



THE AUSTRALIAN NATIONAL UNIVERSITY

TR-CS-01-01

**A Scalable Parallel FEM Surface
Fitting Algorithm for Data Mining**

**Peter Christen, Markus Hegland, Stephen
Roberts and Irfan Altas**

January 2001

Joint Computer Science Technical Report Series

Department of Computer Science
Faculty of Engineering and Information Technology

Computer Sciences Laboratory
Research School of Information Sciences and Engineering

This technical report series is published jointly by the Department of Computer Science, Faculty of Engineering and Information Technology, and the Computer Sciences Laboratory, Research School of Information Sciences and Engineering, The Australian National University.

Please direct correspondence regarding this series to:

Technical Reports
Department of Computer Science
Faculty of Engineering and Information Technology
The Australian National University
Canberra ACT 0200
Australia

or send email to:

`Technical.Reports@cs.anu.edu.au`

A list of technical reports, including some abstracts and copies of some full reports may be found at:

<http://cs.anu.edu.au/techreports/>

Recent reports in this series:

- TR-CS-00-02 Peter Strazdins. *A survey of simulation tools for cap project phase iii.* October 2000.
- TR-CS-00-01 Jens Gustedt, Ole A. Maehle, and Jan Arne Telle. *Java programs do not have bounded treewidth.* February 2000.
- TR-CS-99-02 Samuel Taylor. *A distributed visualisation tool for digital terrain models.* July 1999.
- TR-CS-99-01 Peter E. Strazdins. *A dense complex symmetric indefinite solver for the Fujitsu AP3000.* May 1999.
- TR-CS-98-14 Michael Stewart. *A completely rank revealing quotient uv decomposition.* December 1998.
- TR-CS-98-13 Michael Stewart. *Finding near rank deficiency in matrix products.* December 1998.

A SCALABLE PARALLEL FEM SURFACE FITTING ALGORITHM FOR DATA MINING

PETER CHRISTEN[†], MARKUS HEGLAND[†], STEPHEN ROBERTS[†], AND IRFAN ALTAS[‡]

Abstract. The development of automatic techniques to process and detect patterns in very large data sets is a major task in data mining. An essential subtask is the interpolation of surfaces, which can be done with multivariate regression. Thin plate splines provide a very good method to determine an approximating surface. Unfortunately, obtaining standard thin plate splines requires the solution of a dense linear system of order n , where n is the number of observations. Thus, standard thin plate splines are not practical, as the number of observations for data mining applications is often in the millions.

We have developed a finite element approximation of a thin plate spline that can handle data sizes with millions of records. Each observation record has to be read from an external file once only and there is no need to store the data in memory. The resolution of the finite element method can be chosen independently from the number of data records. An overlapping domain partitioning is applied to achieve parallelism. Our algorithm is scalable both in the number of data points as well as with the number of processors. We present first results on a Sun shared-memory multiprocessor.

Key words. Thin Plate Splines, Finite Element method, Parallel Computing, Linear System

1. Introduction. With *Data Mining* one designates techniques to automatically process and detect patterns in very large data sets [2]. Data mining can be used for spotting trends in data that may not be easily detectable by traditional database query tools that rely on simple queries (e.g. SQL statements). Data mining tools reveal hidden relationships and patterns as well as uncover correlations [4].

The main reason for using data mining tools is the explosive growth in the amount of data being collected in the last decade. The computerization of business transactions and use of bar codes in commercial outlets have provided businesses with enormous amounts of data. Nowadays, Terabyte data bases are common, with Gigabytes added every day. Revealing patterns and relationships in such data sets can improve the goals, missions and objectives of many organizations. For example, sales records can reveal highly profitable retail sales patterns.

A typical data mining application is the analysis of insurance data where there may be more than several hundred thousand policies with tens of variables each. Insurance companies analyze such data sets to understand claim behavior. Some insureds lodge more claims than others do. What are the typical characteristics of insureds who lodge many claims? Would it be possible to predict whether a particular insured will lodge a claim and accordingly offer an appropriate insurance premium? A major task in data mining is to find out answers to these types of questions.

An important technique applied in data mining is multivariate regression which is used to determine functional relationships in high dimensional data sets. A major difficulty which one faces when applying non-parametric methods is that the complexity grows exponentially with the dimension of the data set. This has been called *curse of dimensionality*. Additive and interaction splines can be used to overcome this curse [5]. In the case where interaction terms in the splines are limited to order two interactions, the model consists of a sum of functions of one or two variables and the fitting problem thus is reduced to fitting a sum of functions of two variables. One could call this problem surface fitting of a set of coupled surfaces. As such surface

[†]Computer Science Laboratory, RSISE, Australian National University, Canberra, Australia

[‡]School of Information Studies, Charles Sturt University, Wagga Wagga, Australia

fitting is an important technique for the data mining of large data sets.

We have developed a generic surface fitting algorithm that can handle data sizes with millions of observations. The algorithm combines the favorable properties of finite element surface fitting with the ones of thin plate splines. An introduction to this *thin plate spline finite element method* (TPSFEM) is given in Section 2. We discuss the numerical solution of the linear equations arising from the TPSFEM in Section 3.

In order to interactively analyze large data sets, a considerable amount of computational power is needed in most cases. High-performance parallel computing is crucial for ensuring system scalability and interactivity as data sets grow considerably in size and complexity. The parallel implementation of the algorithm is presented in Section 4 and first performance results are given in Section 5. Conclusions and future work are discussed in Section 6.

2. Surface Fitting Algorithm. Surface fitting and smoothing splines techniques are widely used to fit data. We have developed a surface fitting algorithm, TPSFEM, that can be used for various applications such as data mining but also for digital elevation models.

The TPSFEM algorithm can handle data sizes with millions of records. It combines the favorable properties of finite element surface fitting with the ones of thin plate splines. It can be viewed as a discrete thin plate spline. The following is a brief summary of the derivation of the method. The interested reader is referred to [6, 7] for more details.

The standard thin plate spline is the function f_α that minimizes the functional

$$M_\alpha(f) = \frac{1}{n} \sum_{i=1}^n (f(\mathbf{x}^{(i)}) - y_i)^2 + \alpha \int_{\Omega} \left(\frac{\partial^2 f(\mathbf{x})}{\partial^2 x_1} \right)^2 + 2 \left(\frac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_2} \right)^2 + \left(\frac{\partial^2 f(\mathbf{x})}{\partial^2 x_2} \right)^2 d\mathbf{x}$$

where the observations of the predictor and response variables, respectively are given by $\mathbf{x}^{(i)} \in \mathbb{R}^2$, and $y^{(i)} \in \mathbb{R}$ for $i = 1, \dots, n$. An appropriate value for the smoothing parameter α can be determined by generalized cross validation. The smoothing problem will be solved with finite elements.

In order to reduce the complexity of the problem we suggest that very simple elements are used, namely, tensor products of piecewise linear functions. For these functions, however, the required second derivatives do not exist. Thus one has to use a non-conforming finite element principle and a new functional is introduced which only requires first derivatives.

In a first step towards this goal the functional M_α is reformulated in terms of the gradient of f . Consider $\mathbf{u} = (u_1, u_2)$, which will represent the gradient of f . The function f can essentially be recovered from \mathbf{u} as follows.

Let H^1 denote the usual Sobolev space and $u_0 \in H^1(\Omega)$, such that

$$(\nabla u_0, \nabla v) = (\mathbf{u}, \nabla v), \text{ for all } v \in H^1(\Omega) \quad (2.1)$$

and

$$\int_{\Omega} u_0 d\mathbf{x} = 0. \quad (2.2)$$

Let the function values of u_0 at the observation points be denoted by

$$Pu_0 = [u_0(\mathbf{x}^{(1)}) \cdots u_0(\mathbf{x}^{(n)})]^T.$$

Furthermore let

$$X = \begin{bmatrix} 1 & \cdots & 1 \\ \mathbf{x}^{(1)} & \cdots & \mathbf{x}^{(n)} \end{bmatrix}^T \in \mathbb{R}^{n,3} \quad \text{and} \quad \mathbf{y} = [y^{(1)}, \dots, y^{(n)}]^T.$$

We now consider the minimizer of the functional

$$J_\alpha(u_0, u_1, u_2, \mathbf{c}) = \frac{1}{n} (Nu_0 + X\mathbf{c} - \mathbf{y})^T (Nu_0 + X\mathbf{c} - \mathbf{y}) + \alpha(|u_1|_1^2 + |u_2|_1^2)$$

where the minimum is taken over all functions $u_0, u_1, u_2 \in H^1(\Omega)$ with zero mean and constants $\mathbf{c} = [c_0 \ c_1 \ c_2]^T \in \mathbb{R}^3$ subject to constraints (2.1) and (2.2).

The functional has exactly one minimum if the observation points $\mathbf{x}^{(i)}$ are not collinear. The function defined by $f(\mathbf{x}) = u_0(\mathbf{x}) + c_0 + c_1 x_1 + c_2 x_2$ provides a smoother which has essentially the same smoothing properties as the original thin plate smoothing spline. Note that the smoother has been defined in terms of \mathbf{u} . Formal proofs of these results can be found in [8].

After some manipulations and discretization [7] one gets the following linear system of equations which describes the finite element solution of the minimization problem:

$$\begin{bmatrix} \alpha A & 0 & 0 & 0 & -B_1^T \\ 0 & \alpha A & 0 & 0 & -B_2^T \\ 0 & 0 & M & F & A \\ 0 & 0 & F^T & E & 0 \\ -B_1 & -B_2 & A & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \mathbf{u}_0 \\ \mathbf{c} \\ \mathbf{w} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ n^{-1} N^T \mathbf{y} \\ n^{-1} X^T \mathbf{y} \\ 0 \end{bmatrix} \quad (2.3)$$

The symmetric positive definite matrix A is the finite element approximation of the Laplacian. B_1 and B_2 are the approximations of the derivatives of $\frac{\partial}{\partial x_1}$ and $\frac{\partial}{\partial x_2}$, respectively. M , E and F are the matrices assembled from the observation data. They are of the form $M = N^T N$, $E = X^T X$ and $F = N^T X$, respectively. $\mathbf{c} = [c_0 \ c_1 \ c_2]^T \in \mathbb{R}^3$ is constant and \mathbf{w} is the Lagrange multiplier vector associated with the discretization of the constraints (2.1) and (2.2). \mathbf{u}_0 is a discrete approximation to f_α whereas \mathbf{u}_1 and \mathbf{u}_2 are the discrete approximation of the first derivatives of f_α .

3. Solving the Linear Systems. The size of the linear system (2.3) is independent of the number of observations, n . The observation data have to be read from secondary storage only once during the assembly of the matrices M , E and F and the vectors $N\mathbf{y}$ and $X\mathbf{y}$ in (2.3). The assembly phase of our algorithm thus has a time complexity of $O(n)$ to form (2.3). If the number of nodes in the finite element discretization is m , the size of (2.3) is $4m + 3$ which is independent of n . All sub-systems in (2.3) have the dimension m , except the fourth one that has dimension 3. Thus, the total amount of the work required for the TPSFEM algorithm is $O(n)$ to form (2.3) plus the work required to solve it. The sequential time can be modeled like

$$T_{seq}(n, m) = T_{assembly}(n, m) + T_{solve}(m)$$

with $T_{assembly}(n, m)$ the time to process n data points and assemble the matrices and $T_{solve}(m)$ the solver time, which depends only upon the size of the linear system.

It can be seen that the system (2.3) corresponds to a saddle point problem which is sparse and symmetric indefinite. We reorder the system and re-scale it with α to make the solution more stable and get the linear system

$$\begin{bmatrix} \alpha A & 0 & -\alpha B_1^T & 0 & 0 \\ 0 & \alpha A & -\alpha B_2^T & 0 & 0 \\ -\alpha B_1 & -\alpha B_2 & 0 & 0 & \alpha A \\ 0 & 0 & 0 & E & F^T \\ 0 & 0 & \alpha A & F & M \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \mathbf{w} \\ \mathbf{c} \\ \mathbf{u}_0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ n^{-1} X^T \mathbf{y} \\ n^{-1} N^T \mathbf{y} \end{bmatrix} \quad (3.1)$$

The upper left 2×2 block is regular, so the elimination of the two unknown vectors \mathbf{u}_1 and \mathbf{u}_2 results in the reduced linear system

$$\begin{bmatrix} -\alpha G & 0 & \alpha A \\ 0 & E & F^T \\ \alpha A & F & M \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ \mathbf{c} \\ \mathbf{u}_0 \end{bmatrix} = \begin{bmatrix} 0 \\ n^{-1} X^T \mathbf{y} \\ n^{-1} N^T \mathbf{y} \end{bmatrix} \quad (3.2)$$

where $G = B_1 A^{-1} B_1^T + B_2 A^{-1} B_2^T$. Elimination of \mathbf{c} and \mathbf{w} from equation (3.2) leads to the equation

$$(M + \alpha A G^{-1} A - F E^{-1} F^T) \mathbf{u}_0 = n^{-1} (N^T \mathbf{y} - F E^{-1} X^T \mathbf{y}) \quad (3.3)$$

for \mathbf{u}_0 . Recalling that $E = X^T X$, $F = N^T X$ and $M = N^T N$, we can rewrite (3.3) as

$$(N^T (I - X E^{-1} X^T) N + \alpha A G^{-1} A) \mathbf{u}_0 = n^{-1} (N^T \mathbf{y} - F E^{-1} X^T \mathbf{y}) \quad (3.4)$$

where the matrix is symmetric positive definite since the first part is symmetric non-negative and the second part is symmetric positive definite. Hence we can use the conjugate gradient method to solve this equation.

Each step in this outer conjugate gradient loop requires a matrix-vector multiplication $(M - F E^{-1} F^T + \alpha A G^{-1} A) \mathbf{u}$. In particular we need to approximate the action of G^{-1} . The matrix $G = B_1 A^{-1} B_1^T + B_2 A^{-1} B_2^T$ is positive definite and so the conjugate gradient method can be used to calculate the action of G^{-1} as well. In this intermediate conjugate gradient loop we need to be able to calculate the matrix-vector products of the form $G \mathbf{u}$. This requires calculating the action of A^{-1} twice. Once again we use the conjugate gradient method to calculate this action. All in all, we have three nested conjugate gradient loops.

4. Parallel Implementation. The original implementation of this algorithm has been done in Matlab [6, 7]. A first parallel implementation using C and MPI has been presented in [3]. It applies a functional parallelism to solve the linear system, but it is not scalable with the number of processors.

The TPSFEM algorithm mainly consists of two steps: First, the assembly of the matrices and secondly the solution of the linear system (3.1). Both phases are inherently parallel, as data points can be assembled into the matrices independently of each other and each processor solves a smaller linear system on its local domain. Communication is only needed after the assembly step to redistribute local matrices and after solving the linear system where the solution vector is collected on one processor. Parallelism is achieved by a one-dimensional overlapping domain partitioning, where each processor gets the data and computes the solution for a strip of the original domain.

For a given data file a preprocessing has to be done once. It consists of a cyclic splitting of the original data file into P smaller files, where P is the number of used processors. These files can then be processed independently by the processors. If the original file contains n data points (the observation data), each split file will contain

n/P data points. Ideally, these smaller files are stored on local disks on the processors, so network traffic can be prevented in the assembly phase. Preprocessing also includes the computation of statistical data, like the total number of data points n and the minimal and maximal values for each variable. The time for preprocessing scales linearly with $O(n)$, as each data point is processed only once.

In the assembly phase, data is read from the local files and each processor assembles with its local data P local matrices (M , F and E) and vectors ($N\mathbf{y}$ and $X\mathbf{y}$), one for each processor. As the data sets in data mining applications can be quite large, file reading and assembly of the matrices and vectors can become a quite time-consuming step.

The data files have to be read once only and the data points do not have to be stored. The assembly phase is therefore linearly scalable with $O(n)$ as well. As the data points are distributed cyclically into the local files, each processor has to assemble matrices for all processors. A redistribution has to be carried out after all data points have been read. Basically, P reduce operations are performed, after which each processor got $(P - 1)$ local matrices and vectors from other processors (plus its own ones) which are summed to the final local matrices and vectors. The amount of communicated data is of the order

$$O\left(2\frac{m}{P}\log_2 P\right)$$

with m the matrix dimension. Each sent message has a length proportional to $O(m/P)$. The communication phase – as well as the following solution step – is independent of the number of data points n . An almost ideal speedup can be achieved for the assembly process, if the amount of communicated data is small compared to the number of observations to process from file.

As the matrices A , B_1 and B_2 do not depend on the observations, their assembly is only a function of the matrix dimension m and typically requires much less time than the assembly of the data matrices.

The solution step is basically a serial iterative solver for the system (3.1) as described in Section (3) where each processor solves a smaller linear system, i.e. only with its local matrices. The linear system (3.1) to be solved contains five sub-systems, four having a dimension of m and one having dimension 3. At the end of the solution step, the result vector is collected on one processor and saved into a file.

While the dimension of the original sub-matrices is m , in the parallel case the processors solve linear systems with matrix dimension

$$\frac{m}{P} + \begin{cases} v(n_1 + 1) & \text{on the first and last processor} \\ 2v(n_1 + 1) & \text{all other processors} \end{cases}$$

where v is the number of overlapping grid rows and n_1 is the number of grid rows in the first dimension¹. The global grid is split into stripes in the second dimension. Each processor gets $n_2/P + 2v$ grid rows (again the first and last processor get only $n_2/P + v$ rows).

The matrices in (3.1) are sparse and consist of nine diagonals with non-zero elements. Both matrices M and A are symmetric, so only the diagonal and lower part have to be stored. Matrices B_1 and B_2 are non-symmetric, so all nine diagonals are stored. We have chosen a diagonal storage data structure (Figure 4.1), consisting of a two-dimensional array with m rows and 5 or 9 columns, respectively. Each

¹ The sub-matrix dimension is $m = (n_1 + 1)(n_2 + 1)$

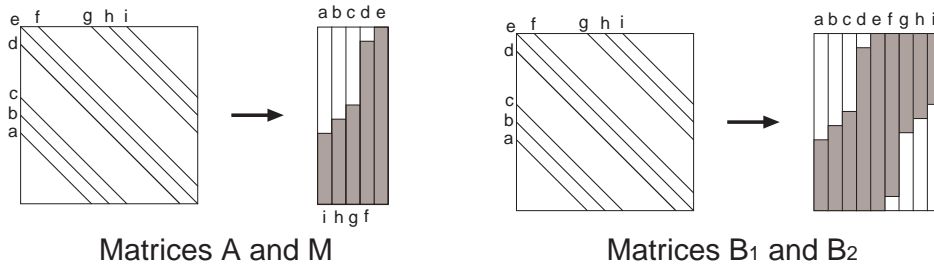


FIG. 4.1. *Sparse Matrix Diagonal Data Structure*

of the columns contains one of the matrix diagonals with non-zero elements. With this packed data structure, good data locality and thus good cache utilization can be achieved on RISC processors. The other matrices used are of dimension $m \times 3$ (matrix F) and 3×3 (matrix E). The dot product and vector addition (DAXPY) implementations use loop unrolling for better RISC pipeline usage.

For portability reasons, we use MPI [11] for communication, although the measurements presented in the next section have been run on an SUN shared-memory multiprocessor. We plan to perform further tests on different platforms. Modeling the parallel time we get:

$$T_{par}(n, m, P, v) = T_{assembly}\left(\frac{n}{P}, \frac{m}{P} + 2v\right) + T_{solve}\left(\frac{m}{P} + 2v\right)$$

One aspect neglected in this model – as well as in this parallel implementation presented here – is load balancing. As the first and last processor get less rows (as they only have one overlapping grid area) they may generally be faster. On the other hand, as each processor now has different data, the number of iterations needed may vary.

5. Parallel Results. The TPSFEM algorithm has already been applied to fit surfaces of large data sets from insurance, flow field, digital elevation and magnetic field areas. We demonstrate the efficiency of the presented parallel TPSFEM implementation by using two large data sets, one with digital elevation and the other with magnetic field data. The algorithm can equally be applicable in data mining as a non-parametric regression technique.

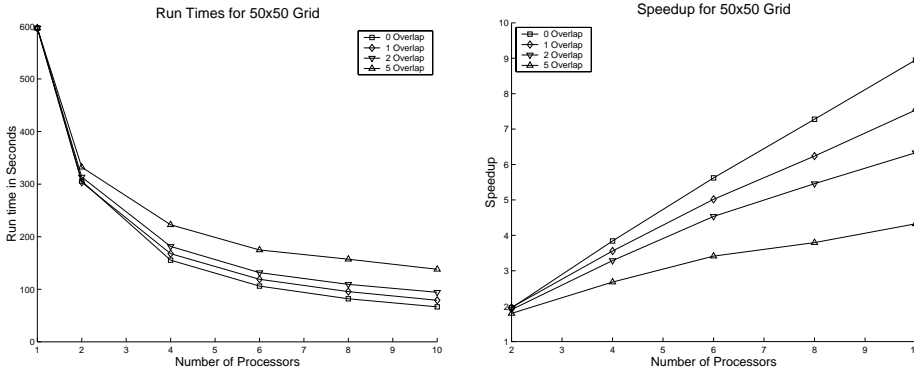


FIG. 5.1. Runtime and Speedup for a 50×50 Grid on Elevation Data

Digital elevation data is an important data component required by Geographical Information Systems (GIS) [12]. An efficient surface fitting algorithm such as TPSFEM is an important tool for GIS applications. The involvement of end-users in this area is usually interactive. Therefore, they can benefit significantly from a fast surface fitting algorithm.

We used a digital elevation data set which has been obtained by digitizing the map of the Murrumbidgee region in Australia. The data set includes $n = 1'887'250$ observation points. As a test platform we used a SUN Enterprise 4000 symmetric-multiprocessor (SMP) with 10 Ultra-Sparc processors, 4.75 GByte main memory and 256 GByte disk-memory. The original data file has been split into $P = 2, 4, 6, 8$ and 10 smaller files, so each process could read and process a file independently. As measurements showed, reading the data from disks scales almost linearly with the number of running processes. We run tests with different grid resolutions and various numbers of overlapping grid rows (0, 1, 2 and 5). The achieved run times in seconds and the resulting speedup values for grid resolutions of 50×50 and 100×100 can be seen in Figures 5.1 and 5.2, respectively.

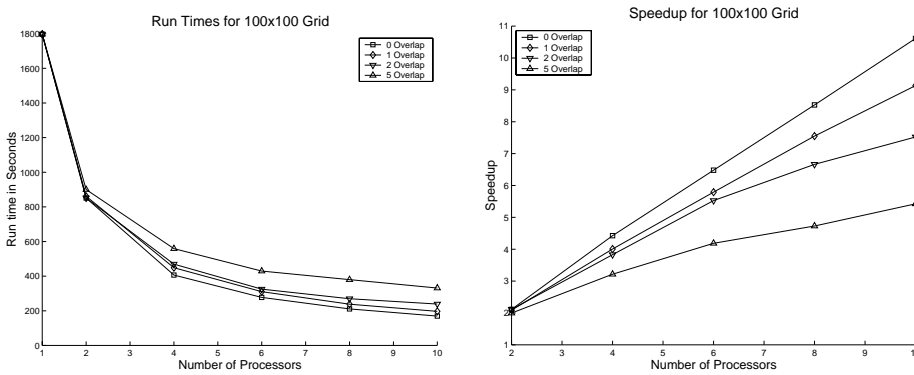


FIG. 5.2. Runtime and Speedup for a 100×100 Grid on Elevation Data

The number of overlapping grid rows – and thus the size of the resulting linear systems – clearly influences the run time. Adding no overlapping grid rows results in the shortest run times, but also in the poorest smoothed surface, as no good smoothing over the processor boundaries is achieved. Finding the optimal number of overlapping

grid rows – both concerning the smoothed surface as well as the run time – is quite hard to do, as this depends on the grid resolution, the number of processors, the value of the smoothing parameter α and on the observation data points.

The following Table 5 shows some more details of the above tests. Times in milliseconds for assembling the data matrices, redistributing them, assembling the FEM matrices and finally solving the linear system are presented for the 100×100 grid with two overlapping grid rows. The average times are presented for the assembly and solving routine, whereas the maximum is listed for the redistribution (communication).

Processors	1	2	4	6	8	10
Assemble M	264913	133228	67261	45275	34224	27963
Redistribute	–	290	375	198	690	2232
Assemble FEM	70	35	18	11	9	8
Solve System	1532320	710355	469486	321239	261915	235308

TABLE 5.1
Timing Details for 100×100 Grid with 2 Overlapping Rows

In order to illustrate the parallel TPSFEM algorithm we presented a magnetic field surface [1] obtained from $n = 735'700$ points. Figure 5.3 shows a 30×30 grid computed on one and five processors, respectively. The number of overlapping grid rows has been set to 0, 2 and 6 with an smoothing parameter $\alpha = 0.00001$. It can be seen how the surface gets smoother in the second dimension with more overlapping grid rows, but it gets never as good as in the serial run. This indicates that we have to choose different values for the smoothing parameter α in the parallel version. For this example the serial run time was 155s, and for the parallel run it was 52s, 68s and 107s for 0, 2 and 6 overlapping grid rows respectively.

6. Conclusions. In this paper we presented a scalable parallel version of the TPSFEM algorithm, which can handle very large data sets to fit smooth surfaces. There are two main steps in the algorithm that consume most processing time and are easy to parallelize: First, the assembly of the matrices in (2.3) and secondly solving the system (3.1). For very large data sets an almost ideal speedup can be achieved in the assembly step. Using an overlapping one-dimensional domain decomposition, reasonable speedups can also be achieved for the solution step, with a tradeoff between smoothing accuracy and run time. We hope to improve the solution step by using multigrid solvers. Theoretical considerations concerning an optimal value for the overlapping grid rows as well as load balancing aspects are also part of our future research.

Acknowledgements. Part of this research is supported by the Australian Advanced Computational Systems CRC. Peter Christen is funded by grants from the *Swiss National Science Foundation* (SNF) and the *Novartis Stiftung, vormalis Ciba-Geigy Jubiläums-Stiftung*.

REFERENCES

- [1] Australian Society of Exploration Geophysicists,
<http://www.aseg.org.au/asegduc/menu.htm>
- [2] Data Mining: An Introduction, Data Distilleries, DD19961, 1996,
<http://www.ddi.nl>
- [3] P. CHRISTEN, I. ALTAS, M. HEGLAND, S. ROBERTS, K. BURRAGE AND R. SIDJE, *A Parallel*

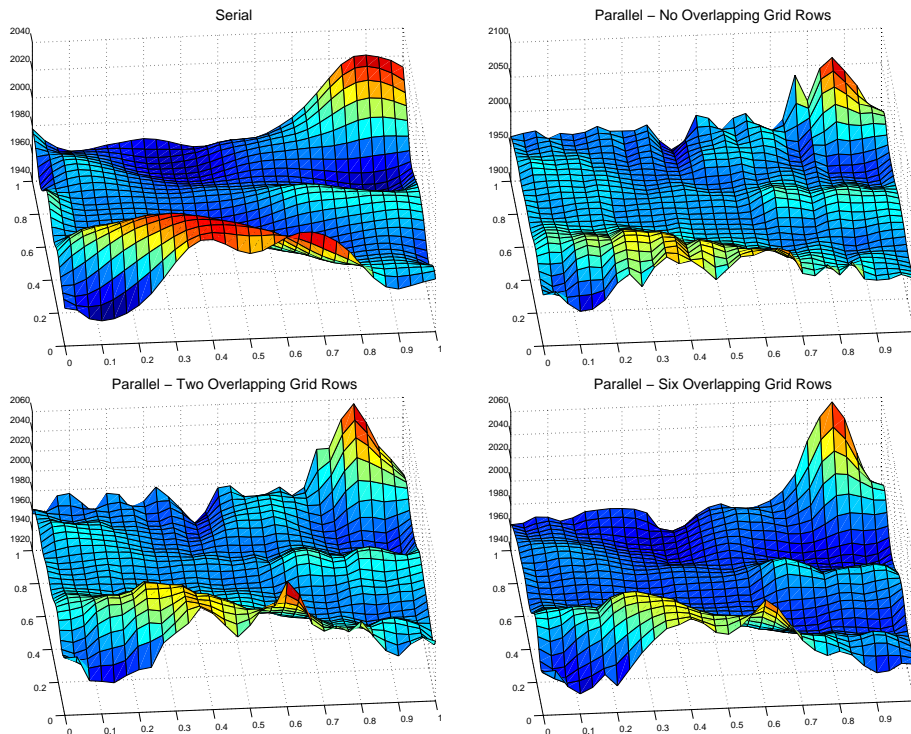


FIG. 5.3. *Smoothed Magnetic Field Surface, on 1 and 5 Processors*

Finite Element Surface Fitting Algorithm for Data Mining, Proceedings of the PARCO-99 Conference, Delft, Holland.

- [4] A. A. FREITAS AND S. H. LAVINGTON, *Mining Very Large Databases with Parallel Processing*, Kluwer Academic Publishers, 1998.
- [5] T. J. HASTIE AND R. J. TIBSHIRANI, *Generalized Additive Models*, Monographs Chapman and Hall, 1990.
- [6] M. HEGLAND, S. ROBERTS AND I. ALTAS, *Finite Element Thin Plate Splines for Data Mining Applications*, in *Mathematical Methods for Curves and Surfaces II*, M. Daehlen, T. Lyche and L. L. Schumaker, eds., pp. 245–253, Vanderbilt University Press, 1998.
- [7] M. HEGLAND, S. ROBERTS AND I. ALTAS, *Finite Element Thin Plate Splines for Surface Fitting*, in *Computational Techniques and Applications: CTAC97*, B. J. Noye, M. D. Teubner and A. W. Gill, eds., pp. 289–296, World Scientific, 1997.
- [8] S. ROBERTS, M. HEGLAND AND I. ALTAS, H^1 *Finite Element Thin Plate Splines*, in preparation, 1999.
- [9] M. FORTIN AND R. GLOWINSKI, *Augmented Lagrangian Methods: Applications to the Numerical Solution of Boundary-Value Problems*, North-Holland, 1983.
- [10] H. C. ELMAN AND H. G. GOLUB, *Inexact and Preconditioned Uzawa Algorithms for Saddle Point Problems*, *SIAM J. Numer. Anal.*, (31) 1994, pp. 1645–1661.
- [11] W. GROPP, E. LUSK AND A. SKJELLUM, *Using MPI – Portable Parallel Programming with the Message-Passing Interface*, The MIT Press, Cambridge, Massachusetts, 1994.
- [12] L. LANG, *GIS Goes 3D*, *Computer Graphics World*, March 1989, pp. 3-8-46.