Least squares scattered data fitting by truncated SVD's

V. Pereyra and G. Scherer

Weidlinger Associates, 4410 El Camino Real, #110, Los Altos, CA, U.S.A.

Abstract

The l_2 fitting of scattered data by tensor products of B-splines in two and three dimensions is considered. A truncated Singular Value decomposition approach is used to automatically account for singularity and ill-conditioning of the Grammian matrix. The method does not require re-gridding and can be applied to data with large holes and irregular footprint, provided that the domain of evaluation is limited to the region where there is enough data. A direct and two iterative methods are discussed and numerical results are offered for several test data sets.

Key words: Scattered data fitting, truncated SVD

1 Introduction

We consider the least squares fitting of scattered 2D $(x, y; z)$, and 3D $(x, y, z; v)$ data by a tensor product of B-splines. The data lies in an irregularly shaped domain bounded by curves (in 2D) or surfaces (in 3D). This domain can be multiply connected. We assume that this is also the domain of evaluation, i.e., the B-spline approximation will never be evaluated outside of that domain. We further limit the data sets to one-to-one height maps, so that our representations will be explicit.

Most of the time we will assume that the data values satisfy:

$$z_i = f(x_i, y_i) + n_i,$$

where $f(x, y)$ is a twice differentiable function with moderate size derivatives, and $\{n_i\}$ is a random noise vector, where the n_i are uncorrelated random variables with mean zero and the same variance σ^2, which we assume small

relative to the size of f. In addition, we will also show that some types of jump discontinuities can be automatically handled.

This work has been motivated by the requirements of modeling geological interfaces from seismic related data, but it has much wider applicability.

As a first step of our algorithm, we embed the domain into a rectangular (parallelepipedal) box and define an uniform grid of B-spline basis functions on it. Clearly, if there are holes or large areas of missing data within this rectangle, where the term "large" refers to the scale of the basis function spacing, the least squares problem will be ill-conditioned or even singular. This will occur, for instance, in the case that a control vertix is isolated from the data. As we recall, in the case of cubic B-spline basis functions, only four control vertices in each coordinate direction are active at any given evaluation point.

Reciprocally, if a control vertix is more than two basis function intervals away from every data point, then it will not enter into any evaluation of the leasts squares matrix, and thus it will produce a column of zeros, leading to a zero singular value and matrix singularity. Additionally, control vertices which are close to only a few data points can be poorly defined by the data and therefore they can lead to small singular values and ill-conditioning.

Two possible remedies to this situation are:

- To limit the minimum mesh size of the B-spline mesh, so that there are no "large" gaps;
- To regrid the data to an uniform mesh, filling any gaps by interpolation or extrapolation.

These two approaches have limitations: the first one may produce smoother approximations than what the data requires, with consequent loss of accuracy in the representation; the second one is fraught with danger of artifacts due to extrapolation to unknown territory. An advantage of the second approach is that, after regridding to an uniform mesh, one can use a more economical algorithm for tensor product fitting of tensor product data [11].

A third alternative, that can be used in the case that the domain is simply connected and the boundary is an irregular quadrilateral bounded by four curves (or six surfaces in 3D), is to define a Coons patch and use its parametric grid for the regridding process. Then, one can use tensor product B-splines on parametric tensor product data.

Finally, [10,13] have also used an elaborated interactive graphic segmentation approach to partition a multiply connected domain, or a domain with more than four bounding curves, into quadrilaterals, in order to apply the above

procedure. Since this segmentation is generally artificial, one needs to take care of joining the segments smoothly, an additional complication. None of these approaches are entirely satisfactory.

A very interesting piece of work addressing a much more general problem and solving it in a robust, automatic way, can be found in [8]. The resulting surface is only piecewise C^1, so is not good enough for our purposes, but we plan to see if we can use in the future some of the ideas described there, specially with regards to automatic detection of boundaries and discontinuities. That would relieve the user from having to indicate the boundaries of the domain of evaluation of the approximating function.

The method we present in this paper involves no re-gridding, and thus leads to the more expensive alternative of fitting non-tensor product data by tensor products of B-splines. However, no extrapolation or interpolation of the data is required.

As observed above, the resulting linear least squares problem can be ill-conditioned or singular. Another problem for this type of data fitting and for parametric modeling in general, is how to decide the dimension of the fitting space (i.e., the number of basis functions).

In order to solve these two problems we propose an algorithm that combines a multigrid or multi-resolution approach with a Truncated Singular Value Decomposition (TSVD) technique. If the problem is relatively small we can use direct solve techniques, and as the problem becomes large we will switch to iterative approaches. Two different iterative methods will be considered:

- A block Gauss-Seidel TSVD method similar to a linear version of the one advocated in [12,14,15], for large scale nonlinear least squares problems arising in inverse geophysical problems.
- An standard iterative SVD approach for large, sparse problems based on Lanczos method [2].

We present also a number of numerical examples in order to demonstrate the practical value of the basic method, to illustrate its behavior, and to make comparisons and recomendations about the different methods. An interesting source of background material and other approaches to this and similar problems can be found in [4].

2 Statement of the problem

We describe in detail the 2D case, since it is easier to explain and visualize. We show later that extensions to 3D are straightforward.

Let $(x_i, y_i; z_i), i = 1, ..., m$ be a set of scattered data, i.e., the independent variables (x_i, y_i) are not necessarily on a regular grid. As indicated above, we assume that the data is associated with a smooth function plus random noise and that the approximating B-spline will be evaluated only in a domain D defined by some boundary curves.

We want to best approximate the data by least squares fitting of a tensor product of B-splines. Again, only for simplicity and to fix ideas, we consider cubic splines. All these restrictions can be removed and we will indicate later the extensions that are not obvious.

Let R be a rectangle containing the domain D, and let $(sx_j, sy_k)\ j = 1, ..., J;\ k = 1, ..., K$, be a uniform mesh defining the location of the basis functions. Then, the B-spline approximant can be written as:

$$B(x, y) = \sum_{j,k} C_{jk} B_j(x) B_k(y),$$

where the C_{jk} are the control vertices.

The least squares problem can then be stated as:

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

If we order the unknowns C_{jk} by columns to form a one-dimensional array, then we can write in matrix notation:

$$\min_{\mathbf{C}} \parallel \mathbf{BC} - \mathbf{Z} \parallel_2^2,$$

where $\mathbf{C}\ and\ \mathbf{Z}$ are the vectors of the control vertices and the data respectively, while $\mathbf{B} = B_j(x_i) B_k(y_i)$ is the Grammian matrix associated with the B-spline basis functions and the data. This is also the least squares matrix, of dimensions $m \times n$, where $n = J \times K$ and, of course, we assume that $n \ll m$.

3 Basic solution of the linear least squares problem (LLSP)

There are many different methods to solve the LLSP. For moderate dimensions, the approaches discussed in [3,6] are classical and advisable because of their numerical stability and efficiency. They are based on orthogonal transformations, but unfortunately they are limited to the full rank and not too ill-conditioned case only. Some modifications can be employed for the exact rank deficient case, but for ill-conditioned problems it is better to use Singular Value Decomposition (SVD) techniques instead.

The SVD of the least squares matrix $\mathbf{B}$ can be written as:

$$\mathbf{B} = \mathbf{U}\Lambda\mathbf{V}^{\mathbf{T}},$$

where the matrix $\mathbf{\Lambda}$ contains the singular values and $\mathbf{U}, \mathbf{V}$ are orthogonal matrices. With this decomposition, an approximate regularized solution of the least squares problem can be written as:

$$\mathbf{C} = \mathbf{B_k}^{\dagger}\mathbf{Z},$$

where $\mathbf{B_k}^{\dagger}$ is the truncated pseudoinverse of $\mathbf{B}$.

The truncated pseudoinverse of $\mathbf{B}$ is defined as:

$$\mathbf{V}\mathbf{\Lambda_k}^{\dagger}\mathbf{U}^{\mathbf{T}},$$

where

$$\Lambda_k{}^{\dagger} = diag(\lambda_1^{-1}, ..., \lambda_k^{-1}, 0, ...0)$$

with $k \leq min(n, r)$. Here r is the rank of $\mathbf{B}$ (i.e., if, as usual, we assume the singular values ordered from largest to smallest, then λ_r is the last non-zero singular value).

The pseudoinverse takes care naturally of exact rank deficiency leading to the unique solution of minimum norm. By choosing appropriately the cut-off point, the truncated version can take care also of ill-conditioning. See the seminal paper [9] for an early discussion of this method and [7] for many additional modern insights.

In order to obtain a practical algorithm from these ideas, we have to respond to the following questions:

- How do we choose the number of basis functions (i.e., J *and* K)?

- How do we choose the cut-off level for the singular values?
- How do we monitor that the resulting representation is a "good one" (i.e., preserves the character of the discrete data, without introducing unsightly artifacts or excessive smoothing)?
- How do we solve large scale problems?

In the next sections we will try to answer these questions.

4 Multigrid and regularization

We propose a multigrid approach to answer the first question of the previous section. Starting from a coarse mesh for the basis functions and possibly a decimated data set, we solve the resulting least squares problem and verify if the residual is less than a prescribed level, ϵ.

One good aspect of the present problem is that we can verify the quality of the fit by calculating a norm of the residual at the data points. In addition, we can also estimate other discrete properties of the data by evaluating local difference quotients and comparing them with derivatives of the fitting function. In this way we can try dynamically not only to satisfy a certain goodness of fit, but also to preserve the character of the discrete data. Again, [8] does a more complete job of identifying edges, creases, and other discontinuities automatically, at the cost of a fairly complicated and expensive algorithm.

By the character of the data we mean local discrete properties that approximate gradients, and curvatures. We will not go as far as to say that this is a full blown shape preserving approximation technique, but we will try, within its limitations, to avoid obvious pitfalls.

By considering a Delaunay triangulation of the data points (x_i, y_i) we can also calculate for every point a finite difference approximation δ_x, δ_y to the first order partial derivatives, z_x, z_y.

These can then be compared with the partial derivatives of the approximant function:

$$\mathbf{B_x} = \sum_{j,k} C_{jk} B_j'(x) B_k(y),$$

$$\mathbf{B_y} = \sum_{j,k} C_{jk} B_j(x) B_k'(y).$$

The norm of the difference between these quantities can either be introduced in the functional to be minimized as a penalty term, or it can be used in con-

junction with the function residual to decide if the current mesh is adequate. If not, a refinement of the basis functions is introduced and the procedure continues until either the goal is achieved, the problem exceeds the machine capabilities (or our budget), or there is not enough data redundancy.

This is a simple multigrid or multi-resolution approach designed to help choose dinamically the dimension of an appropriate parameter space. More sophisticated local refinements, such as those of Bartels and his collaborators [5] are very appealing and could be useful to improve efficiency in the case that the data behavior varies considerably in the domain of interest, but they are beyond the scope of the current effort.

For a given set of basis functions we choose the cut-off level by a method similar to the L curve method of regularization [7]. We essentially calculate the norm of the control vertices and plot it, in a double logarithmic scale, against the norm of the residuals, for each cutting level k, and choose that one for which the elbow of the L curve is reached (see Figure 1).

Observe that the function $(k, \| \mathbf{res_k} \|_2^2)$ can be easily evaluated for all k by considering the following relationships:

$$\mathbf{C} = \mathbf{V}\mathbf{\Lambda}_\mathbf{k}^\dagger\mathbf{U}^\mathbf{T}\mathbf{Z}, \tag{1}$$

$$\mathbf{res_k} = \mathbf{BC} - \mathbf{Z}, \tag{2}$$

$$\| \mathbf{res_k} \|_2^2 = \| \mathbf{U}\mathbf{\Lambda}\mathbf{\Lambda}_\mathbf{k}^\dagger\mathbf{U}^\mathbf{T}\mathbf{Z} - \mathbf{Z} \|_2^2, \tag{3}$$

$$\| \mathbf{res_k} \|_2^2 = \| (\mathbf{\Lambda}\mathbf{\Lambda}_\mathbf{k}^\dagger - \mathbf{I})\mathbf{W} \|_2^2, \tag{4}$$

$$\tag{5}$$

where $\mathbf{W} = \mathbf{U}^\mathbf{T}\mathbf{Z}$, and where we have used the fact that $\mathbf{U}$ is an orthogonal matrix. Therefore,

$$\| \mathbf{res_k} \|_2^2 = \sum_{l=k+1}^{n} w_l^2.$$

Instead of varying k as indicated above, we can alternatively choose a prescribed positive value τ, and select k as that for which res_k falls below τ. We can then combine both procedures by varying τ monotonically.

It is somewhat surprising that this simple TSVD method, with an appropriate cut-off value τ, produces good approximants for scattered data on an irregular domain, even in the presence of random noise and even when the rectangle where the basis functions reside is much larger than the domain of interest D. Even more surprisingly, large holes and severe discontinuities between disconnected regions are also well tolerated. This is achieved without re-gridding and without extending the data in any way, as we show in the next examples.

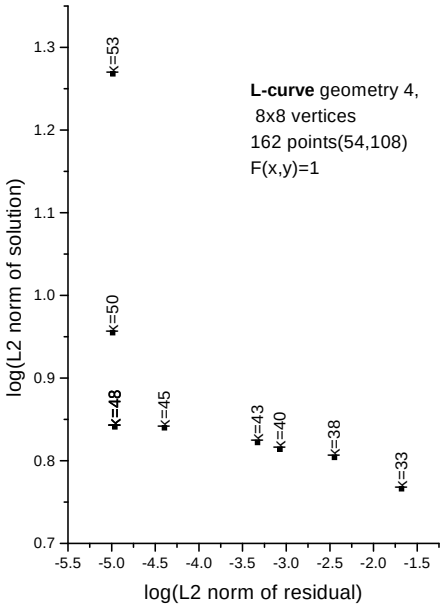

Fig. 1. Example of L-curve for domain 4 and a constant function

5 Examples of scattered data fitting with TSVD

We consider some simple examples first, consisting of data from five different functions sampled in four subsets of the unit square. The functions are:

$$f_1(x,y) = (1+x+x^2+x^3)(1+y+y^2+y^3), \tag{6}$$

$$f_2(x,y) = 25(x-0.25)(x-0.3)(x-0.7)(x-0.75) + 25(y-0.25)(y-0.3)(y-0.5)(y-0.75), \tag{7}$$

$$f_3(x,y) = 2/(1+25(x-0.5)^2) + 2/(1+25(y-0.5)^2), \tag{8}$$

$$f_4(x,y) = x^{(1/3)}y^{(1/3)}, \tag{9}$$

$$f_5(x,y) = x\ sin(9x^2-1) + y\ sin(9y^2-1) \tag{10}$$

The different geometries are depicted in Figure 2, where the crosses correspond to the position of the data points (generated at random), while the white areas are empty. The B-spline basis is defined in the whole unit square in all cases.

We have implemented the cutting levels by introducing a threshold τ for the normalized singular values (λ_i/λ_1). In Tables 1 and 2 we present information on the problems tested and the results obtained. In this first test we chose to make uniform decisions over all the problems and geometry to see the resulting different behaviors and identify the hardest problems. For this purpose we set

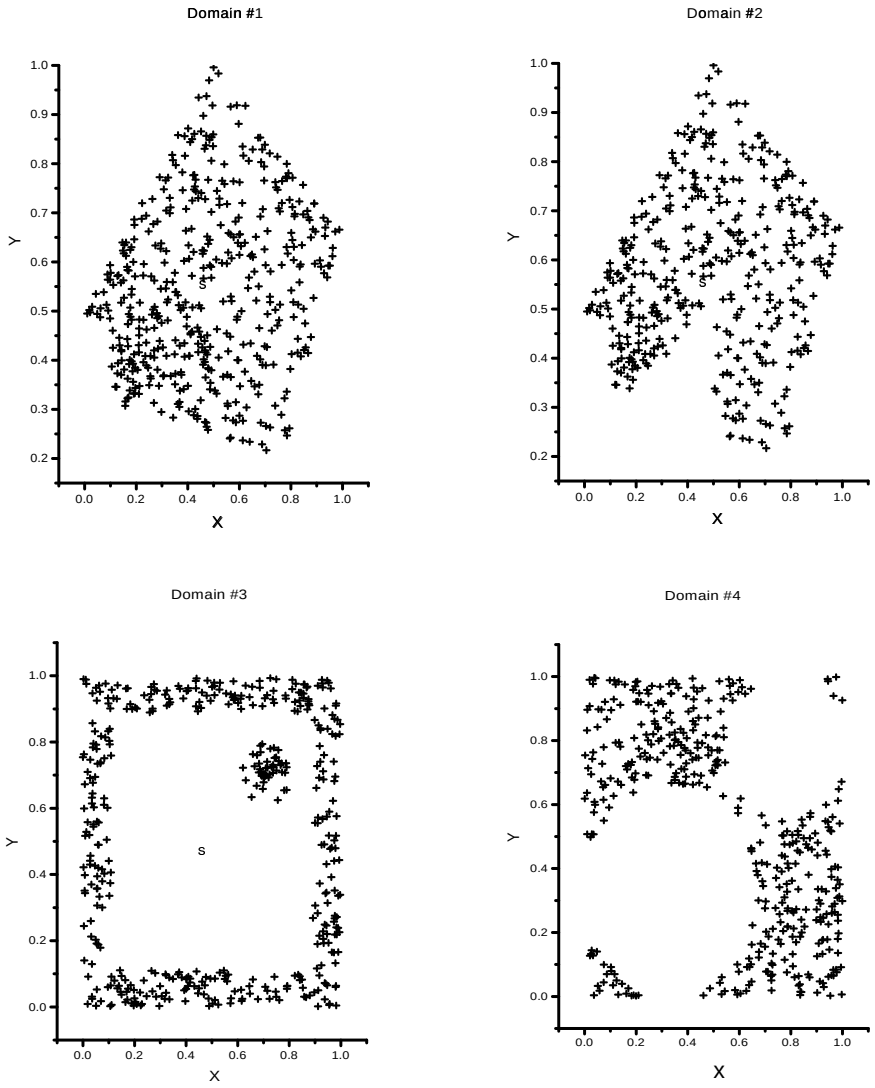

Fig. 2. The different domains we use for the examples

$\tau = 0.0001$, 12×12 nodes for the B-spline approximant, and no random error. Domain 0 is the whole unit square, and it is used as reference.

Next we consider Function 3 in Domain 3 for different numbers of control vertices. The cutting threshold was $\tau = 1.0e-4$. We recall that there are 481 data points in this example.

Finally we redo this calculation but including a 0.01 level of random noise.

We see that the improvement with control vertices mesh refinement is limited by the noise level, as expected, but no other side effect seems apparent.

In Figure 3 we show color maps of the absolute error between the original

Table 1
Basic information on the test problems

Dom/Fcn-1	SV cut	# data	fmin	fmax
0	0	400	1.13	12.81
1	33	472	-0.005	1.88
2	36	481	0.56	3.96
3	20	450	0.07	0.95
4	16	412	-1.57	1.53

Table 2
Residual mean square

Dom/Fcn-1	0	1	2	3	4
0	1.2e-4	3.8e-3	1.6e-3	5.0e-3	1.7e-2
1	5.6e-5	1.8e-3	1.4e-3	1.2e-3	8.6e-3
2	6.5e-5	2.0e-3	1.2e-3	1.3e-3	9.6 e-3
3	1.4e-4	4.4e-3	1.4e-3	7.3e-3	1.8e-2
4	1.6e-4	5.0e-3	6.2e-4	6.2e-3	2.5e-2

Table 3
Domain 3, function 3

# CV	# Cut SV	$\| res \|_\infty$	rms
5x5	0	0.09	0.028
10x10	10	0.016	0.0045
20x20	123	0.25e-3	0.59e-6

Table 4
Domain 3, function 3 with 0.01 noise level

# CV	# Cut SV	$\| res \|_\infty$	rms
5x5	0	0.097	0.029
10x10	10	0.041	0.0064
20x20	123	0.019	0.32e-4

sampled function and the B-spline approximation for some of the functions on domain 3. The map corresponds to a 100×100 mesh. The two columns of pictures are showing exactly the same data, what changes is the color map. In the first column we have a linear map between the range of the error and the available colors, while in the second column we show a histogram equalization scale, which uses more colors for ranges where there are more pixels. That is the reason for the apparent high frequency variability in those pictures, and

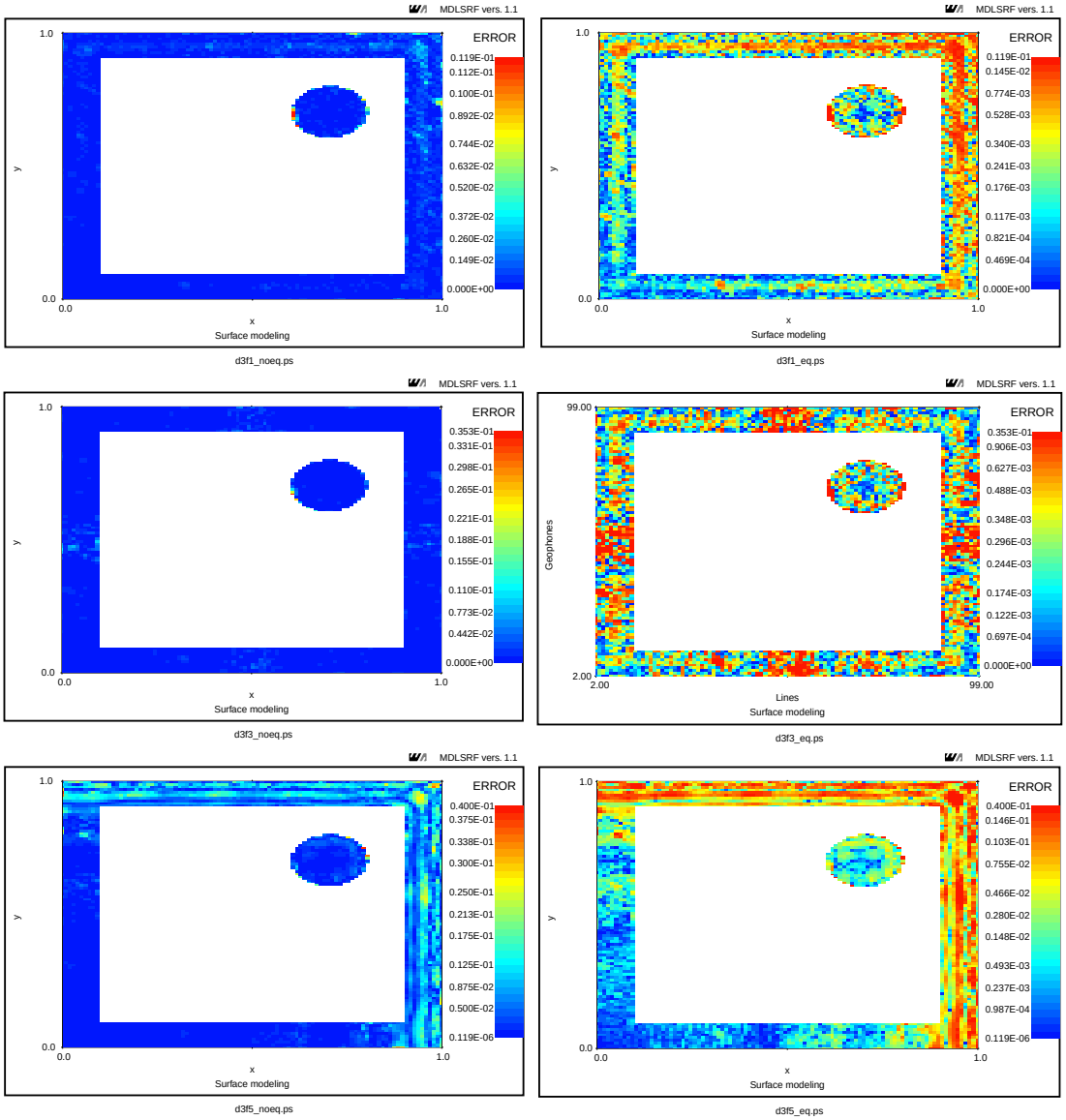

Fig. 3. Absolute error for some of the functions on domain 3

the message is: pay attention to the color scale! In Figure 4 we show a similar map for function f_3 for some of the other domains.

5.1 Results for discontinuous data

We now show how some types of jump discontinuities can be handled if they have a missing data gap between them (as it would be the case for normal faults with a dipping fault plane).

In Figure 5 we show both the error distribution and a map of the error for two examples. For the first example we have the same function but now defined

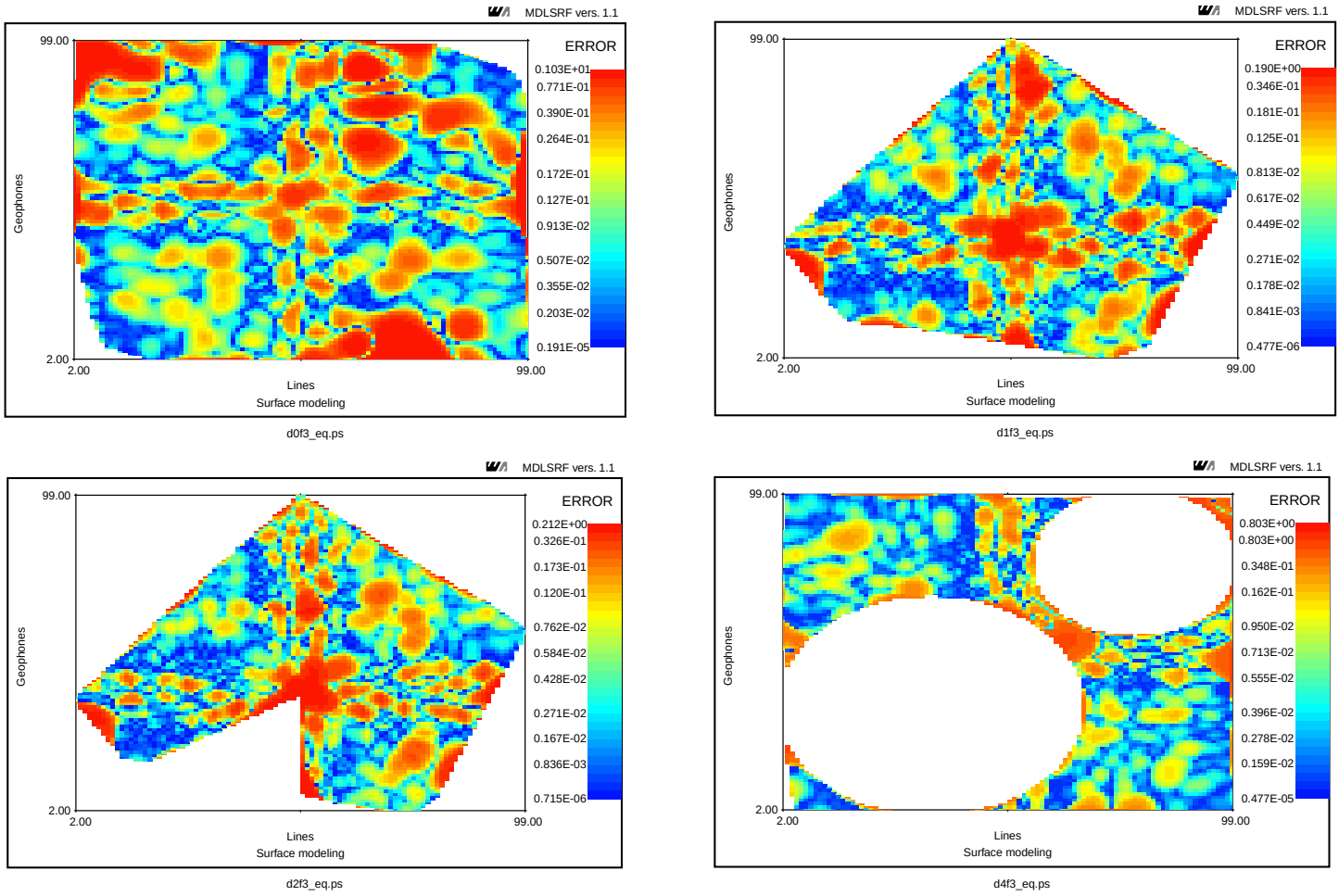

Fig. 4. Absolute error for function 3 in different domains

in two triangles separated by a gap. The maximum absolute residual in this case was 0.052, for an 8×8 B-spline.

In the second one, the function is:

$$z(x,y) = 1, (x,y) \in [0.5,1] \times [0,0.5], \tag{11}$$
$$= 2, (x,y) \in [0,1/3] \times [2/3,1]. \tag{12}$$

The data points have been selected at random, with 108 of them in each subregion. The number of control vertices for the B-spline approximation in the unit square was 8×8. Singular values were cut below 1.0e-6. The maximum absolute residual was 0.0077.

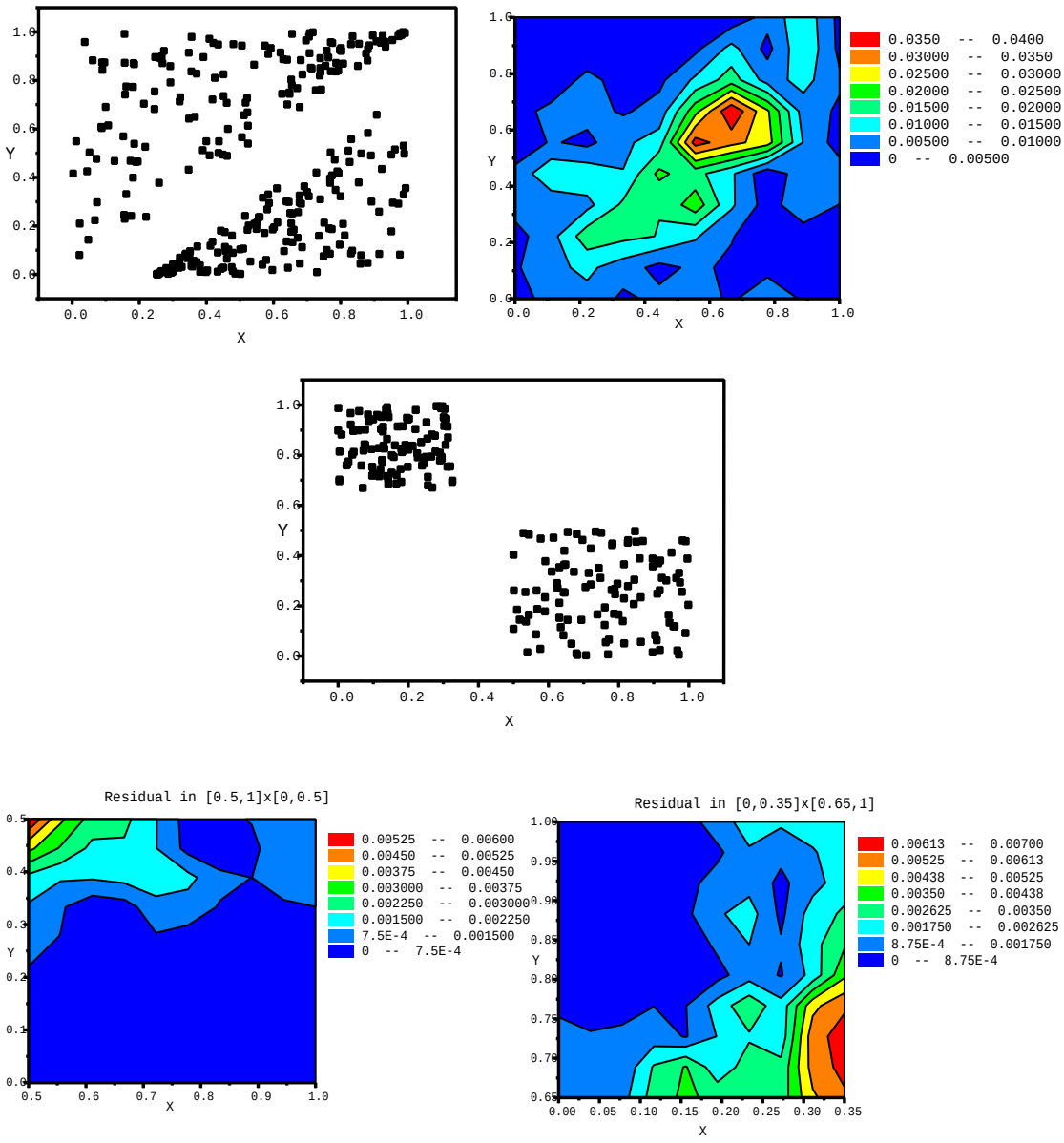

Fig. 5. Data distribution and absolute error for discontinuous functions

6 Block Gauss-Seidel method for large dimensional problems

Once the size of the problem increases beyond the capabilities of direct methods we need to resort to an iterative approach. In this section we discuss the application of the ideas of [12,14,15] for the solution of large scale nonlinear least squares to the current problem.

The main two principles for the success of this block approach are that:

- The data of the problem can be subdivided in groups of points that are geometrically close;

- The model is also local in nature.

In the present problem we can naturally subdivide the domain, grouping the data for each sub-domain, and the B-spline representation is local by nature. Thus, by solving the least squares problem for each sub-domain we have an ideal setting for divide and conquer techniques. In addition we also assume that the number of sub-regions is such that the sub-problems are amenable to solution by the previous direct TSVD method.

The only overlap between the regions will occur at the boundaries and is due to the so called "phantom vertices". These are control vertices that are outside of the rectangular domain of definition and that are required to evaluate the B-spline function near the boundaries.

However, with the block approach there is a difference between external and internal boundaries: for internal boundaries of adjacent subdomains that are in contact we need to communicate already calculated information. Thus we propose the following algorithm:

- Subdivide the enclosing rectangle R into sub-domains $R_s,\ s = 1, ..., S$.
- Let the data belonging to each sub-domain be denoted by $\{x_i^s, y_i^s; z_i^s\},\ i = 1, ..., m^s$.
- For each subdomain, trim the Grammian matrix of all the columns with norm below a small threshold. What remains are the active basis functions. We could also use an SVD based sensitivity analysis as explained in [14] and eliminate some additional, poorly defined control vertices. The truncated SVD algorithm has a similar effect, and thus this sensitivity analysis is not required here.
- Solve the resulting sub-problems in sequence, or asynchronously in parallel, updating the vector of control vertices as new values are produced. For internal boundaries, if the phantom vertices are already initialized from a calculation for an adjacent subdomain, use those values.
- A pass through all the sub-domains is called a sweep. Perform several sweeps, if necessary, until the desired solution is achieved.

A version of this method, appropriate for large scale ill-conditioned, non-linear least squares problems, has been extensively tested and a convergence theory for the parallel asynchronous algorithm has been published (see references above). Sometimes a Jacobi approach, in which the parameters are only updated at the end of a sweep gives better numerical results. On the other hand, that requires a synchronization step and thus affects negatively parallel performance.

6.1 Some numerical results for the block decomposition iterative method

We subdivide the unit square in four equal parts and perform a couple of sweeps of the Block Iterative algorithm for various combinations of the functions and domains of Section 5. In Table 5 we show the results obtained. An 11×11 mesh was used throughout for the B-spline approximation on the unit square. More realistic, large scale tests will be reported in a future paper.

Table 5
Results for the block method

Dom.Fcn.	rms
D1f1	1.2e-4
D2f2	5.2e-3
D3f3	1.6e-3

Finally, in Table 6 we show a comparison between the direct and the block methods.

Table 6
Comparison of the block method with the direct one

Mesh	Dom.Fcn.	Direct rms	Block rms	# sweeps
5×5	D1f1	1.2e-3	2.9e-3	3
5×5	D3f3	2.8e-2	2.9e-2	1
11×11	D1f1	7.3e-5	1.2e-4	2
11×11	D3f1	1.4e-3	1.6e-3	1

7 Lanczos based iterative SVD calculation for sparse problems

Although we have not emphasized it above, another benefit arising from the local support property of the B-spline basis functions, is that the Grammian matrix is sparse. In fact, regardless of the number of basis functions (number of columns), there are only a maximum of 16 non-zero coefficients in 2D (and 64 in 3D) for any data point. For large problems, where these iterative techniques become necessary, we may have hundreds of columns (or thousands in 3D), and thus the density of non-zero matrix elements is fairly small.

Thus, iterative methods for sparse matrices that calculate first the largest singular values and then others with decreasing accuracy are quite appealing for this type of approach. Berry [1,2] has done extensive work in this area, including public domain implementations of several basic algorithms and also parallel versions of them. We have selected LASVD-1, a modified version of

LANSO, a package originally written by B. Parlett and his colleagues at U.C. Berkeley. According to Berry's extensive comparisons, LASVD-1 is the fastest available methods for the low and moderate accuracies required by practical engineering problems. If memory is a concern, then other methods, such as SISVD are advisable, while for systems which can exploit a high degree of coarse grain parallelism the prefered program would be TRSVD.

LASVD-1 is designed to calculate iteratively, via a single vector Lanczos method, some of the eigentrios of an sparse rectangular matrix A, by considering the symmetric 2-cyclic matrix:

$$B = \begin{pmatrix} O & A \\ A^T & 0 \end{pmatrix}.$$

We plan to use this code to approximate some of the more significant SV's of B, use them as our truncated SVD approximation, and compare the results and performance with the block approach.

8 Approximation of volumetric properties given by a scattered data set

The 3D version (actually 4D, if we include the function values) of the above algorithms is quite straightforward. The B-spline approximation will now live in a 3D box and have the form:

$$B(x, y, z) = \sum_{jkl} C_{jkl} B_j(x) B_k(y) B_l(z),$$

while the least squares problem will look like:

$$min_{C_{jkl}} \sum_i (\sum_{jkl} C_{jkl} B_j(x_i) B_k(y_i) B_l(z_i) - v_i)^2.$$

From here on, the rest is the same, except that now the problems will get large very fast, so that the iterative techniques will be essential. In a future paper we will show results for the iterative methods and 4D data sets.

References

[1] M.W. Berry, Large-scale sparse singular value computations, Manuscript, 1990.

[2] M.W. Berry, Large-Scale Sparse Singular Value Decompositions, Internat. J. Supercomputing Appl. 6 (1992) 13-49.

[3] A. Bjorck, Numerical Methods for Least Squares Problems, SIAM Pub., Philadelphia, 1996.

[4] P. Dierckx, Curve and Surface Fitting with Splines, Clarendon Press, 1993.

[5] D. Forsey, and R. Bartels, Hierarchical B-spline Refinement, Comp. Graphics 22 (1988) 205-212.

[6] G. Golub, and C. F. Van Loan, Matrix Computations, John Hopkins University Press, 1989.

[7] P.C. Hansen, Rank-Defficient and Discrete Ill-Posed Problems, SIAM Pub., Philadelphia, 1998.

[8] H. Hoppe, Surface reconstruction from unorganized points, Ph. D. Thesis, Univ. of Washington, 1994.

[9] L. Jupp, and K. Vozoff, Stable Iterative Methods for the Inversion of Geophysical Data, Geophys. J. R. astr. Soc. 42 (1975) 957-976.

[10] G. Turk, and M. Levoy, Zippered polygon meshes from range images, Computer Graphics. SIGGRAPH'94 Proceedings, 1994.

[11] V. Pereyra, and G. Scherer, Efficient Computer Manipulation of Tensor Products with Applications to Multidimensional Approximation, Math. Comp. 27 (1973) 595-605.

[12] V. Pereyra, Parallel Block Inversion of Geophysical Data, in Proc. Math. Methods in Geophysical Imaging, SPIE Internat. Symp., pp. 192-198, 1993.

[13] V. Pereyra, M. Koshy, and L. Carcione, Salt domes with faulted flanks: INTEGRA model construction from irregularly digitized discontinuos data, Weidlinger Associates Geophysical Inversion Project, VI, pp. 64-90, 1993.

[14] V. Pereyra, Modeling, Ray Tracing, and Block Nonlinear Travel Time Inversion in 3D, Pure Appl. Geophys. 148 (1996) 343-386.

[15] V. Pereyra, Asynchronous Distributed Solution of Large Scale Nonlinear Inversion Problems, Applied Numerical Mathematics 30 (1999) 31-40.