A Scalable Parallel FEM Surface Fitting Algorithm for Data Mining

Peter Christen*[1], Markus Hegland[1], Stephen Roberts[1], Ole M. Nielsen[1], and Irfan Altas[2]

[1] Australian National University, Canberra, ACT 0200, Australia
URL: `http://csl.anu.edu.au/ml/dm/`

[2] School of Information Studies, Charles Sturt University, Wagga Wagga, NSW 2678, Australia

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 determination of interaction effects in predictive models which uses surface fitting. Thin plate splines proved a good tool for surface fitting but require in general the solution of a dense linear system of equations which has as many unknowns as the data set has records. Thus such an approach is infeasible for data mining with millions of records.

We have developed a finite element approximation of a thin plate spline that can handle data sizes with millions of records. Each data record has to be read from an external file once only and there is no need to store the data in memory. For the efficient solution of the resulting system of equations (which is much smaller than the one for ordinary thin plate splines) a new parallel algorithm has been developed. First results obtained on a Sun shared-memory multiprocessor are presented.

1 Introduction

With *Data Mining* one designates techniques to automatically process and detect patterns in very large data sets [6]. 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 computerisation of business transactions and use of bar codes in commercial outlets have provided businesses with enormous amounts of data. Nowadays, Terabyte databases are common, with Gigabytes added every day. Revealing patterns and relationships in such data sets can improve the goals, missions and objectives of many organisations. For example, sales records can reveal highly profitable retail sales patterns.

* Corresponding author, E-Mail: Peter.Christen@anu.edu.au

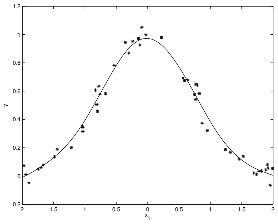

Fig. 1. One-dimensional smoothing spline from 50 data points.

A typical data mining application is the analysis of insurance data where there may be more than several hundred thousand policies with tens of attributes each. Insurance companies analyse such data sets to understand claim behaviour. Some insurants lodge more claims than others do. What are the typical characteristics of insurants who lodge many claims? Would it be possible to predict whether a particular insurant 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.

Predictive models provide the main tool to answer these questions [6, p.319]. Such models are real-valued functions of the form $y = f(x_1, \ldots, x_d)$. They allow the prediction of costs, probabilities of events and physical quantities. Often linear models are used, however, in general the functions f contain nonlinear trends and often it is only known that they are smooth. In the case of only one predictive variable x_1 smooth relationships are commonly modelled with smoothing splines. Figure 1 shows an example of a smoothing spline. This spline is obtained as the minimum of $\frac{1}{n}\sum_{i=1}^{n}(f(x_1^{(i)}) - y^{(i)})^2 + \alpha \int f''(x_1)dx_1$ where the smoothing parameter α controls the tradeoff between fit to the data $(x^{(i)}, y^{(i)})$ and smoothness. When generalising this approach to higher dimensions, one immediately faces the *curse of dimensionality* [7] as the complexity of functions grows exponentially with the dimension which is of major concern as data mining problems are notoriously high-dimensional. One particularly successful approach in dealing with this challenge uses sums of one-dimensional functions f_i to approximate high-dimensional predictors f:

$$f(x_1, \ldots, x_d) = \sum_{i=1}^{d} f_i(x_i).$$

The components f_i of this *additive model* can be determined iteratively using one-dimensional smoothing splines only with the *backfitting algorithm* [7]. The models are similar to the ones used in the analysis of variance but have the disadvantage that no interactions between the predictive variables are included. For example, the effect of the age on the risk of a person lodging an insurance claim may be different for women and men. Models of the form

$$f(x_1, \ldots, x_d) = \sum_{i,j=1}^{d} f_{i,j}(x_i, x_j)$$

include such interactions $f_{i,j}$ which can again be determined by backfitting. In this case, however, two-dimensional splines are used. The effective determination of these two-dimensional splines for the interaction models used in data mining with very large data sets is the topic of this paper. Techniques for determining such interactions are also called *surface fitting*.

We have developed a generic surface fitting algorithm that can handle data sizes with millions of observations. They have been specifically designed for the determination of interactions in predictive models for data mining applications. The algorithm combines the favourable 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 analyse 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 with Finite Elements

We briefly summarise the TPSFEM algorithm for surface fitting. The interested reader is referred to [8, 9] for more details. The standard thin plate spline is the function f_α that minimises 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 $\mathbf{x}^{(i)} \in \mathbb{R}^2$ and $y^{(i)} \in \mathbb{R}$ for $i = 1, ..., n$ are the data records.

Highly efficient function evaluation is obtained by a piecewise bilinear approximation. This approximation, however, necessitates the derivation of a *non-conforming finite element* method in order to be applicable to the minimisation of M_α due to the low smoothness. First auxiliary unknowns $\mathbf{u} = (u_1, u_2)$ and u_0 are introduced. The $\mathbf{u}$ should approximate the gradient of u_0 and

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

where $H^1(\Omega)$ denotes the Sobolev space of smooth functions and

$$\int_\Omega u_0 \, d\mathbf{x} = 0. \tag{2}$$

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

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

Now consider the minimiser 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 (1) and (2).

The functional has exactly one minimum if the observation points $\mathbf{x}^{(i)}$ are not collinear and $f(\mathbf{x}) = u_0(\mathbf{x}) + c_0 + c_1x_1 + c_2x_2$ provides a smoother which has essentially the same smoothing properties as the original thin plate smoothing spline. The mathematical details can be found in [14].

Discretisation [8] leads to the linear system of equations for the value of the functions u_i

$$\begin{bmatrix} \alpha A & 0 & 0 & 0 & -B_1^T \\ 0 & \alpha A & 0 & 0 & -B_2^T \\ 0 & 0 & M & F & A \\ 0 & 0 & F^T & E & 0 \\ -B_1 & -B_2 & A & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \mathbf{u}_0 \\ \mathbf{c} \\ \mathbf{w} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ n^{-1}N^T\mathbf{y} \\ n^{-1}X^T\mathbf{y} \\ 0 \end{bmatrix} \tag{3}$$

The symmetric positive definite matrix A is the finite element approximation of the Laplacian. The vector $\mathbf{w}$ is the Lagrange multiplier vector associated with the discretisation of the constraints (1) and (2).

3 Solving the Linear Systems

The size of the linear system (3) is independent of the number of data records, n. The data has 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 (3). The assembly step of our algorithm thus has a time complexity of $O(n)$ to form (3). The total sequential run time can be modelled as

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

with $T_{assembly}(n, m)$ the time to load n data records 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 (3) corresponds to a saddle point problem which is sparse and symmetric indefinite. We reorder the system and rescale it with α to make the solution more stable and get the linear system

$$\begin{bmatrix} \alpha A & 0 & -\alpha B_1^T & 0 & 0 \\ 0 & \alpha A & -\alpha B_2^T & 0 & 0 \\ -\alpha B_1 & -\alpha B_2 & 0 & 0 & \alpha A \\ 0 & 0 & 0 & E & F^T \\ 0 & 0 & \alpha A & F & M \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \mathbf{w} \\ \mathbf{c} \\ \mathbf{u}_0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ n^{-1}X^T\mathbf{y} \\ n^{-1}N^T\mathbf{y} \end{bmatrix} \tag{4}$$

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

$$\begin{bmatrix} -\alpha G & 0 & \alpha A \\ 0 & E & F^T \\ \alpha A & F & M \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ \mathbf{c} \\ \mathbf{u}_0 \end{bmatrix} = \begin{bmatrix} 0 \\ n^{-1}X^T\mathbf{y} \\ n^{-1}N^T\mathbf{y} \end{bmatrix} \tag{5}$$

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 (5) leads to the equation

$$\left(M + \alpha A G^{-1} A - F E^{-1} F^T\right) \mathbf{u}_0 = n^{-1}(N^T\mathbf{y} - F E^{-1} X^T \mathbf{y}) \tag{6}$$

for $\mathbf{u}_0$. Recalling that $E = X^T X$, $F = N^T X$ and $M = N^T N$, (6) can be rewritten as

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

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

Earlier prototype implementations of the TPSFEM either used only one processor [8, 9] or did not scale well in the number of processors [1]. This is why a new implementation was derived. It is based on the fact that both the assembly and and solution steps are inherently parallel and communication is only required to complete the assembly step.

In a preprocessing step a data set is split into p smaller files which are stored on local discs, and basic statistics about the data set are gathered. This step only requires to be done once and is $O(n)$. After this, the assembly step is performed independently on each processor. At the end the locally assembled matrices are

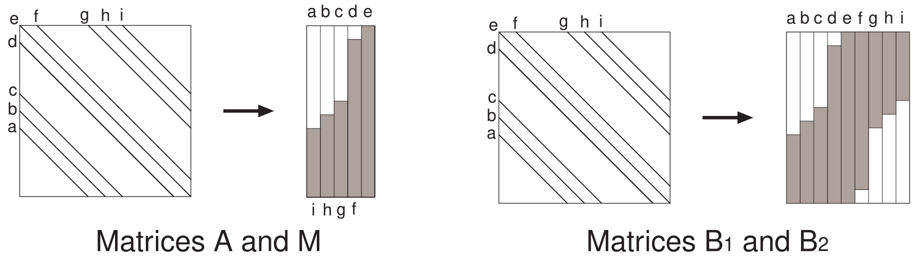

Fig. 2. Sparse matrix diagonal data structure.

summed up and redistributed. In this step the data is communicated in blocks of size $O(m/p)$ and the total communication volume is

$$O(2\frac{m}{p}log_2p).$$

This is independent of the number of data records. The assembly of A, B_1 and B_2 is cheap as it does not require reading the data.

The solution step is done using an overlapping domain partitioning. Basically each processor solves its own independent linear system. The smoothness of the solution is obtained by having a certain overlap of the domains. 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 dimension[1]. The global grid is splitted into stripes in the second dimension. Each processor gets $n_2/p + 2v$ grid rows (again the first an last processor get only $n_2/p + v$ rows), with n_2 the number of grid points in the second dimension.

The matrices in (4) are sparse and consist of nine diagonals with non-zero elements (2-dimensional case). Symmetric storage for M and A is used. All matrices are stored using a diagonal storage scheme, see Figure 2. This packed data structure results in good data locality and thus good cache utilisation can be achieved on RISC processors.

For portability reasons, we use MPI [5] for communication, although the measurements presented in the next section have been run on an SUN shared-memory multiprocessor. We plan do perform further tests on other parallel platforms. Modelling the parallel run time we get:

$$T_{par}(n,m,p,v) = \ T_{assembly}(\frac{n}{p}, \frac{m}{p} + 2v) + T_{solve}(\frac{m}{p} + 2v).$$

---

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

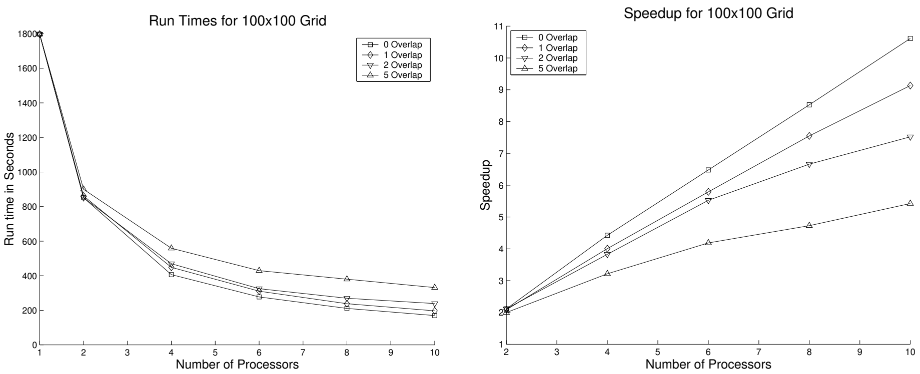

Fig. 3. Run time and speedup for a 100×100 grid on elevation data.

One aspect neglected in this model – as well as in the 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. As each processor has different data, the number of iterations needed to solve the local linear system may vary which further contributes to load imbalance.

5 Parallel Results

The TPSFEM algorithm has already been applied to fit surfaces of large data sets from business and science. Here we demonstrate its efficiency on two large data sets, one with digital elevation data and the other with magnetic field data.

We used a digital elevation data set which has been obtained by digitising the map of the Murrumbidgee region in Australia. The data set includes $n = 1,887,250$ data records. As a test platform we used a SUN Enterprise 4000 symmetric-multiprocessor (SMP) with 10 Ultra-Sparc processors, 4.75 Gigabytes main memory and 256 Gigabytes of disk space. The run times for different numbers of overlapping grid rows $(0, 1, 2$ and $5)$ for a 100×100 grid can be seen in Figure 3.

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 process 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 processes, the value of the smoothing parameter α and on the data. Further information about these tests can be found in Table 1. Average times are presented for the assembly and solving routine, whereas the maximum is listed for the redistribution (communication) of the linear system.

Table 1. Timing in seconds for 100×100 grid with 2 overlapping rows.

Processors	1	2	4	6	8	10
Assemble M	264,913	133,228	67,261	45,275	34,224	27,963
Redistribute	–	290	375	198	690	2,232
Assemble FEM	70	35	18	11	9	8
Solve System	1,532,320	710,355	469,486	321,239	261,915	235,308

In order to illustrate the parallel TPSFEM algorithm we present a magnetic field surface[2] obtained from $n = 735,700$ records. Figure 4 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 $\alpha = 10^{-5}$. 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 sequential run. This indicates that we have to choose different values for the smoothing parameter α for parallel runs. For this example the sequential 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 Ongoing Research and 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 parallelise: First, the assembly of the matrices and secondly the solution of the resulting linear system. For very large data sets an almost ideal speedup has been obtained for the assembly. For the solution using an overlapping one-dimensional domain decomposition, reasonable speedups are obtained.

We are currently working on two other methods based on smoothing. A first technique called ADDFIT [10] is based on additive models and does not contain any interactions. A second technique called HISURF [11] contains interactions which have been approximated using wavelets. These techniques are compared with the TPSFEM technique in Figure 5 for the approximation of the dependency of a geomagnetic field on two spatial coordinates. Further details and comparisons are given in [2]. We are also taking steps towards implementing these methods in a framework for use in data mining consultancies [3, 12, 13].

Acknowledgements

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

[2] Australian Society of Exploration Geophysicists, `http://www.aseg.org.au`

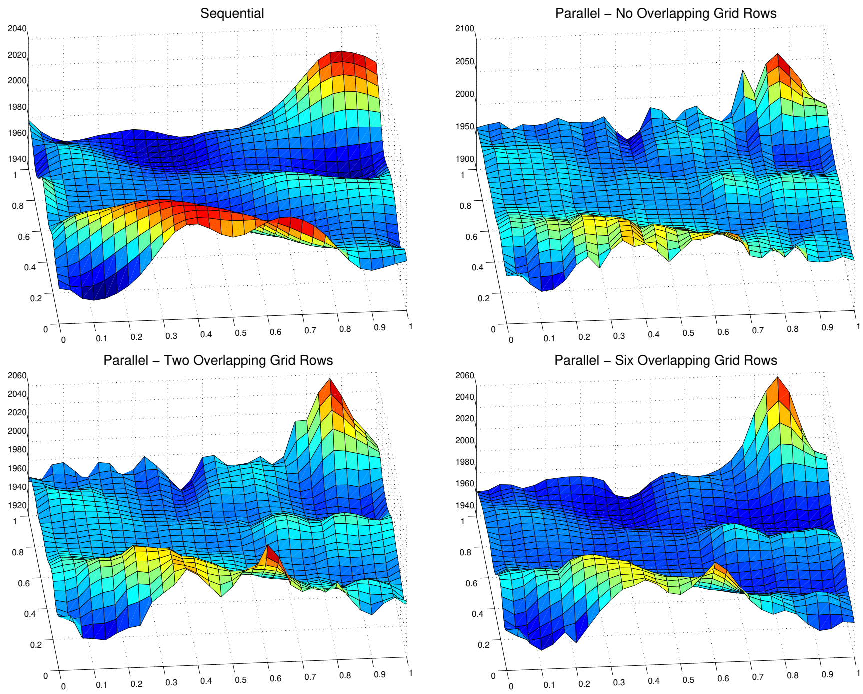

Fig. 4. Smoothed magnetic field surface, on 1 and 5 processors.

References

1. P. Christen, I. Altas, M. Hegland, S. Roberts, K. Burrage and R. Sidje: A Parallel Finite Element Surface Fitting Algorithm for Data Mining, Proceedings of the Parco-99 Conference, Delft, Holland.
2. P. Christen, M. Hegland, O.M. Nielsen, S. Roberts, P.E. Strazdins and I. Altas: Scalable Parallel Algorithms for Surface Fitting and Data Mining, accepted by the Elsevier Journal of Parallel Computing, special issue on Aspects of Parallel Computing for Linear Systems and Associated Problems, September 2000.
3. P. Christen, M. Hegland, O.M. Nielsen, S. Roberts, P. Strazdins, T. Semenova and T. Hancock: Towards a Parallel Data Mining Toolbox, accepted for the PDDM-2001 Workshop, San Francisco, April 2001.
4. A.A. Freitas and S.H. Lavington: Mining Very Large Databases with Parallel Processing, Kluwer Academic Publishers, 1998.
5. W. Gropp, E. Lusk and A. Skjellum: Using MPI – Portable Parallel Programming with the Message-Passing Interface, MIT Press, Cambridge, Massachusetts, 1994.
6. J. Han and M. Kamber: Data Mining – Concepts and Techniques, Morgan Kaufmann Publisher, August 2000.
7. T.J. Hastie and R.J. Tibshirani: Generalized Additive Models, Monographs Chapman and Hall, 1990.
8. 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., World Scientific, 1997.

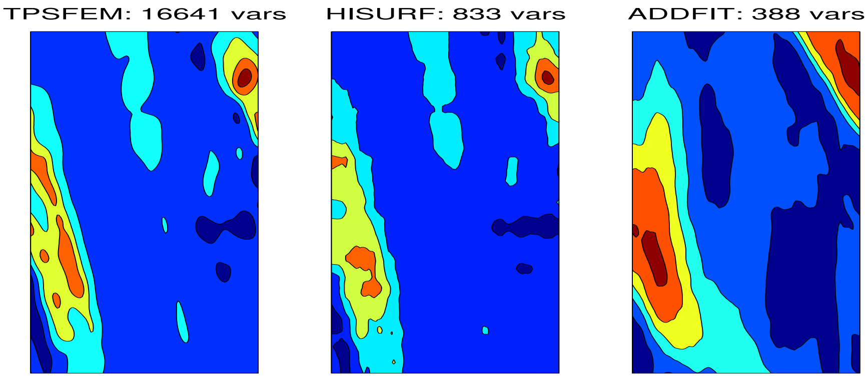

Fig. 5. Modelling 2D aeromagnetic data with 735,700 records using TPSFEM, HISURF, and ADDFIT.

9. 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., Vanderbilt University Press, 1998.
10. M. Hegland and V. Pestov: Additive Models in High Dimensions, Research Report 99–33, School of Mathematical and Computing Sciences, Victoria University of Wellington, 1999.
11. M. Hegland, O.M. Nielsen and Z. Shen: High Dimensional Smoothing Based on Multilevel Analysis, submitted to SIAM J. Scientific Computing, November 2000.
12. O.M. Nielsen, P. Christen, M. Hegland and T. Semenova: Data Mining with Python, Proceedings of the 9th International Python Conference, Foretex Sem. Inc., March 2001.
13. O.M. Nielsen, P. Christen, M. Hegland and T. Semenova: A Toolbox Approach to Flexible and Efficient Data Mining, accepted for the Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'2001), Hong Kong, April 2001.
14. S. Roberts, M. Hegland and I. Altas: H^1 Finite Element Thin Plate Splines, submitted, 2001.