario is a situation where all the more horizontal line segments lie to the left of all the more vertical ones – e.g., a perfectly well-posed problem. This situation results in an increased number of loops and comparisons but no additional floating point operations.

The "origin" $\mathcal{O}$ is now defined as the intersection between the horizontal line at $\log \|x_{\ell_h}\|_2$ and the line defined by the vector v_{ℓ_{v-1},ℓ_v}. Using the line defined by the vector v_{ℓ_{h-1},ℓ_h} instead of the strictly horizontal line will move $\mathcal{O}$ upwards, because v_{ℓ_{h-1},ℓ_h} (due to monotonicity of the norms) will always be declined compared to horizontal. Therefore we would increase the risk of selecting a corner slightly to the left of the true corner. Since it is often less critical to choose a point slightly to the right of the true corner, the first approach is used.

5 Numerical Tests and Examples

Here we illustrate the performance and robustness of our adaptive pruning algorithm for finding the corner of discrete L-curves, and we compare the algorithm to the two previously described algorithms: l_corner from [5] and the triangle method from [3]. To perform a general comparison of state-of-the-art methods, we also compare with the General Cross Validation (GCV) method, which tries to minimize the predictive mean-square error $\|A\,x_k - b^{\mathrm{exact}}\|_2$, where b^{exact} is the noise-free right-hand side. In case of TSVD, the parameter k chosen by the GCV method minimizes the function

$$\mathcal{G}_k = \frac{\|A\,x_k - b\|_2^2}{(n-k)^2}.$$

For white noise, minimizing the GCV function $\mathcal{G}_k$ corresponds well to minimizing the predictive mean-square error. But while the theory for GCV is well established, the minimum is occasionally very flat resulting in (severely) underregularized solutions. These problems are described, e.g., in [6, §§ 7.6–7.7].

5.1 Test Problems

We use a broad selection of standard test problems from Regularization Tools [5] as well as problems with ill-conditioned matrices from Matlab's "matrix gallery." In addition we use a test problem from [4]. When no exact solution is provided, the exact solution from the test problem shaw is used. The test problems are listed in Table 1.

Table 1
The test problems used in our comparison. All problems from REGULARIZATION TOOLS use the default solution, except ilaplace where third solution is used. All "gallery" matrices use the exact solution from the shaw test problem. To obtain a coefficient matrix that represents a discrete ill-posed problem, prolate is called with parameter 0.05.

Number	1	2	3	4	5	6	7	8	9	10	11	12	13
Name	baart	shaw	wing	hilbert	lotkin	moler	foxgood	gravity	heat	ilaplace	phillips	regutm	prolate
Type	Reg. Tools			"gallery"			Reg. Tools						"gallery"

All test problems consist of an ill-conditioned coefficient matrix A and an exact solution x^{exact} such that the exact right-hand side is given by $b^{\text{exact}} = A\,x^{\text{exact}}$. To simulate measurement errors, the right-hand side $b = b^{\text{exact}} + e$ is contaminated by additive white Gaussian noise e scaled such that the relative noise level $\|e\|_2/\|b^{\text{exact}}\|_2$ is fixed. The TSVD method is used to regularize all test problems. To evaluate the quality of the regularized solutions, we define the best TSVD solution as the solution x_{k^*} where k^* is given by

$$k^* = \text{argmin}_k \frac{\|x^{\text{exact}} - x_k\|_2}{\|x^{\text{exact}}\|_2}.$$

For problem sizes of $n = 64$ and $n = 128$, and a relative noise level of $\|e\|_2/\|b^{\text{exact}}\|_2 = 5 \cdot 10^{-3}$, all test problems are generated with 8 different realizations of the noise. Let $i = 1, \ldots, 13$ denote the problem and $j = 1, \ldots, 8$ the realization number. For each i and j, we compute the optimal parameter k^*_{ij} as well as k^{L}_{ij} from l_corner, k^{G}_{ij} from the GCV method, k^{T}_{ij} from the triangle metod, and k^{A}_{ij} from the new adaptive pruning algorithm.

The quality of all the solutions are measured by the quantity

$$Q^{\square}_{ij} = \frac{\|x_{k^{\square}_{ij}} - x^{\text{exact}}\|_2}{\|x_{k^*} - x^{\text{exact}}\|_2}, \qquad \square = \text{A, L, G, and T},$$

where A, L, G, and T refer to adaptive pruning algorithm, l_corner, GCV method and triangle method, respectively. The minimum value $Q^{\square}_{ij} = 1$ is optimal, and a value $Q^{\square}_{ij} > 100$ is considered off the scale.

Figures 7 and 8 show the quality measure for all tests. In some occasions, k^L_{ij} and k^G_{ij} produce regularized solutions that are off the scale; this behavior is well-known because the spline might fit the local behavior of the L-curve and thus find corners far from the global corner, and the GCV-function can have a very flat minimum. The new pruning algorithm is never off the scale, and the triangle method is only far off in test problem 9 for problem size $n = 128$. On the other hand, the triangle method seems slightly better than the new pruning algorihtm for test problem eleven. Overall, both algorithms perform

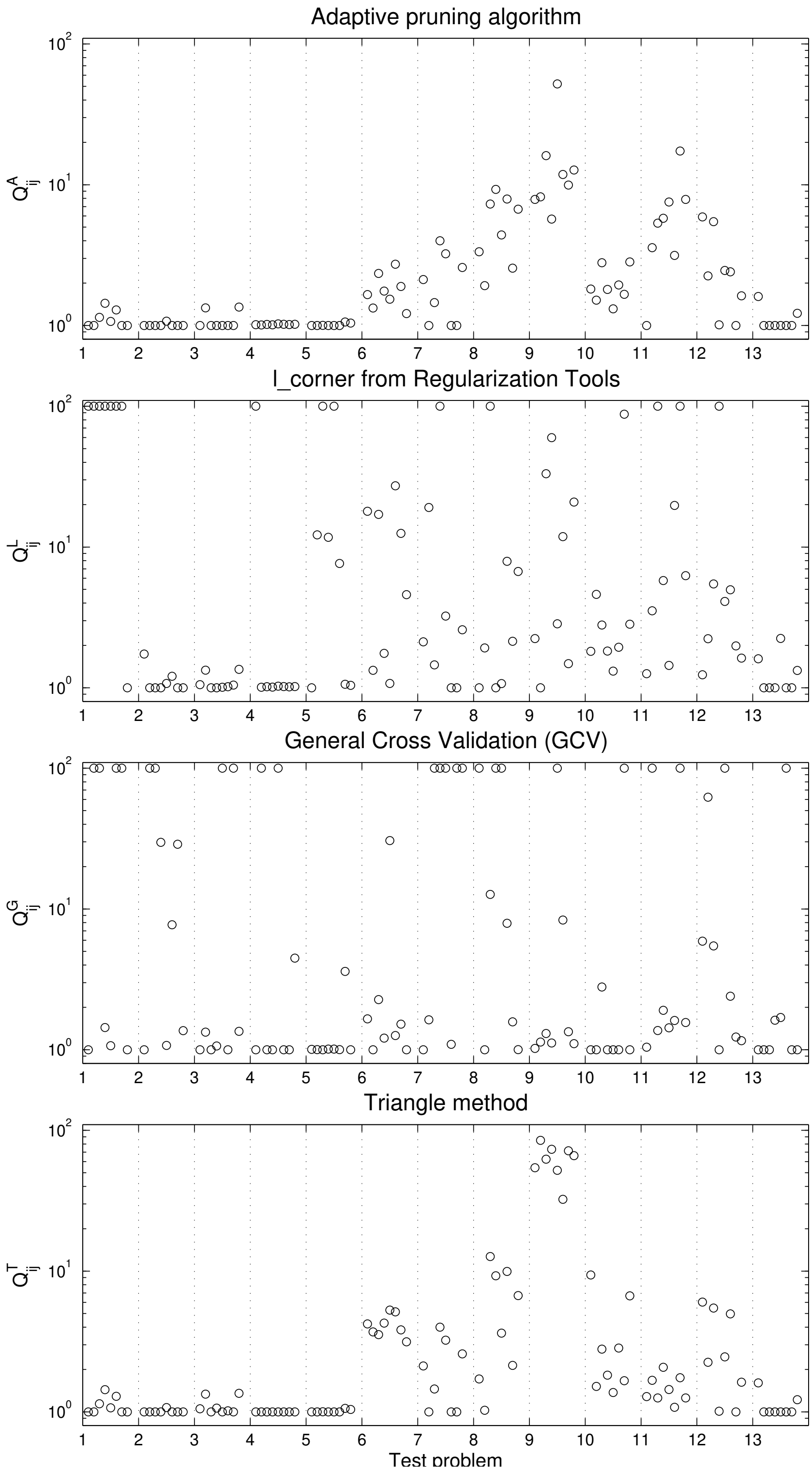


Fig. 7. Quality measure $Q^{\square}_{ij}$ for the four methods and all 13 test problems, with 8 realizations of the noise for a problem size of $n = 64$. A measure of one is optimal, and all values above 10^2 are set to 10^2.

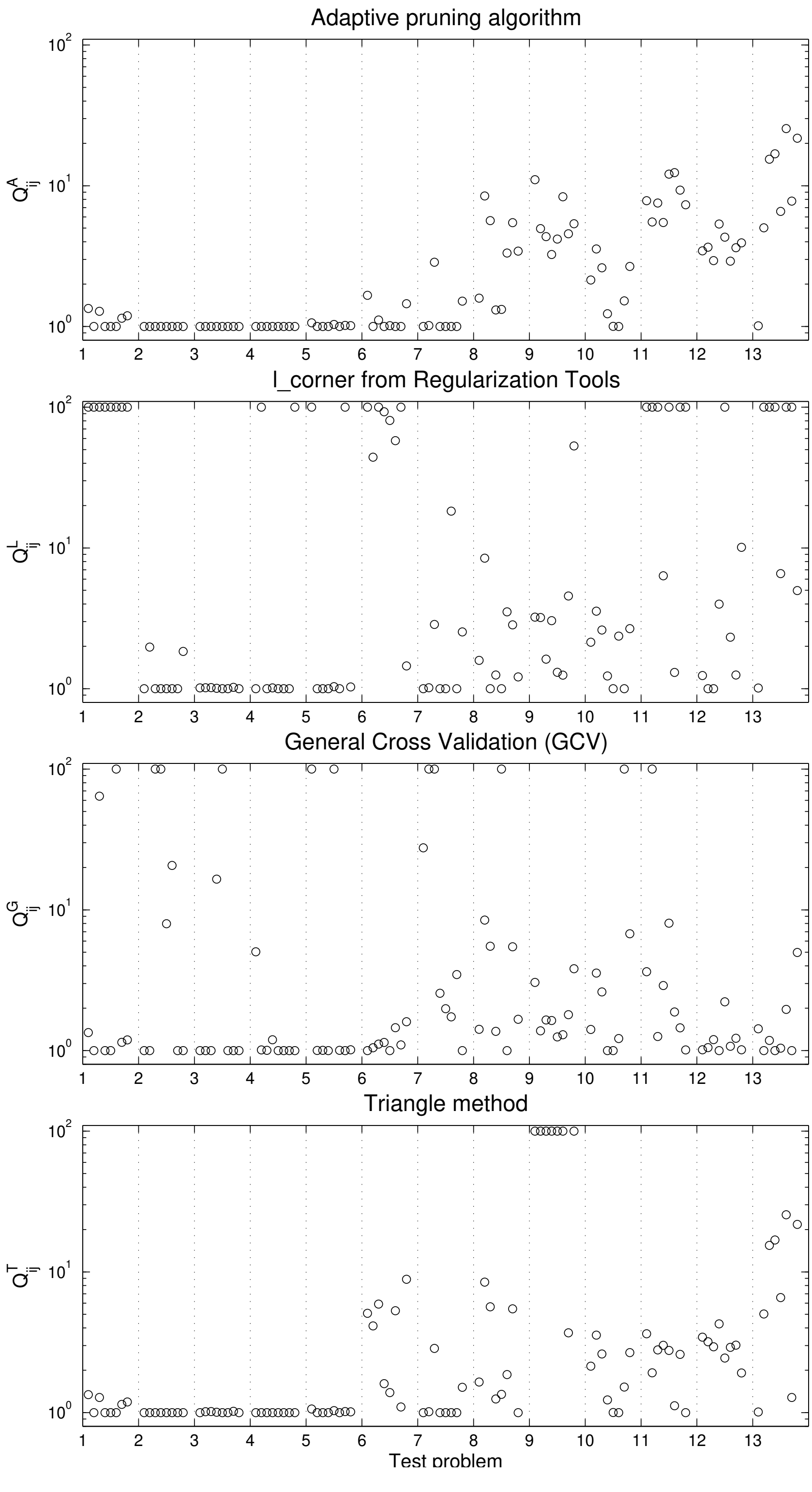


Fig. 8. Quality measure $Q^{\square}_{ij}$ similar to Fig. 7, but with problem size $n = 128$.

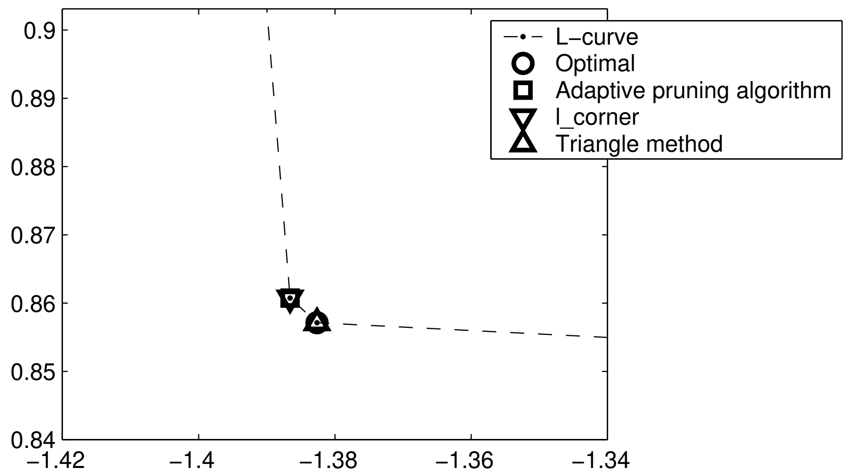


Fig. 9. “Nice” L-curve for problem $(i, j) = (4, 8)$. The corner is simple, and the optimal solution lies in the corner.

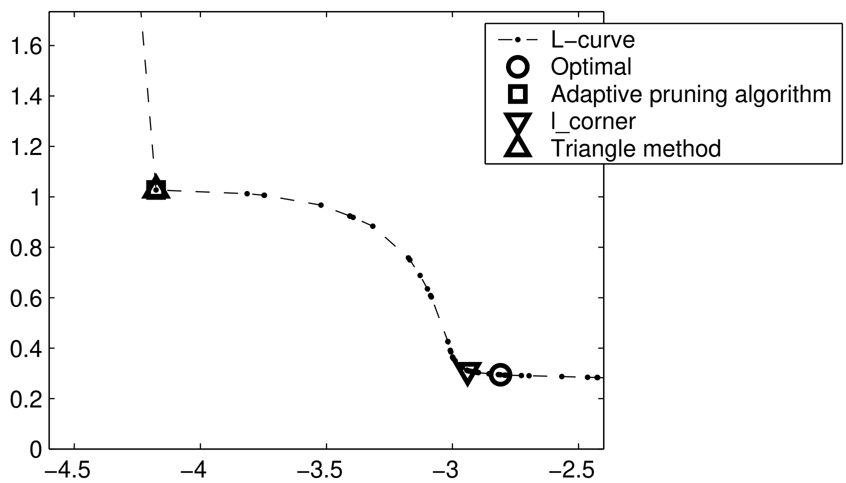


Fig. 10. Problematic L-curve for problem $(i, j) = (5, 9)$. The curve has no simple corner, and the optimal solution lies neither in the corner nor near the corner.

equally good.

It is interesting to observe that GCV behaves somewhat similar for all test problems, whereas the three L-curve algorithms seem to group the problems into two groups: one group that seems easy to treat, and one group where the L-curve criterion seems likely to fail. The effect is less significant for $n = 128$, where also GCV seems to favorize some of the test problems. To illustrate these different cases, Figs. 9 and 10 show the corner of the L-curves of the eighth realization of test problem four, $(i, j) = (4, 8)$, and the fifth realization of test problem nine, $(i, j) = (9, 5)$, both of size $n = 64$. The former is an “easy” problem where all three L-curve algorithms work well, and the latter is a problem where all three algorithms fail.

It is obvious from the figures that test problem four gives rise to a simple L-curve with a corner consisting of merely two points, of which one represents the optimal solution. The error is small for both points. The large error for $(i, j) = (4, 1)$ for l_corner (see Fig. 7) is due to an unwanted corner of the fitted spline curve, which lead to the large error.

The other example is more interesting. This L-curve exhibits two corners of which the pruning algorithm chooses the wrong one, leading to large errors. Moreover, the optimal solution does not lie exactly in the other corner. Thus, although $k^L_{9,5}$ corresponds to a point near the corner, the optimal solution lies slightly off the corner on the horizontal branch of the L-curve. This behavior is a more serious problem with the L-curve heuristic as mentioned in Section 3.

To analyze the problem in more detail, Fig. 11 shows the Picard plot as well as plots of the residual and solution norms. As anticipated, we see that the singular values decay very slowly in the part of the spectrum where the noise is significant. This means that the solution norm increases very slowly while the residual norm slowly levels off. This leads to a large transition area between the region with fast decaying residual norm and the region with fast increasing solution norm, and therefore a "corner" consisting of many points. Most of the contributions to the solution in this transition area are due to inverted noise, and the optimal solution lies in left part of the transition area far before the solution norm starts to increase, as seen in Fig. 11. This illustrates that the optimal solution might lie to the right of the corner of the L-curve, which is actually the case in Fig. 10.

The tests illustrate that the new adaptive pruning algorithm is more robust than the l_corner algorithm and the GCV method, and that it performs similar to the triangle method. The tests also illustrate that we cannot always expect to get the optimal regularization parameter by using the L-curve criterion, as this optimum is not always identified as the corner of the L-curve. It is noted that all three L-curve algorithms have some difficulties with test problems $i = 6, 7, \ldots, 12$.

As mentioned earlier, the most serious problem with the triangle method is the complexity of $\mathcal{O}(n_p^2)$ whereas the adaptive pruning algorithm tends to have an overall complexity $\mathcal{O}(n_p \log n_p)$. To illustrate this fact, all thirteen test problems have been run with noise levels $\|e\|_2/\|b^{\text{exact}}\|_2 = 5 \cdot 10^{-2}$, $\|e\|_2/\|b^{\text{exact}}\|_2 = 5 \cdot 10^{-3}$ and $\|e\|_2/\|b^{\text{exact}}\|_2 = 5 \cdot 10^{-4}$ varying the problem size from $n_p = 16$ to $n_p = 128$ and the number of floating point operations has been approximately recorded. The result is shown in Fig. 12 (a) for the adaptive pruning algorithm and the triangle method, showing the average over the three noise levels and all test problems. We see that the complexity of the triangle method is about $3n_p^2$, whereas the complexity of the adaptive pruning algorithm is about $25 n_p \log n_p$.

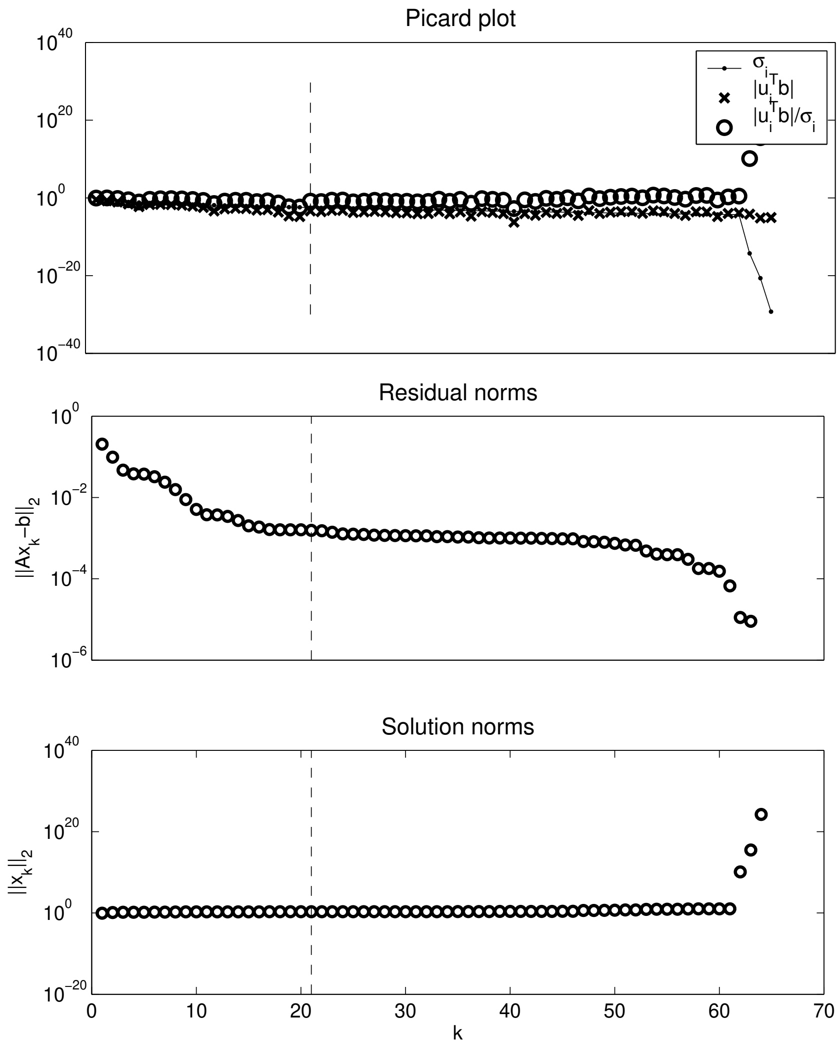
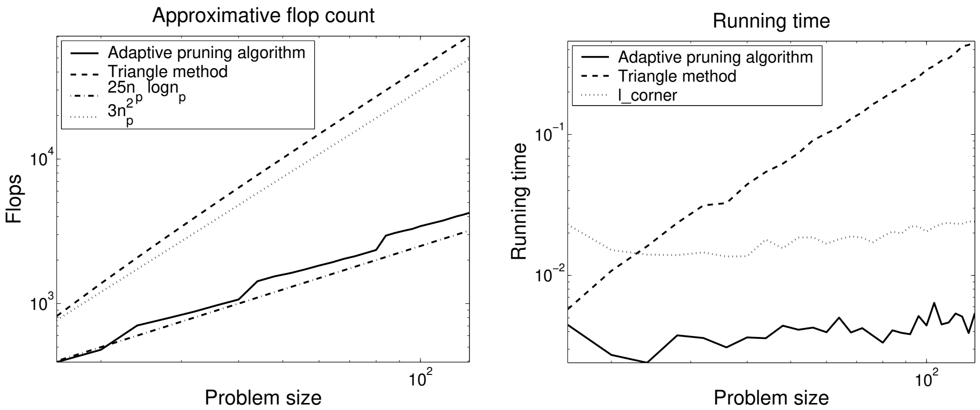


Fig. 11. Example of Picard plot (top) and illustration of decay of $\|A\,x_k - b\|_2$ and increase of $\|x_k\|_2$ (bottom) for problem $(i, j) = (5, 9)$.

To include also the l_corner, we show in Fig. 12 (b) a graph of the average running times. This measure is very sensitive to implementation details, but show the same trend as the approximative flop count. Furthermore, the latter figure shows that the adaptive pruning algorithm is faster than l_corner from Regularization Tools.

5.2 L-Curves for Large-Scale Problems

For illustrative purposes we also show a large-scale example in the form of an image deblurring problem. Figure 13 shows the exact image X of size 100×100, together with a blurred and noisy image B. The blurring is spatially invariant and seperates into column and row blur, and zero boundary conditions are used in the reconstruction. This leads to a formulation of the problem of the



(a) (b)

Fig. 12. (a) Illustration of approximative flop count for the adaptive pruning algorithm and the triangle method. (b) Illustration of running times for the three L-curve algorithms.

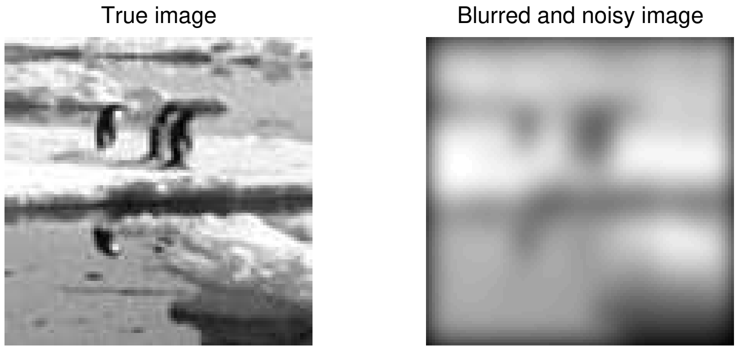


Fig. 13. Exact image and blurred-and-noisy image.

form

$$Kx = b,$$

where K is a Kronecker product of two Toeplitz matrices, and x and b are the columnwise stacked images X and B. The construction of the blurring operator is described in [9, Appendix], and the parameters used are $\sigma_c = \sigma_r = 5$, $\alpha_c = 0$, and $\alpha_r = 1$. This leads to a $10^4 \times 10^4$ nonsymmetric coefficient matrix K. For the reconstruction we use CGLS with full reorthogonalization.

The CGLS L-curve for the image problem is shown in Fig. 14, and we see that both the adaptive pruning algorithm and l_corner find points close to the true corner of the L-curve. The triangle method erroneously identify a corner far off on the horizontal branch of the L-curve. Furthermore, the running time for the triangle method is much larger than for the other L-curve algorithms due to the $\mathcal{O}(n_p^2)$ complexity. To illustrate this fact, a simple timing of the methods shows a running time of approximately 28 seconds for the triangle method compared to about half a second for the adaptive pruning algorithm and the l_corner algorithm, using a laptop with a Pentium Centrino 1.4GHz

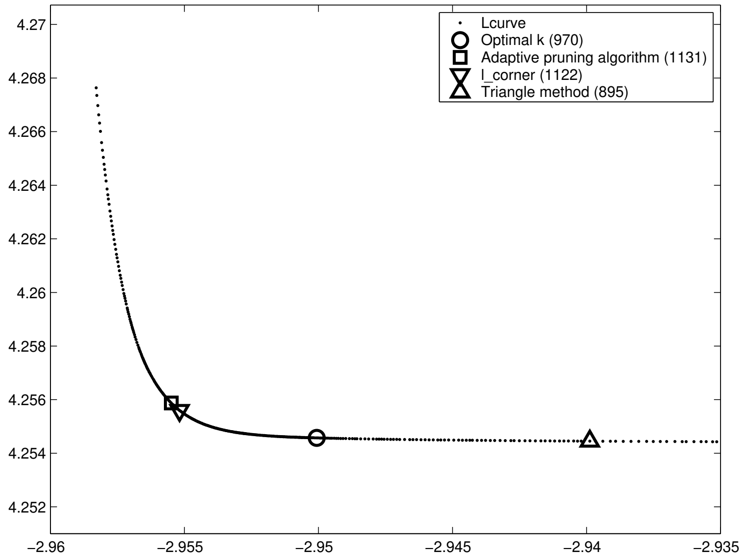


Fig. 14. Corner part of L-curve for large-scale image reconstruction problem. The optimal solution lies on the horizontal part of the curve to the right of the corner, and denoted by a circle. The legend shows in parentesis the corresponding number of CGLS iterations.

processor. The GCV function is not well-defined for CGLS and is therefore not applicable here.

As anticipated in Section 3, it can be problematic to use the L-curve criterion for large-scale problems when the singular values decay slowly over the entire spectrum (or a large portion of it). Such a slow decay results in a large corner region, and probably the optimal solution does *not* lie near the point of maximum curvature. For the problem considered here, the optimal solution lies in the right part of the corner of the L-curve, and due to the large problem dimensions the difference corresponds to more than 150 CGLS iterations.

Although these extra iterations do not increase the solution norm dramatically, there are several problems. The extra iterations waste both time and memory, and all the extra iterations include inverted noise that does not carry useful information about the wanted solution. In fact, the inverted noise exhibits some artificial structures which becomes increasingly disturbing to the human eye as more iterations are performed. To illustrate the degradation of the solution when going from the optimal solution to the solution in the corner of the L-curve, Fig. 15 shows the optimal solution and the solution found by the adaptive pruning algorithm. We clearly see the visual effects of performing too many iterations.

This example demonstrates that the new pruning algorithm is able to find a point near the corner, even for large-scale problems; but also that this corner is maybe not a good choice for a regularization parameter.

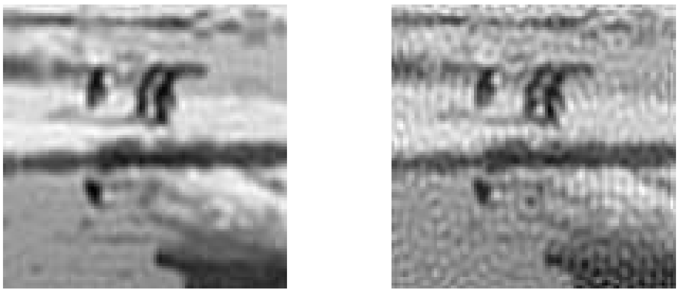


Fig. 15. Illustration of degradation in going from the optimal solution to the "corner" solution from Fig. 14.

6 Application of the Adaptive Pruning Algorithm

Since the proposed algorithm can find the corner of a discrete L-curve very confidently, the same method can also be used to find an approximation to the Tikhonov regularization parameter λ via the corresponding continuous L-curve. By evaluating points on the continuous L-curve for a limited set of λs, a crude approximation to the Tikhonov corner can be found. By successively refining the resolution of the L-curve around the found corner, the regularization parameter λ can be found within a certain precision using only a limited number of computed L-curve points. This approach is particularly favorable when a functional expression for the continuous L-curve is not present such that the maximum curvature is not directly computable. Furthermore, if each solution of the problem for a given λ is expensive, we do not want to solve too many Tikhonov systems, which is avoided by only enhancing the resolution around the optimal discrete approximation to the corner.

Connected to the above, another application is to use the global view on the horizontal and vertical parts of the L-curves to perhaps define a better "origin" of the logarithmic coordinate system in connection with the algorithm described in [1] where some "origin" is needed to find the corner of a continuous L-curve.

7 Conclusion

We described a new adaptive algorithm for finding the corner of a discrete L-curve. Numerical examples show that the algorithm is faster and more robust than previous algorithms. It is also shown that some L-curves are not well-behaved due to certain properties of the SVD coefficients. In these cases any L-curve algorithm will fail to find the exact optimal solution no matter how it is implemented, and the proposed algorithm is in these cases seen to perform at least as good as previous L-curve algorithms.

References

[1] M. Belge, M. E. Kilmer and E. L. Miller, *Efficient determination of multiple regularization parameters in a generalized L-curve framework*, Inverse Problems, 18 (2002), pp. 1161–1183

[2] D. Calvetti, P. C. Hansen and L. Reichel, *L-curve curvature bounds via Lanczos bidiagonalization*, Electronic Trans. Numer. Anal., 14 (2002), pp. 134–149.

[3] J. L. Castellanos, S. Gómez and V. Guerra, *The triangle method for finding the corner of the L-curve*, Appl. Num. Math., 43 (2002), pp. 359–373.

[4] P. C. Hansen, *Deconvolution and regularization with Toeplitz matrices*, Numer. Algo., 29 (2002) pp. 323–378.

[5] P. C. Hansen, *Regularization Tools: A Matlab package for analysis and solution of discrete ill-posed problems*, Numerical Algorithms, 6 (1994), pp. 1–35.

[6] P. C. Hansen, *Rank-Deficient and Discrete Ill-Posed Problems: Numerical Aspects of Linear Inversion,* SIAM, Philadelphia, 1998.

[7] P. C. Hansen, *The L-curve and its use in the numerical treatment of inverse problems;* invited chapter in P. Johnston (Ed.), *Computational Inverse Problems in Electrocardiology*, WIT Press, Southampton, 2001; pp. 119–142.

[8] 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 (1993), pp. 1487–1503.

[9] T. K. Jensen and P. C. Hansen, *Regularizing iterations for image deblurring*, Manuscript in preparation.

[10] G. Rodriguez and D. Theis, *An algorithm for estimating the optimal regularization parameter by the L-curve*, Rend. di Matematica, 24 (2004), to appear.