S0168-9274(02)00147-2/FLA AID:1388 Vol.●●●(●●●)
ELSGMLTM(APNUM):m2 2002/05/06 Prn:6/05/2002; 12:54
ANM1388
P.1(1-15)
by:ES p. 1

ELSEVIER

Applied Numerical Mathematics ●●● (●●●●) ●●●–●●●

APPLIED NUMERICAL MATHEMATICS

www.elsevier.com/locate/apnum

Large scale least squares scattered data fitting

V. Pereyra *, G. Scherer

Weidlinger Associates, 4410 El Camino Real #110, 94022 Los Altos, CA, USA

Abstract

The least squares approximation by tensor products of B-splines of large sets of scattered data is considered. This ill-conditioned or even singular problem requires special techniques, some of which are described in this paper. The performance of a Block Truncated Singular Value Decomposition (BTSVD) algorithm and two Lanczos based algorithms for sparse LSQ is compared. Several regularization methods are discussed. © 2002 IMACS. Published by Elsevier Science B.V. All rights reserved.

1. Introduction

In [19] we have considered the least squares fitting of scattered data by tensor products of B-splines. The main thrust in that paper was to use the Truncated Singular Value Decomposition (TSVD) as the tool to regularize the ill-conditioned or even singular problems that resulted when large zones with missing data existed. There, "large" meant relative to the spacing of the B-spline basis functions.

In this paper we revisit the theme, but now the emphasis is on large scale problems, i.e., those problems where there are many data points and basis functions and for which direct TSVD algorithms would be inadequate. These types of problems occur in high resolution surface or image fitting, in lossy compression, and they are even more prevalent in three-dimensional situations, where the data may correspond to volumetric properties or video sequences.

The method we advocate does neither involve regridding, nor manual or automatic segmentation of the data. Although we prefer to limit its application to regular data, possibly contaminated with a small amount of Gaussian error, we show that it also works reasonably well in the presence of discontinuities and sharp gradients.

We consider a block (domain decomposition) BTSVD approach, in which the data domain is partitioned in sufficiently small subdomains, in order to make a TSVD algorithm practical, and also

* Corresponding author.
E-mail address: victor@ca.wai.com (V. Pereyra).

0168-9274/02/$ – see front matter
PII: S0168-9274(02)00147-2

to facilitate parallelization. The blocks are visited sequentially in a block Gauss–Seidel (BGS) iteration, where a master copy of all the control vertices is updated as soon as a block fit has been completed. An asynchronous parallel version is also possible, appealing to the theory of chaotic iterations for its convergence [18].

We also consider the algorithm LSQR of Paige and Saunders [17]. This is used in two ways. First, to solve the global problem (i.e., without partitioning), in order to have a baseline of comparison with an existing method. Then we propose a combination with the BTSVD approach. We use LSQR at the block level to initialize the control vertices and then switch to the TSVD solver for subsequent sweeps through the blocks.

Finally, we compare with an iterative version of a global TSVD, based on the Lanczos algorithm, that preserves the inherent sparseness of the problem.

An important issue that comes into play is regularization, since there is a good likelihood that the problem will be ill-conditioned. That was the original reason for the choice of a TSVD algorithm. LSQR also has some natural regularization capabilities if interrupted prematurely. The challenge here is to see if we can devise a reliable, automatic implementation, in order to produce acceptable data approximations over the domain of interest.

2. Surface fitting

We assume that a data set $\{x_i, y_i; z_i\}$, $i = 1, \ldots, m$, representing a smooth, single valued function is given. The points need not lie on a regular mesh, i.e., they can be scattered. For approximant functions we consider tensor products of cubic B-splines, although any appropriate set of basis functions could be used:

$$B(x, y) = \sum_{1 \leqslant j \leqslant J} \sum_{1 \leqslant k \leqslant K} C_{jk} B_j(x) B_k(y). \tag{2.1}$$

The basis functions are associated with a uniform mesh subdivision (sx_j, sy_k), $j = 1, \ldots, J$; $k = 1, \ldots, K$, of a rectangle D that contains all the data points. This rectangle is arbitrary, but it should not be made unnecessarily large.

The least squares problem seeks to find a spline that best fits the data in the l_2 norm:

$$\min_{C_{jk}} \sum_i \sum_{j,k} \big(C_{jk} B_j(x_i) B_k(y_i) - z_i\big)^2,$$

where we assume that $n = J \times K \ll m$.

By storing the elements C_{jk} in an 1-D array C, varying first the index k, and organizing the elements of the least squares coefficient matrix, $B_j(x_i)B_k(y_i)$, consistent with this convention, the least squares problem can be written in matrix notation:

$$\min_C \|BC - Z\|_2^2, \tag{2.2}$$

where B is the $m \times n$ Grammian and Z is the data vector $\{z_i\}$.

As we recall, in the case of B-splines basis functions, only few control vertices in each coordinate direction are active at any given evaluation point, so the matrix B is sparse. In addition to that, if there are holes, or regions with not enough data, the problem can be ill-conditioned and even singular.

3. Algorithms for least squares computation

Given the general least squares problem defined by Eq. (2.2) we will consider a block TSVD and two Lanczos based iterative methods of solution.

3.1. Block TSVD

An important tool in least squares problems is the singular value decomposition of $B : B = U\Sigma V^{\mathrm{T}}$, with U, V, orthogonal matrices and $\Sigma = (\sigma_1, \ldots, \sigma_r, 0, \ldots, 0)$, where r is the rank of B. The least squares solution C_{LS} is then given by:

$$C_{LS} = \sum_{k=1}^{r} \left(u_{k+1}^{\mathrm{T}} Z/\sigma_{k+1}\right) v_{k+1}. \tag{3.3}$$

Both the solution and the residual norm, $\rho^2 = \|BC - Z\|_2^2$, can be computed efficiently by a recursion. If C_k and ρ_k^2, are the current iterates, then:

$$C_{k+1} = C_k + \left(u_{k+1}^{\mathrm{T}} Z/\sigma_{k+1}\right) v_{k+1}, \tag{3.4}$$

$$\rho_{k+1}^2 = \rho_k^2 - \left(u_{k+1}^{\mathrm{T}} Z\right)^2, \quad k = 1, \ldots, r. \tag{3.5}$$

An approximate regularized solution of the least squares problem can be obtained by truncating these recursions after a certain $\bar{k} \leqslant r$. The resulting procedure is called TSVD.

To implement the method, the SVD is computed via Chan's [7], version that preprocesses the coefficient matrix using Householder transformations to obtain an upper triangular form. The control is then transferred to the Golub–Kahan–Reinsch algorithm for determining the SVD via Householder bidiagonalization and an iterative implicit-shift QR method applied to $B^{\mathrm{T}}B$ [8].

For large problems the SVD computation is prohibitive, but due to the local character of the B-splines, this global LSQ problem can be partitioned into smaller subproblems corresponding to a domain decomposition of the data set and the control vertices. A description of this so-called block TSVD approach, similar to the one used in [18] was given in [19].

The local subproblems have some overlap due to the phantom vertices, i.e., the control vertices associated with the two outermost B-spline basic functions in each direction. Among the several possibilities to determine a value for these control vertices in the master copy, the most convenient seems to be, either to assign a dynamic mean of the local values or, simply to overwrite them with the last computed values, unless the phantom control vertices are obtained from an ill-defined region with few data. The goal is to devise a patching strategy so that the global approximation is C^1.

3.2. Iterative Lanczos based methods

One can take advantage of the sparsity of the coefficient matrix by using iterative methods based on the Lanczos algorithm. The solution will only be approximate, and it is not always clear which and how regularization techniques ought to be applied.

There are two possible approaches to solve the least squares problem based on Lanczos iterations:

1. Sparse SVD via an eigenproblem. A first possibility is based on the relationship between the SVD of the matrix B and the Schur decomposition of associated matrices. We preferred the symmetric matrix $\mathcal{B} = \begin{pmatrix} 0 & B \\ B^{\mathrm{T}} & 0 \end{pmatrix}$, over the use of the more compact associated matrices $B^{\mathrm{T}}B$ or BB^{T}, because of the condition of the eigenvalue problem. It should be pointed out that the method is only efficient if at most roughly one fourth of the eigenpairs are required.

When applying the Lanczos algorithm to the matrix $\mathcal{B}$, one can generate a sequence of progressively better estimates of the extremal eigenvalues (and corresponding vectors) for $\mathcal{B}$. The convergence will depend on the separation ratios of the eigenvalues [1,21]. The singular values and vectors of B can then be recovered from these eigenpairs.

2. Bidiagonalization and QR. From a Lanczos iteration with starting vector $Z/\|Z\|_2$, one can obtain a bidiagonalization of the rectangular $m \times n$ matrix B into orthogonal and (for efficiency and stability [13, p. 149]) upper bidiagonal matrices $B = U\binom{G}{0}V^{\mathrm{T}}$, U and V orthogonal, G bidiagonal. This approach to the solution of the least squares problem is employed in [4,15].

Using the partial factors U_k, V_k, and G_k, it is then possible to substitute the original $m \times n$ LSQ problem:

$$\min_C \|BC - Z\|_2,$$

with a sequence of $(k+1, k)$ bidiagonal ones:

$$\min_{y_k} \left\| G_k y_k - \|Z\|_2 e_1 \right\|_2.$$

Here $e_1 = (1, 0, \ldots, 0)$.

Once these are solved via QR, successive approximations to C can be recovered from $c_k = V_k y_k$. The algorithm works best when B is well conditioned and has many nearly similar singular values. The residuals associated to the sequence of approximations $\{c_k\}$ are monotonically decreasing. This behavior is important in connection with the use of the L-curve regularization criterion.

The different implementations available in the public domain, try to compute the eigenproblems or QR decompositions of successive approximating problems more efficiently, as well as taking care of the reorthogonalization of the Lanczos vectors for numerical stability. We have chosen as suitable:

(1) LASVD, Berry's program, based on LANSO designed by Parlett: it uses a single-vector Lanczos algorithm with Simon's selective reorthogonalization to calculate the most significant singular values-vectors pairs of matrix $\mathcal{B}$ [1].
(2) LSQR, Paige and Saunders's algorithm and implementation: it uses the Golub and Kahan bidiagonalization and a modified QR to compute the least squares solution [17].

4. Regularization

The approximation problem under consideration is often ill-conditioned and even rank-deficient. Thus, the usual algorithms are unstable. Regularization methods replace the original problem by a related better-conditioned one, so that the contribution of noise to the computed solution is lessened, trading-off the quality of the data fit for controlling the "size" of the regularized solution.

The methods in use for small problems are well known: Tikhonov regularization and the afore mentioned truncated SVD. These two techniques often give similar results (see [4, p. 512]). For TSVD the regularising parameter is $\bar{k}$, the number of terms used in the least squares solution, when written in terms of the SVD decomposition (see Eq. (3.3) with r replaced by $\bar{k}$).

For more details see [19]. It is of interest to point out that the truncation parameter $\bar{k}$ can be chosen with a strategy based either on the size of the accepted singular values or that of the spectral coefficients (Rust's strategy, see [20]).

For large problems, some iterative methods for the LSQ problem, among them projection methods such as, for example, LSQR, CG (see, for example: [12], and [15]), are self-regularising. Defined by Krylov iterations, they restrict the solution in the kth iteration to a k-dimensional subspace spanned by the right Lanczos vectors. Using the Courant–Fischer mini–max theorem one can see that the successive k-approximations are better conditioned than the original problem. The subspace dimension $\bar{k}$ acts now as the regularising parameter.

It is known from experience that the Krylov subspace algorithms produce early on regularized solutions $x(\bar{k})$ that capture the main components of the solution, i.e., the ones corresponding to the largest singular values. However, at later iterations, as the approximating problems converge to the original ill-conditioned one, more noise enters the solution even before all the large singular values have been captured.

As it is pointed out in [3], a complete understanding of the regularising effect of Krylov subspace methods is still lacking. Reliable stopping rules are imperative, since an overshooting of the optimum $\bar{k}$ brings with it a deterioration due, among other reasons, to a loss of orthogonality in the Lanczos vectors as the iteration progresses.

If the self-regularising effect of the projection alone is insufficient, hybrid methods can be used. Once the Krylov subspaces are large enough to capture the desired information in the data, regularization methods like TSVD or Tikhonov, are introduced for the now much smaller and bidiagonal matrices. For details see [13,15]. The disadvantage is that now two parameters have to be chosen: the number of iterations and the truncation parameter.

In each of the above described regularization techniques, some parameters must be selected. There are several possibilities, depending on the information available, and the L-curve technique seems to be the most versatile and robust. It determines the best parameter from log / log plots of the residual versus solution norm, for different values of the regularising parameter.

It can be used for a regularising method when the behavior, both of the norms of the solution and of the residual, is monotone. Often a log / log graph of these quantities has an L-shape, with the regularization error dominating the flat part and the perturbation error the steep one. The idea is to determine the parameter corresponding to the corner of the L-curve, which balances both errors in the computed solution, in order to minimize the total error (see [13, p. 176]). The technique fails though if the solution is smooth, i.e., if the solution is dominated by the first SVD components (see [14]).

The monotone behavior, decreasing for the residual norm, increasing for the solution norm, given increasing values of the parameter k, is obvious for the "exact" TSVD. (See Eqs. (3.4), (3.5).) This monotonicity also holds for the LSQR and CG algorithms (see [13, p. 142], or [17, p. 51]), and the L-curve can be applied for both selection processes.

If the approximation problem under consideration is rank-deficient but not otherwise ill-conditioned, it is worth trying out some less expensive regularising techniques. Rank-deficient problems of this type are characterized by a coefficient matrix with a cluster of small singular values and a well-defined gap

with the rest of the spectrum. It is therefore possible, in principle, to determine numerically the rank r of the matrix, and thus substitute the ill-conditioned problem with a better conditioned approximate problem, without using the above mentioned techniques. Notice that here the choice of the regularization parameter is independent of the right-hand side, depending only on the "geometry" of the data.

The numerical ε-rank, r_ε, is the number of linearly independent columns of B, with respect to any perturbation with norm ε. It is determined by the index of the singular value such that $\sigma_{r_\varepsilon} > \varepsilon \geqslant \sigma_{r_\varepsilon+1}$.

If r_ε is robust with respect to perturbations of the threshold and the singular values, i.e., if the gap is well defined, B can be said to be represented unambiguously by the principal part of the dyadic decomposition of B, B_{r_ε},

$$B = \sum_{k=1}^{r} u_k \sigma_k v_k^{\mathrm{T}} = B_{r_\varepsilon} + \sum_{k=r_\varepsilon+1}^{r} u_k \sigma_k v_k^{\mathrm{T}},$$

and the truncated solution $C_{\bar{k}}$ with $\bar{k} = r_\varepsilon$, is the appropriate regularized one.

From our earlier numerical experiments (see [19], examples D1–D4), the graphs (see Fig. 1) show that there are several clear gaps in the singular value spectrum, but they do not necessarily correspond to the above tentative threshold. Also the problems were still ill-conditioned when truncating at the probable numerical rank. We therefore implemented the L-curve criteria, using a simplified algorithm described below.

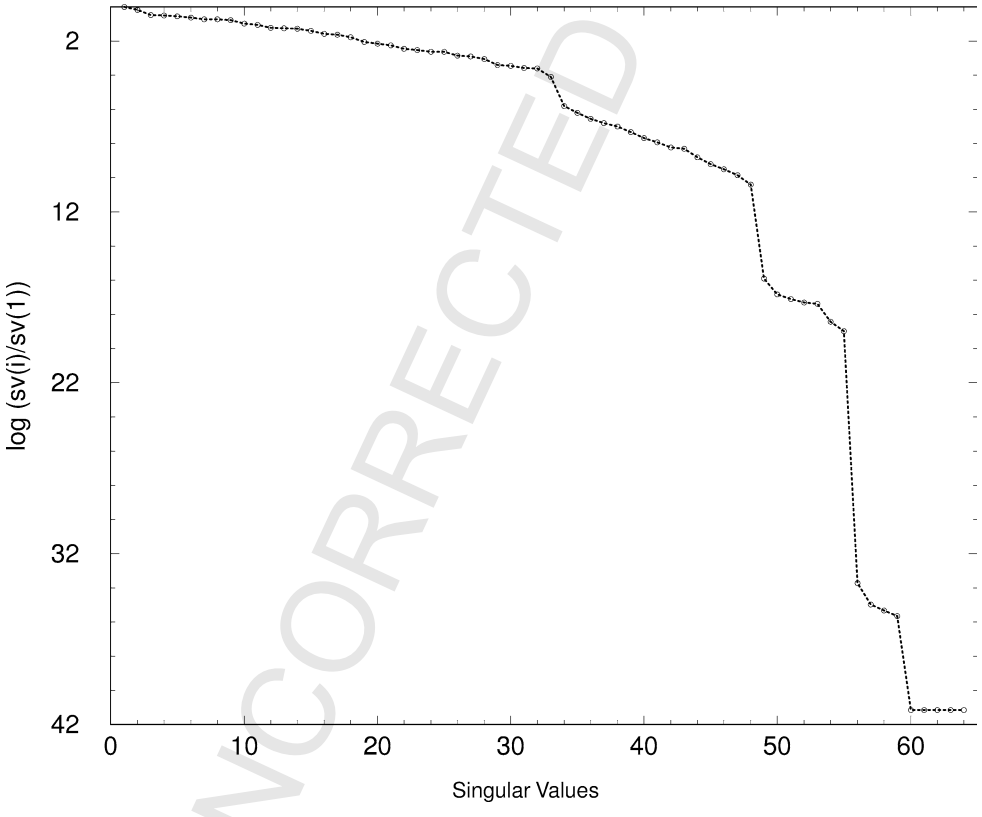

Fig. 1. Singular value distribution for problem D4.

Both the solution and the residual norm for the TSVD can be easily updated to obtain the values corresponding to a larger $\bar{k}$ (which corresponds effectively to a smaller filter) using Eqs. (3.4) and (3.5).

This leads naturally to a simplified "trial" strategy for determining the parameter value corresponding to the maximum curvature: starting from a tentative value of k, corresponding to a bound τ on the normalized singular values σ_k/σ_1, of say $\tau \simeq 10^{-4}$, successive approximations of C_k, ρ_k , are computed, until the desired solution quality is obtained.

For this strategy to succeed, it is essential that the plot of the solution versus the residual has in fact an L-shaped appearance with a sharp corner separating the steep and flat "monotone" parts. In practice this is not always the case as we have observed, for example, in some trials with LSQR. An algorithm such as the one developed by Gomez et al. [16] may be necessary in this case.

5. Numerical tests

5.1. Computational requirements and regularization

Although the efficiency of the algorithms defined in Section 3 cannot be compared theoretically due to the indetermination of the convergence, an analysis of the computational requirements gives a first idea on how their performance will compare in our approximation problem.

The following dimensions will be used throughout:

The matrix $B = U\Sigma V^{\mathrm{T}}$, $B_{m\times n}$, $U_{m\times m}$, $V_{n\times n}$, with m, total number of data points; and n, total number of control vertices;

The number of sub-domains used in BTSVD is s; each with a uniform local number of vertices l. Therefore $n \simeq sl$.

Flops and memory. These approximate flop counts are per iteration. Notice that BTSVD usually needs 2–3 iterations, whereas LSQR and LASVD may need up to $O(n)$. See Table 1.

The eigenpairs are computed with an implicit QL algorithm implemented as IMTQL2 in EISPACK.

Notice that the size/number of the subdomains plays an important part in the computing costs for the BTSVD. In fact, the best strategy seems to be, many subdomains with comparatively few vertices each, and to disregard altogether subdomains with too few data to be of relevance on the global approximation. On the other hand, the numerical errors in the approximation, resulting from patching vertices at the subdomain intersections, do not make it advisable to increase the amount of blocks too much.

Regularization. For BTSVD the regularization is done at the subdomain level, using a uniform threshold τ on the normalized singular values, for all subproblems. The regularization for LSQR is obtained by choosing the allowed number of iterations. The optimum parameter values for the numerical results shown below, were obtained for both methods manually by using partial L-curves plottings.

Table 1

	TSVD	BTSVD	LSQR	LASVD
Flops	$n^2(6m+20n)$	$l^2(6m+20n)$	$3m+7n$	$40(m+n)+eigenpairs$
Memory		$\sim 4ml/s$	$2m+3n$	$\sim (m+n)^2$

Stopping criteria and expected convergence for ill-conditioned systems. For BTSVD, the stopping criteria for the sweeps is a bound on the total RMS. For LSQR, Paige and Saunders specify a criterion (S3 in [17]) for ill-conditioned systems, which requires to choose the parameter CONLIM as an estimate of the cond(B):

STOP at iteration k if cond$(G_k) \geqslant$ CONLIM.

This heuristic should have a similar effect as TSVD with a singular value threshold of $\tau \approx 1/\text{CONLIM}$.

BTSVD usually requires 1–2 sweeps through all the subproblems, whereas LSQR, which in theory should need at most n iterations to converge, in practice needs far fewer or far more. In [17] it is mentioned that LSQR is most efficient if the problem is well-conditioned and has many nearly equal singular values. For most of our examples, roughly $n/3$ iterations were needed.

The algorithms were tested on both synthetic and real data sets.

5.2. Synthetic data tests

A set of tests were run using the synthetic domains D3 and D4, used in [19] to generate strongly ill-conditioned least squares problems. z values were defined by functions f3, f4 also of [19].

In the results below, RMS is used as a measure of the approximation quality. It is defined by:

$$RMS = \sqrt{\frac{\sum_1^m \rho_i^2}{m}}. \tag{5.6}$$

For the BTSVD routine, the normalized singular value threshold was chosen as $\tau = 10^{-4}$, and a subdivision of 2×2 was employed except for the last example were 5×5 subdomains were used. The iterations in LSQR were stopped using a value of the parameter CONLIM = 10^4, except for the last example where CONLIM $= 5 \times 10^2$. See Table 2.

Notice that for most examples, the value of RMS is slightly lower when using BTSVD than for LSQR, whereas the time ratio ranges from 1.2 to almost 5.

The behavior shown by BTSVD in the last example, to name but one of the tests run, stresses the need to generalize the algorithm to accept a non-uniform knot distribution, thereby saving on the costs of SVD computation in very sparse subdomains.

The graph in Fig. 1 shows the singular value distribution for D4.

It exemplifies the difficulty in determining the numerical rank and the need to use the more elaborated regularization techniques described above.

Table 2

Problem	Data	Vertices	BTSVD RMS	Time (sec.)	LSQR RMS	Time (sec.)
D3f3	4500	15×15	2.73×10^{-4}	17	3.22×10^{-4}	14
	9000	19×19	7.84×10^{-5}	45	9.26×10^{-5}	30
D4f3	9000	19×19	1.91×10^{-4}	45	2.29×10^{-4}	35
Noise	9000	19×19	5.43×10^{-3}	44	5.43×10^{-3}	30
D4f1	40000	46×46	1.9×10^{-3}	164	1.16×10^{-3}	34

The next step in the design of competitive algorithms for our problem must be the automatic selection of the optimum regularization parameter from an L-curve plot. Some preliminary tests for LSQR show that this will not always be a trivial task, because the L-curve has no sharp corner, indeed it has an inverted L shape. See, for example, the graph in Fig. 2, a plot for LSQR applied to the above D4f3 example with 9000 data.

The L-curve for BTSVD can also be plotted. Notice however that, although in the 1-subdomain case, monotonicity of the L-curve is assured, this is not the case for the block TSVD. Here the partial least squares problems are affected by the results and strategies applied in the neighboring subdomains. It does make sense then to use the L-curve applied to each subproblem. Here the L-curves are monotone, and for all the examples they had the characteristic shape.

We have run only some small test cases, comparing all three programs: BTSVD, LASVD and LSQR. The main reason being the large storage requirements of LASVD.

To give one representative example, for D4f3 with 4000 data points and 15×15 vertices the results are in Table 3.

They are clearly unfavorable to LASVD.

Table 3

Method	RMS	Time (sec.)
BTSVD	3.72×10^{-4}	13
LSQR	3.03×10^{-4}	13
LASVD	5.38×10^{-4}	245

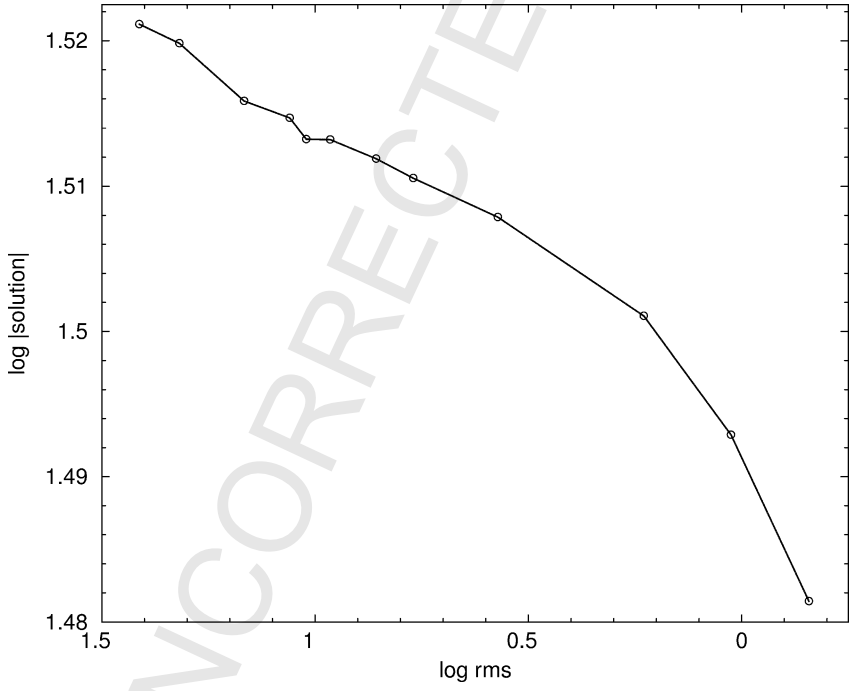

Fig. 2. L-curve for LSQR method.

5.3. SALT/SEG tests

A real data set identified by HRZ_4, was also tested extensively. It is a geological depth model of the boundary surface between two different sedimentary rocks pierced by a salt body (center hole). It includes a large normal fault that runs across the salt, from NW to SE, and some smaller faults that produce sharp gradients in the surface. See Fig. 3 for a plot of the data.

The set contains 71952 data points, with a depth range: $z \in [400, 17300]$.

The least squares approximation results in a rank deficient problem. The singular value spectrum follows a pattern similar to the one shown in Fig. 4, for a subset of the data.

The difficulties of the problem are related to the data "gaps" and in particular to the discontinuities in the depth values. The effect of these features on the quality of the approximation can be seen in the following tests, run with BTSVD, which compare the RMS for the original HRZ_4 set with a set which has the HRZ_4 "geometry", i.e., the same (x_i, y_i), but a smooth function value for the corresponding z_i,

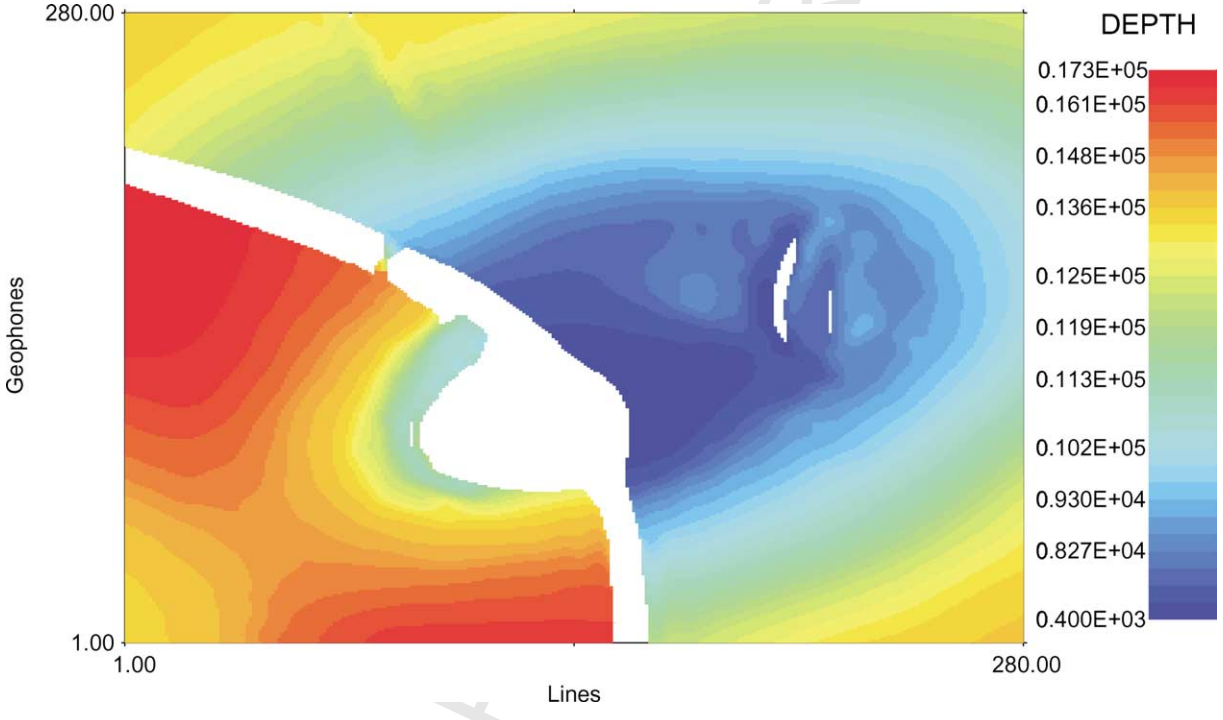

Fig. 3. SEG-EAEG salt model.

Table 4

Data set	RMS	Data (Error $\leqslant 0.5\%$)
HRZ_4	8.12×10^{-3}	91.4%
HRZ_4 geometry, smooth z	2.91×10^{-4}	99.91%
Random points, smooth z	8.41×10^{-5}	100%

and then, as a benchmark, with a random data distribution on the same rectangle with smooth function values (see Table 4).

In order to ascertain that the patching of the subregion leads to acceptable approximations, experiments were run to measure the approximation difference, when using the global spline approximation and when using the local subregion splines. Using a 5×5 domain decomposition leads to subdomains with varying number of data points and, as the knots distribution is uniform for all subdomains, to least squares subproblems with differing condition number. Table 5 shows the results for the worst and the best subregions.

Here, large residual means more than 1 Std. dev. from the mean.

Some representative results of the numerical experiments with BTSVD are the following. Using a 5×5 domain decomposition, with 10×10 local vertices, a common threshold $\tau = 10^{-4}$, and CONLIM $= 10^2$,

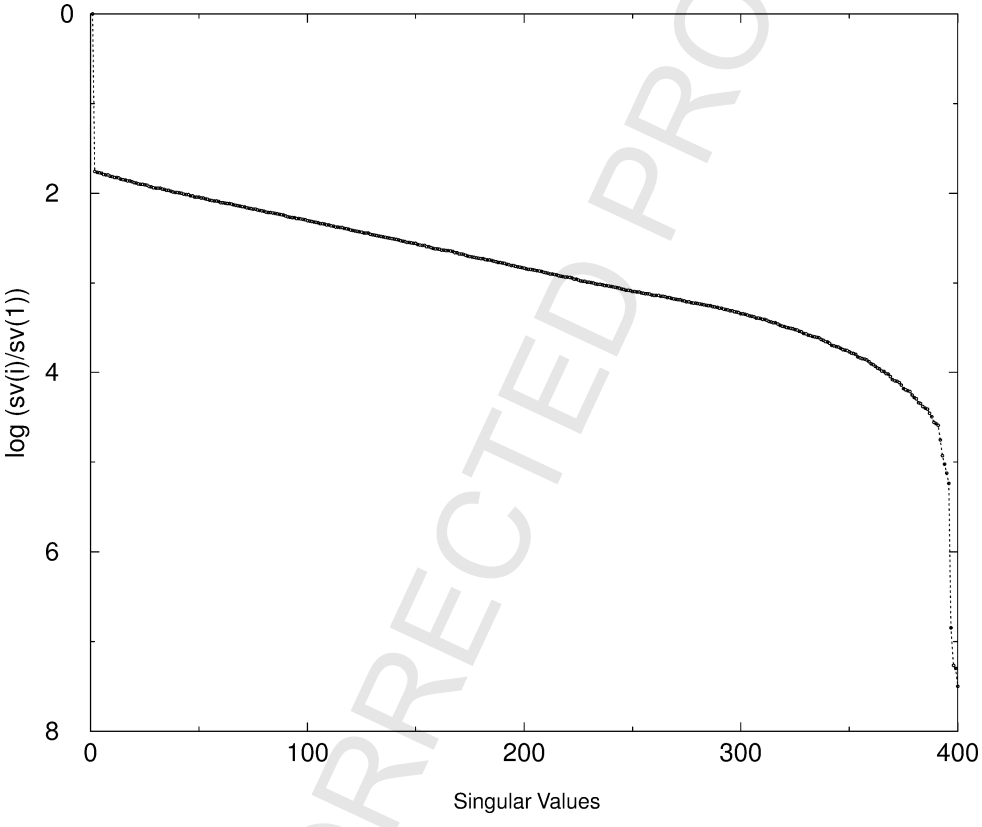

Fig. 4. Singular value distribution for salt model.

Table 5

Case	RMS local	RMS global	Large residual local	Large residual global
Worst	2.0×10^{-2}	3.8×10^{-2}	6%	10%
Best	1.33×10^{-4}	1.55×10^{-4}	negligible	negligible

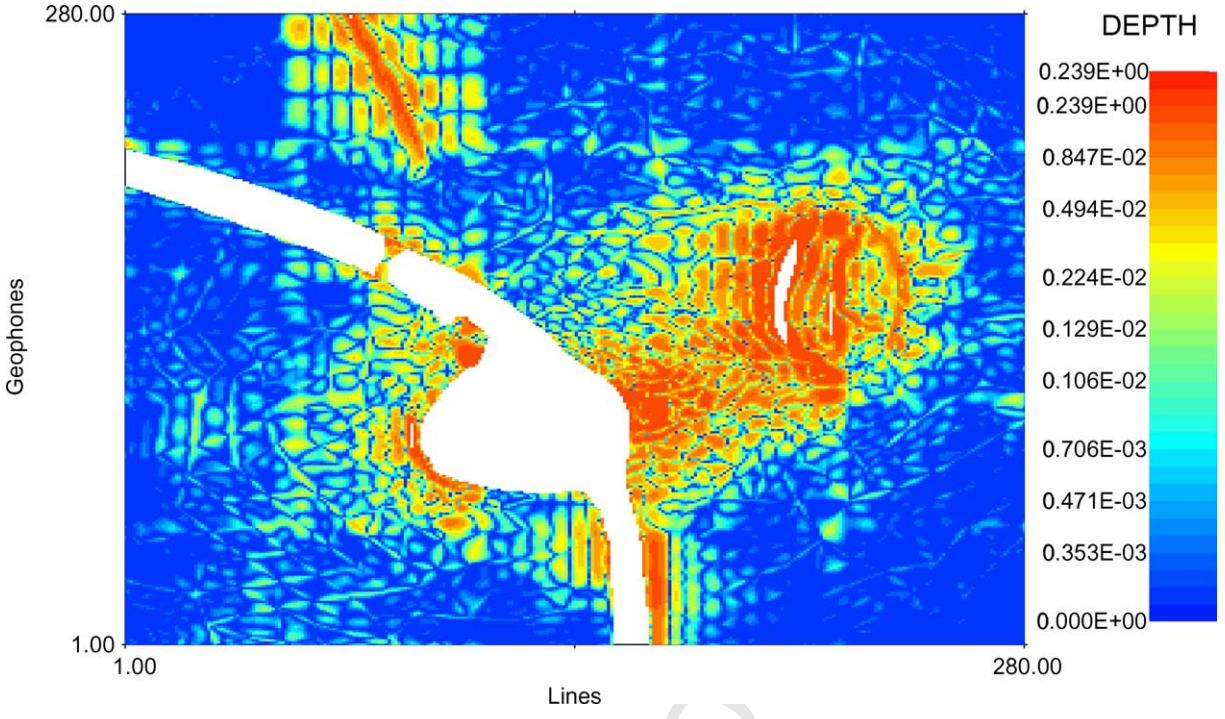

Fig. 5. Error in salt model for BTSVD method.

Table 6

CONLIM	RMS	Large residual	# Iterations	Time (sec.)	Ratio
10^2	8.5×10^{-3}	7.9%	31	25	0.1
10^3	7.1×10^{-3}	4.9%	166	66	0.24
$5{\times}10^3$	7.06×10^{-3}	4.9%	498	167	0.6
10^4	7.05×10^{-3}	4.9%	2762	863	3.1

(Ratio = LSQR time/BTSVD time.)

an RMS $= 8.18 \times 10^{-3}$, with 91% of the data having an error less than 0.5% is obtained in 2 sweeps, the initial one using local LSQR (32 iterations) and the second one using TSVD. The computing time was 280 seconds. In Fig. 5 we show a plot of the relative error.

The same test problem was approximated with an equal number of control vertices, using LSQR. Here the termination criteria used was again, as above, a threshold for the estimated condition number: CONLIM.

It can be seen from the table below and a frequency plot of the error, that with the choice of the regularization parameter CONLIM $= 10^3$ (66 seconds), one obtains an approximation of similar quality as with the BTSVD algorithm, and in a fraction of the computing time. (For an error plot see Fig. 6.)

On the other hand, that choice is not obvious and another value which gives a similar RMS is obtained in a computer time that is a factor larger than using the more robust BTSVD algorithm as can be seen from Table 6.

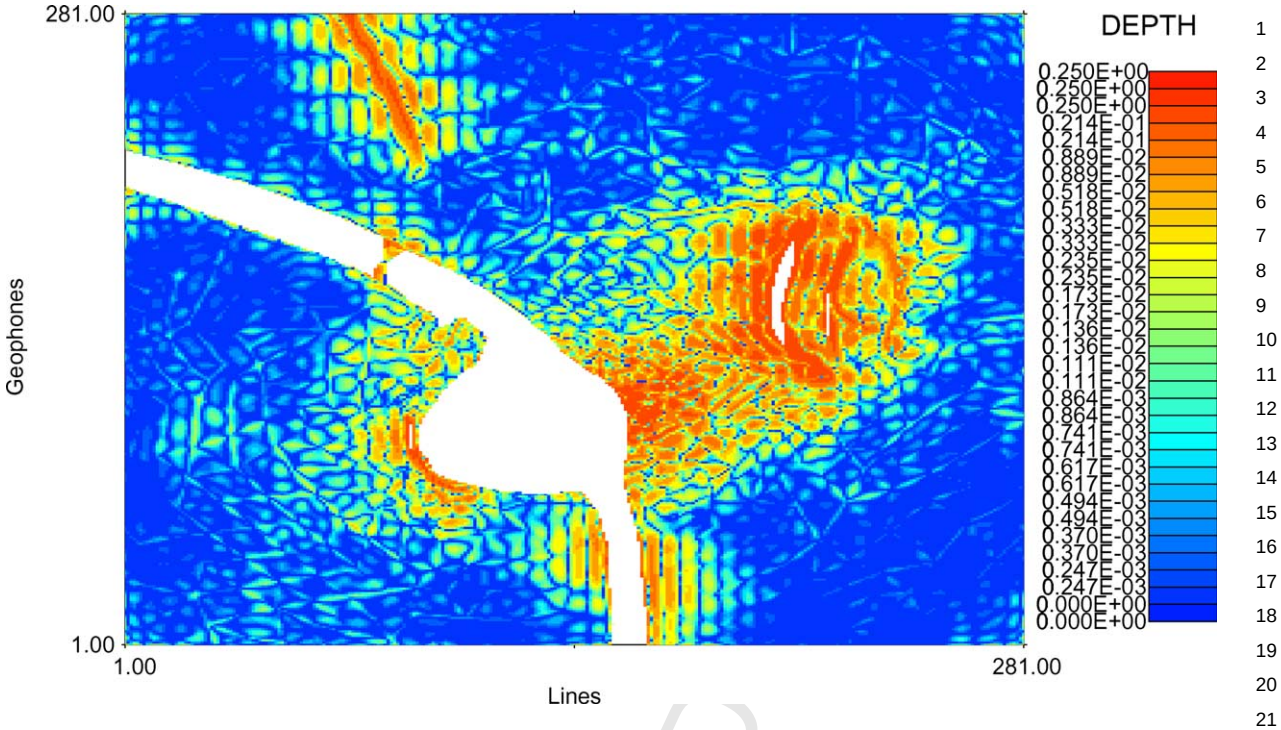

Fig. 6. Error in salt model for LSQR method.

5.4. *Comparisons and improvements*

A comparison of the methods should consider numerical stability and cost.

The flop count indicates a clear advantage for the LSQR algorithm versus the block TSVD, unless a very large number of iterations is needed, as maybe the case for a strongly ill-conditioned problem. The computing times of the examples tested corroborate this.

To shorten the computing time for the block TSVD, one could consider using the less robust LSQR in the trouble free regions (i.e., subregions with smooth data without large gaps) and switch to the more expensive TSVD algorithm for the more "singular" regions.

Automatic selection of regularization parameters is not implemented, neither for BTSVD, nor for the Lanczos iteration methods, but we cannot foresee mayor difficulties, at least for BTSVD. For both TSVD and projection methods, the L-curve is defined only by discrete data points, so prior to determining the maximum curvature by some optimization method, a twice differentiable approximating function must be determined. The preferred methods are variations of cubic splines approximations. (See [13, p. 190] for details.)

An interesting new method has recently been proposed by Gomez et al. [16]. It is a geometrically based algorithm that is computationally simpler and has less memory requirements than previously proposed methods. It works well as long as the L-curve is monotone, even if it does not have the characteristic L-shape. It may therefore be a promising alternative for LSQR since, to judge from the examples above, it seems to fall sometimes into this last category.

In addition to the well-known L-curve, when using iterative algorithms based on Lanczos bidiagonalization, envelopes to the L-curve, the so-called L-ribbon [5,6] could also be used to estimate the best parameter.

Another parameter selection technique that can be used and does not need additional statistical information about the noise is Generalized Cross-Validation (GCV). First proposed in 1979 by Golub et al. [10], it seeks to minimize the predictive mean-squares error. In [13,15] there are careful comparisons with both L-curve and GCV parameter selection techniques and different regularization methods applied to sets of ill-conditioned least squares problems. The general conclusion is that the optimal parameter selection technique is problem and regularization method dependent.

Our study of the comparative performance of algorithms for large systems would be incomplete without mentioning some additional state of the art algorithms. Two interesting hybrid algorithms based on projection by bidiagonalization combined with TSVD are the O'Leary and Simmons [15] and the Björck [2], and Björck et al. [4] methods. According to comments in [15], the results should be equivalent to the ones obtained with TSVD on the original problem. See also [9] for a method based on moment calculations and additional interesting comparisons.

Some important points in the O'Leary and Simmons algorithm are:

Preconditioning must be applied previously to the problem, eliminating therefore the potential difficulty from loss of orthogonality. After k iterative steps, the SVD of a $(k+1) \times k$ bidiagonal matrix must be computed. This requires $O(k)^3$ flops or $O(k)^2$ if updating the SVD from the corresponding previous matrix using the method in [11]. The threshold for the TSVD is determined from the $L(k)$ curve.

The Björck, Grimme and Van Dooren algorithm uses the Paige and Saunders bidiagonalization process. It computes the kth step SVD in $O(k)^2$ operations and uses GCV to determine the truncation point for the projected SVD. It adds reorthogonalization in an efficient way by applying an implicitly restarted Lanczos algorithm, thus improving the convergence of the Lanczos method.

Unfortunately, none of the above algorithms are available in the public domain, and could not be used for comparisons. A. Björck kindly made available to us a trial program in MATLAB with most of the afore mentioned details. Results using this program will be published in an extension of the present work.

6. Conclusions

BTSVD works well, even for large gaps, jump discontinuities and rough data, but computational cost, both in memory and computing time, is always more for large problems than using LSQR. This is compensated by a more robust regularization, and often by a better accuracy, since it may be difficult to find optimal parameters for LSQR in ill-conditioned problems.

From the tests run for this work, the sparse SVD approach using LASVD, seems to be too expensive to be considered any further.

The block TSVD implementation must be improved, in particular concerning patching smoothness and the treatment of subregions with large variations in data density. This should also make it more competitive in terms of computing time.

LSQR must be provided with an automatic regularization parameter selection, since failure to choose it correctly may result in very long computing times in the best case, or divergence in the worst.

Last but not least, BTSVD, is naturally pararellelizable and therefore could be more suitable for very large problems.

References

[1] M. Berry, Large scale sparse singular computations, Internat. J. Supercomp. Appl. 6 (1992) 13–49.
[2] A. Björck, A bidiagonalization algorithm for solving large and sparse ill-posed systems of linear equations, BIT 28 (1988) 659–670.
[3] A. Björck, Numerical Methods for Least Squares, SIAM, Philadelphia, PA, 1996.
[4] A. Björck, E. Grimme, P. Van Dooren, An implicit shift bidiagonalization algorithm for ill-posed systems, BIT 34 (1994).
[5] D. Calvetti, G. Golub, I. Reichel, Estimation of the L-curve via Lanczos bidiagonalization, BIT 39 (1999) 603–619.
[6] D. Calvetti, P. Hansen, I. Reichel, L-curve curvature bounds via Lanczos bidiagonalization, Report IMM-TR-2001-5, May 2001.
[7] T.F. Chan, An improved algorithm for computing the singular value decomposition, ACM Trans. Math. Software 8 (1982) 72–83.
[8] G. Golub, C. van Loan, Matrix Computations, 2nd edn., John Hopkins, 1989.
[9] G. Golub, Urs von Matt, Tikhonov regularization for large scale problems, Stanford SCCM Report 97-03, 1997.
[10] G. Golub, M. Heath, G. Wahba, Generalized cross-validation as a method for choosing a good ridge parameter, Technometrics 21 (1979) 215–223.
[11] M. Gu, S. Eisenstat, A stable and fast algorithm for updating the singular value decomposition, Tech. Report RR-939, Yale University, Department of Computer Science, 1993.
[12] M. Hanke, P.Ch. Hansen, Regularization methods for large-scale problems, Surveys Math. Indust. 3 (1993) 253–315.
[13] P.Ch. Hansen, Rank-Deficient and Discrete Ill-Posed Problems, SIAM, Philadelphia, PA, 1998.
[14] P.Ch. Hansen, The L-curve and its use in the numerical treatment of inverse problems, in: P. Johnston (Ed.), Comp. Inverse Problems in Electrocardiology, WIT Press, 2001, pp. 119–142.
[15] M.E. Kilmer, D.P. O'Leary, Choosing regularization parameters in iterative methods for ill-posed problems, Comp. Sci. Dept. Report CS-TR-3937, Inst. for Adv. Comp. Studies Report UMIACS-TR-98-48, Univ. of Maryland, 1998.
[16] J. Longina Castellanos, S. Gomez, V. Guerra, A novel method to find the corner of the L-curve to solve ill-conditioned linear systems, Appl. Numer. Math., submitted for publication.
[17] Ch.C. Paige, M.A. Saunders, LSQR: An Algorithm for sparse linear equations and sparse least squares, ACM Trans. Math. Software 8 (1) (1982) 43–71.
[18] V. Pereyra, Asynchronous distributed solution of large scale nonlinear inversion problems, Appl. Numer. Math. 30 (1999) 31–40.
[19] V. Pereyra, G. Scherer, Least squares scattered data fitting by truncated SVD, Appl. Numer. Math. (2001), to appear.
[20] B.W. Rust, Truncating the singular value decomposition for ill-posed problems Tech. Report NISTIR 6131, Math. and Comp. Sci. Div. Nat. Inst. of Stand. and Tech., 1998.
[21] C.R. Vogel, J.G. Wade, Iterative SVD-based methods for ill-posed problems, SIAM J. Sci. Statist. Comput. 15 (3) (1994) 736–754.