Discrete thin plate spline smoothing in 3D

S. Roberts[*] L. Stals[†]

(Received 7 August 2003; revised 31 January 2004)

Abstract

The thin plate spline method is often used to fit data in high dimensions. Standard thin plate splines require the solution of a dense linear system of equations whose size increases with the number of data points and can be expensive when used on large data sets. In this paper we present a discrete thin plate spline method that uses polynomials with local support defined on finite element grids. The resulting system of equations is sparse and its size depends only on the number of nodes in the finite element grid so this method is efficient when dealing with large data sets. Theory is developed for general d-dimensional data sets and several example results are given for 3D models.

[*]Dept. of Maths, Australian National University, Canberra, Australia. mailto:Stephen.Roberts@anu.edu.au

[†]as above, mailto:stals@maths.anu.edu.au

See http://anziamj.austms.org.au/V45/CTAC2003/Rob2/home.html for this article, © Austral. Mathematical Soc. 2004. Published July 18, 2004. ISSN 1446-8735

Contents

1 Introduction

The problem of fitting data in high dimensions arises in a number of applications including; data mining, 3D reconstruction of geometric models, finger print matching, image warping, medical image analysis and optic flow computations.

A commonly used technique to fit the data is the thin plate spline method. This method is favoured because it is insensitive to noise in the data and it minimises the bending energy of a thin-shell object. The thin plate spline for a general domain Ω, as formulated by Wahba [12] and Duchon [5], is the function f that minimises the functional

$$J_\alpha(f) = \frac{1}{n}\sum_{i=1}^{n}\left[f(\boldsymbol{x}^{(i)}) - y^{(i)}\right]^2 + \alpha\int_\Omega \sum_{|\boldsymbol{\nu}|=2}\binom{2}{\boldsymbol{\nu}}(D^{\boldsymbol{\nu}} f(\boldsymbol{x}))^2\,d\boldsymbol{x}\,, \qquad (1)$$

where $\boldsymbol{\nu} = (\nu_1, \ldots, \nu_d)$ is a d dimensional multi-index, $|\boldsymbol{\nu}| = \sum_{s=1}^{d} \nu_s$, the predictor variable $\boldsymbol{x}$ is in $\boldsymbol{R}^d$, and $\boldsymbol{x}^{(i)}$ and $y^{(i)}$ are the corresponding ith predictor and response data value ($1 \leq i \leq n$). The smoothing parameter α controls the trade-off between smoothness and fit. In the limit $\alpha \to 0$ the function f becomes an interpolant. If α is large, f becomes very smooth but may not reflect the data very well. Techniques for choosing α appropriately can be found in [12].

Often radial basis functions are used to represent f as they give an analytical solution to the minimum of the functional in Equation (1):

$$f(\boldsymbol{x}) = \sum_{k=1}^{M} a\phi_k(\boldsymbol{x}) + \alpha \sum_{i=1}^{n} w_i U(\boldsymbol{x}, \boldsymbol{x}^{(i)}),$$

where ϕ_k are monomials of order up to 1 and U are suitable radial basis functions. The coefficients $\boldsymbol{a}^T = (a_1, \ldots, a_M)$ and $\boldsymbol{w} = (w_1, \ldots, w_n)$, are found by solving

$$\begin{aligned} (K + n\alpha I)\boldsymbol{w} + P\boldsymbol{a} &= \boldsymbol{y}, \\ P^T\boldsymbol{w} &= 0, \end{aligned}$$

where $K_{ij} = U(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)})$, $P_{ij} = \phi_j(\boldsymbol{x}^{(i)})$ and I is the identity matrix. See for example [2, 8].

The system is dense and its size is directly proportional to the number of data points. In the case of, for example, data mining, 3D reconstruction of models from MRI data or interpolation of geophysical data sets, the number of data points is very large and hence the system is very expensive to compute. Although this initial approach was improved by a number of later works [3, 9] these techniques still lead to complex data structures and algorithms that usually require $\mathcal{O}(n)$ amount of memory.

In this paper we propose a discrete thin-plate spline method that uses polynomial basis function with local support defined on a finite element mesh.

The resulting system of equations is sparse and its size depends only on the number of grid points in the finite element mesh. A preconditioned conjugate gradient (CG) method is used to solve the system of equations. This method is computationally cheap for the interpolation of large data sets. We demonstrate this technique by reconstructing models in 3D.

2 Discrete thin plate splines

Standard thin plate splines require the solution of a dense linear system of equations whose size increases with the number of data points and therefore is expensive when used on large data sets. Roberts et al. [4, 6, 7] have developed a finite element approximation of thin plate splines

The smoothing problem is approximated with finite elements so that the discrete smoother f is a piecewise multi-linear function. In vector notation

$$f(\boldsymbol{x}) = \boldsymbol{b}(\boldsymbol{x})^T \boldsymbol{c}\,.$$

The idea is to minimise J_α over all f of this form. Unfortunately the smoothing term is not defined for piecewise multi-linear functions, but we use the non-conforming finite element principle to introduce piecewise multi-linear functions $\boldsymbol{u} = (\boldsymbol{b}^T\boldsymbol{g}_1, \ldots, \boldsymbol{b}^T\boldsymbol{g}_d)$ to represent the gradient of f. The functions f and $\boldsymbol{u}$ satisfy the relationship

$$\int_\Omega \nabla f(\boldsymbol{x}) \cdot \nabla v(\boldsymbol{x})\, d\boldsymbol{x} = \int_\Omega \boldsymbol{u}(\boldsymbol{x}) \cdot \nabla v(\boldsymbol{x})\, d\boldsymbol{x}\,, \tag{2}$$

for all piecewise multi-linear function v. This is equivalent to the relationship

$$L\boldsymbol{c} = \sum_{s=1}^{d} G_s \boldsymbol{g}_s\,, \tag{3}$$

where L is a discrete approximation to the negative Laplace operator and $(G_1, \ldots, G_d)$ is a discrete approximation to the gradient operator.

Now consider the minimiser of the functional

$$\begin{aligned} & J_\alpha(\boldsymbol{c}, \boldsymbol{g}_1, \ldots, \boldsymbol{g}_d) \\ &= \frac{1}{n}\sum_{i=1}^{n}\left[\boldsymbol{b}(\boldsymbol{x}^{(i)})^T\boldsymbol{c} - y^{(i)}\right]^2 + \alpha\int_\Omega \sum_{s=1}^{d} \nabla(\boldsymbol{b}^T\boldsymbol{g}_s)\cdot\nabla(\boldsymbol{b}^T\boldsymbol{g}_s)\, d\boldsymbol{x} \\ &= \frac{1}{n}\sum_{i=1}^{n}\left[\boldsymbol{b}(\boldsymbol{x}^{(i)})^T\boldsymbol{c} - y^{(i)}\right]^2 + \alpha\sum_{s=1}^{d}\boldsymbol{g}_s^T L \boldsymbol{g}_s\,. \end{aligned} \tag{4}$$

Our smoothing problem consists of minimising this functional over all vectors $\boldsymbol{c}$, $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_d$ subject to the Constraint (3).

The boundary conditions are assumed to be Dirichlet with $\boldsymbol{c}$, $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_d$ all set to zero along the boundary. This means that the interpolating spline will be clamped and set to zero at the boundary. Other types of boundary conditions may be used and this will be the subject of future research.

The function defined by $f(\boldsymbol{x}) = \boldsymbol{b}(\boldsymbol{x})^T\boldsymbol{c}$ provides a smoother that has essentially the same smoothing properties as the original thin plate smoothing spline, provided the discretisation is small enough [7].

3 Interpolation splines of 3D data sets

In the 3D case the discrete minimisation problem (4) is equivalent to finding the minimum of

$$J_\alpha(\boldsymbol{c}, \boldsymbol{g}_1, \boldsymbol{g}_2, \boldsymbol{g}_3) = \boldsymbol{c}^T A\boldsymbol{c} - 2\boldsymbol{d}^T\boldsymbol{c} + \boldsymbol{y}^T\boldsymbol{y} + \alpha\left(\boldsymbol{g}_1^T L\boldsymbol{g}_1 + \boldsymbol{g}_2^T L\boldsymbol{g}_2 + \boldsymbol{g}_3^T L\boldsymbol{g}_3\right)\,, \tag{5}$$

subject to

$$L\boldsymbol{c} - G_1\boldsymbol{g}_1 - G_2\boldsymbol{g}_2 - G_2\boldsymbol{g}_2 = 0\,. \tag{6}$$

The matrices L, G_1, G_2 and G_3 are independent of the data points but the matrix

$$A = \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{b}(\boldsymbol{x}^{(i)})\boldsymbol{b}(\boldsymbol{x}^{(i)})^T\,,$$

and vector

$$\boldsymbol{d} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{b}(\boldsymbol{x}^{(i)})y^{(i)}\,,$$

must be assembled by sweeping through the data points. The matrix A is symmetric indefinite and sparse.

Using Lagrange multipliers, the Minimisation Problem (5) is rewritten as the solution of the linear system

$$\begin{bmatrix} A & 0 & 0 & 0 & L \\ 0 & \alpha L & 0 & 0 & -G_1^T \\ 0 & 0 & \alpha L & 0 & -G_2^T \\ 0 & 0 & 0 & \alpha L & -G_3^T \\ L & -G_1 & -G_2 & -G_3 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{c} \\ \boldsymbol{g}_1 \\ \boldsymbol{g}_2 \\ \boldsymbol{g}_3 \\ \boldsymbol{w} \end{bmatrix} = \begin{bmatrix} \boldsymbol{d} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \tag{7}$$

where $\boldsymbol{w}$ is a Lagrange multiplier associated with Constraint (6).

One of the advantages of this approach is that the size of the linear system depends on the discretisation size m instead of the number of the data points n. All sub-systems in (7) have dimension m. The time required to assemble the matrix does depend on n (that is, to build A and $\boldsymbol{d}$) but it depends linearly on n and the observation data only have to be read from secondary storage once if using a uniform finite element grid.

3.1 Solution of linear system

One approach to solve Equation (7) is to eliminate all the variables except $\boldsymbol{g}_1$, $\boldsymbol{g}_2$ and $\boldsymbol{g}_3$: that is,

$$\begin{bmatrix} \alpha L + G_1^T Z G_1 & G_1^T Z G_2 & G_1^T Z G_3 \\ G_2^T Z G_1 & \alpha L + G_2^T Z G_2 & G_2^T Z G_3 \\ G_3^T Z G_1 & G_3^T Z G_2 & \alpha L + G_3^T Z G_3 \end{bmatrix} \begin{bmatrix} \boldsymbol{g}_1 \\ \boldsymbol{g}_2 \\ \boldsymbol{g}_3 \end{bmatrix} = \begin{bmatrix} G_1^T L^{-1}\mathbf{d} \\ G_2^T L^{-1}\mathbf{d} \\ G_3^T L^{-1}\mathbf{d} \end{bmatrix}, \tag{8}$$

where $Z = L^{-1}AL^{-1}$, $\boldsymbol{c} = L^{-1}(G_1\boldsymbol{g}_1 + G_2\boldsymbol{g}_2 + G_3\boldsymbol{g}_3)$. See that the system (8) is symmetric positive definite and thus is solved efficiently using the preconditioned CG method. The preconditioner used here is

$$M = \begin{bmatrix} L^{-1} & 0 & 0 \\ 0 & L^{-1} & 0 \\ 0 & 0 & L^{-1} \end{bmatrix}.$$

Applying Z to a vector equates to solving a system of equations involving the Laplacian; similarly when applying the preconditioner M so it is important to use an efficient Poisson solver. Fortunately there are techniques, such as the multigrid method, that are optimal for the solution of such problems.

4 Implementation details

The above technique was implemented in a finite element multigrid code developed by Stals [10]. The results presented in this report use a uniformly refined tetrahedron grid, but we are also investigating the use of adaptive grids.

The computational model consists of two stages: building the system of equations given in (8), and solving the system of equations. The CG method as described in Section 3.1 was used to solve the system. The multigrid method was used as the Poisson solver. The time complexity of the first stage is dependent on the number of data points n, whereas the second is dependent on the discretisation size m.

The stopping criterion for the CG method was based on the Hestenes–Stiefel rule as described in [1, 11]. Note that if the preconditioner is not used, then the value for delay length d_e has to be greatly increased, as expected since the delay length depends on the condition number.

In this section we describe in more detail how the system of equations is built, focusing on A and $\boldsymbol{d}$. The row j column k entry of the A matrix is

$$a_{jk} = \frac{1}{n}\sum_{i=1}^{n} b_j(\boldsymbol{x}^{(i)})b_k(\boldsymbol{x}^{(i)})\,,$$

where $\{b_j\}$ are the standard finite element linear basis functions (hat functions). Since b_j has local support

$$a_{jk} = \left\{\frac{1}{n}\sum_{\boldsymbol{x}^{(i)}} b_j(\boldsymbol{x}^{(i)})b_k(\boldsymbol{x}^{(i)}) : \boldsymbol{x}^{(i)} \in \text{ support}(b_j) \cap \text{ support}(b_k)\right\}. \qquad (9)$$

To evaluate Equation (9) the algorithm must determine which tetrahedron in the finite element grid contain which data points. To reduce the search time the domain is divided into equally spaced cubes and a record is kept of which data points sit in which cube. Given a particular tetrahedron, the coordinates of the vertices are used to find the nearby cubes. The final step is to loop through the data points in the nearby cubes and find those sitting within the tetrahedron.

Since the value assigned to a_{jk} is a cumulative sum it is possible to evaluate it in parallel or using out-of-core techniques (as long as the finite element grid remains in core).

A similar technique is used to find $\boldsymbol{d}$.

5 Example 3D applications

We now present three examples of data fitting in 3D. The first example was constructed by randomly generating a million points on a sphere with centre $(0.5, 0.5, 0.5)$ and radius $= 1/3$. In the second example, all of the points on the sphere with $x < 0.5$ were removed. The third example uses the MRI image of a skull that is available in Matlab, see Figure 1.

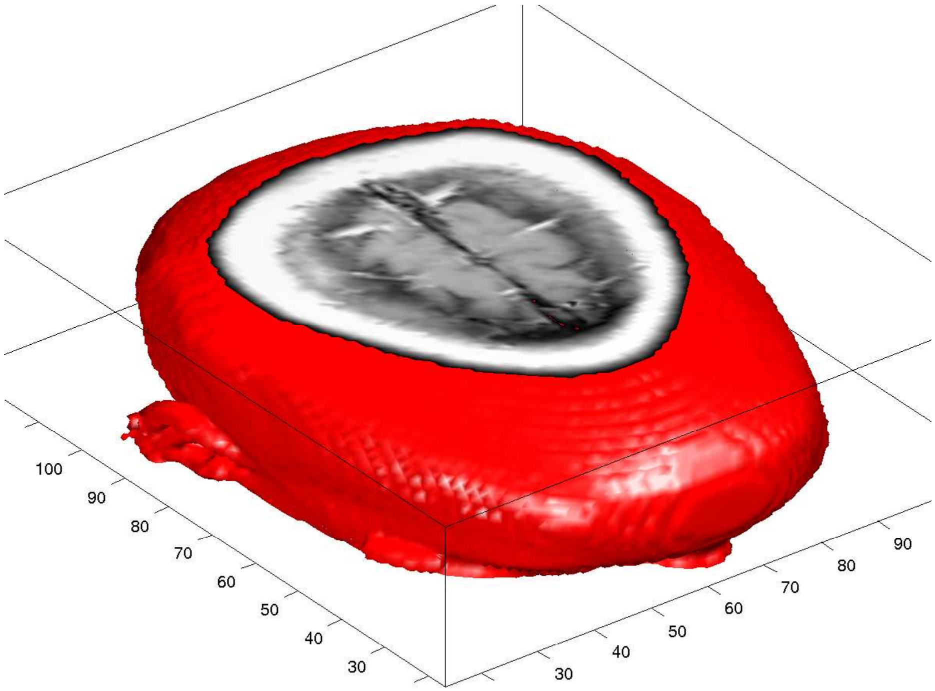

Figure 1: MRI Image of a skull available in Matlab.

The tolerance used for the stopping criterion was 10^{-3}. A fairly high tolerance was chosen for these examples since we were only interested in the visual output.

Figure 2 gives the Iso-surface plot of the results obtained using a grid with 68705 nodes. The data set contained 10^6 points. The smoothing parameter α was set to 10^{-3}. The delay length $d_e = 5$, and the CG method took 7 iterations to converge (including the calculations for the error approximation).

Figure 3 shows a slice of the semi-sphere cut along the $y = 0.5$ axis. Comparing left and right plots in Figure 3 we see that the interpolation is no longer symmetric, as expected. The data set contained approximately

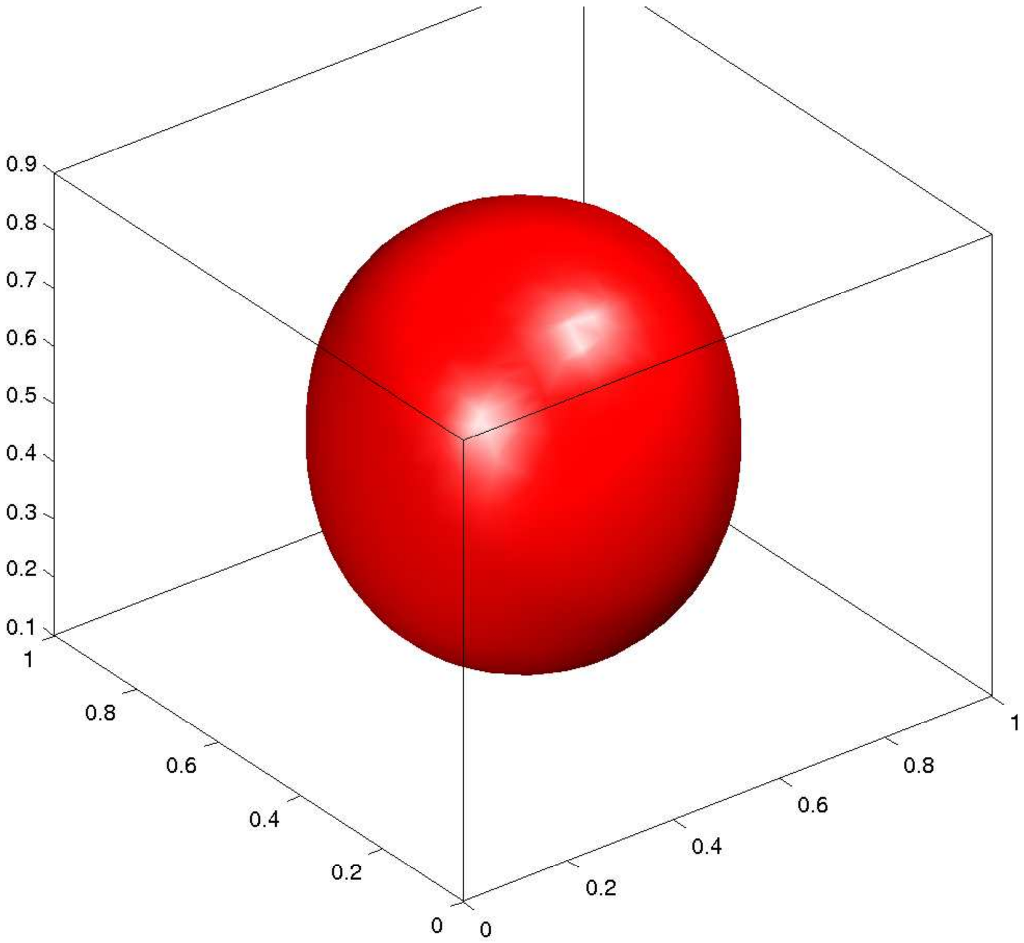

Figure 2: Iso-surface plot of the sphere modelled on a grid with 68705 nodes.

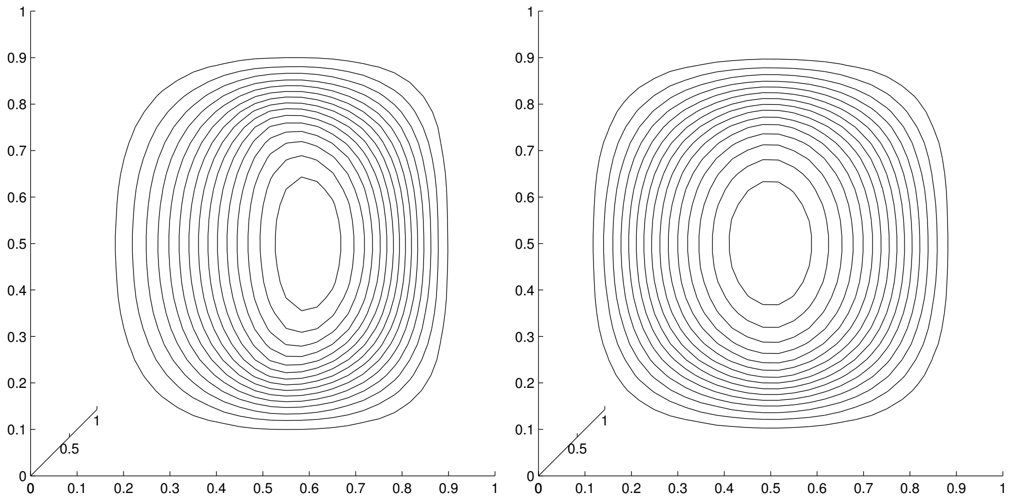

Figure 3: Slice of the semi-sphere along the $y = 0.5$ axis. The finite element grid had 68705 nodes. The left plot shows the semi-sphere, the right plot shows the corresponding plot of the whole sphere.

$1/2 \times 10^6$ data points. The CG method took 5 steps to converge. The smoothing parameter α was set to 10^{-3}.

The final example given in Figure 4 shows the thin plate spline fit to the MRI data given in Figure 1. The smoothing parameter α was reduced to 10^{-7} to better capture the contours of the face and the stopping criterion was 10^{-2}. The delay length $d_e = 20$, and the method took 117 iterations to converge.

A set of Poisson equations must to be solved when multiplying a vector by the Z matrix. This was done by applying the multigrid method until the maximum residual norm was within 10^{-15}. We believed that it was necessary to solve these systems accurately to ensure that the correct matrix is used in the CG iterations. A further set of Poisson equations must be solved when applying the preconditioner. Initially only a small number of multi-

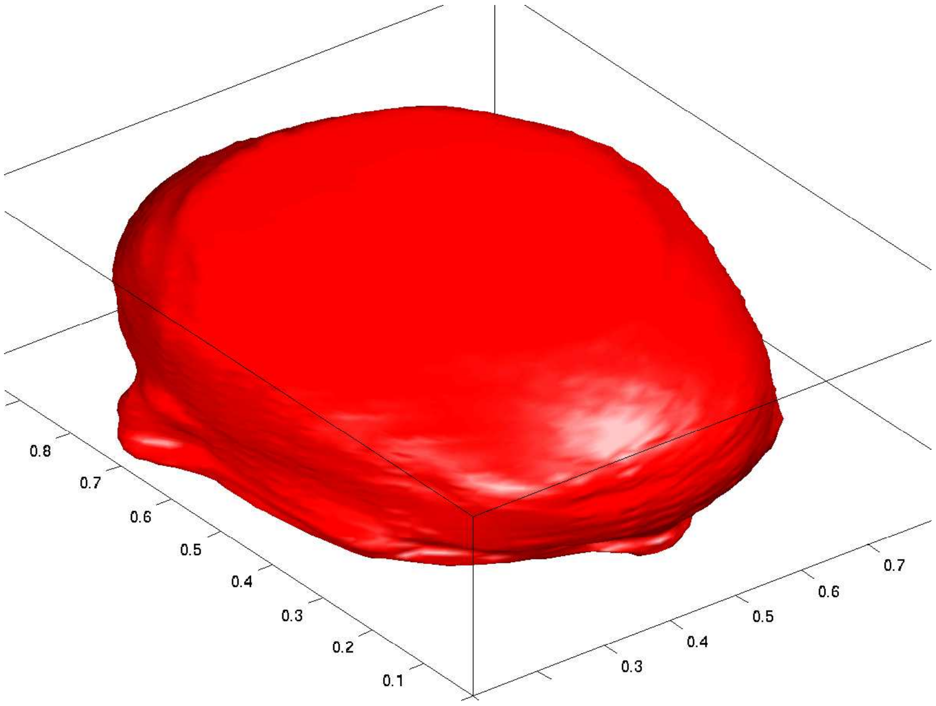

Figure 4: Thin plate spline approximation to skull when the finite element grid contained 68705 nodes.

grid iterations was used, but we observed that the preconditioned matrix was illconditioned and consequently the overall number of CG iterations increased. Therefore the number of multigrid iterations was increased so that the maximum residual norm was less than 10^{-10}. Note that the results from the previous conjugate gradient iteration are used as initial guesses for the Poisson solver.

6 Future research

The work described here is in its early stages. Other options that we wish to explore include; the use of higher-order finite element basis functions, adaptive grid refinement, different solution techniques, automatic calculation of α and different boundary conditions.

References

[1] M. Arioli Backward error analysis and stopping criteria for Krylov space method. presented at: *20th Biennial Conference on Numerical Analysis, University of Dundee* 24–27 June 2003. 41 pages. [Online] http://www.numerical.rl.ac.uk/people/marioli/Dundee-03.pdf. C652

[2] A. Bab-Hadiashar, D. Suter and R. Jarvis R. Optic flow computation using interpolating thin-plate splines, In *Second Asian Conference on Computer Vision ACCV'95*, pages 452–456, Singapore, December 1995. [Online] http://www.ses.swin.edu.au/~ali/page2.html C648

[3] R .K. Beatson, G. Goodsell, and M. J. D. Powell. On multigrid techniques for thin plate spline interpolation in two dimensions. In *The Mathematics of Numerical Analyses*, Lectures in Appl. Math., pages 77–97. 1996. C648

[4] P. Christen, M. Hegland, O. Nielsen, S. Roberts, P. Strazdins and I. Altas. Scalable parallel algorithms for surface fitting and data mining. *Parallel Computing*, 27:941–961, 2001. C649

[5] J. Duchon. Splines minimizing rotation-invariant. In *Lecture Notes in Math*, Vol. 571, pages 85–100. 1977. C647

[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*, pages 245–253. Vanderbilt University Press, 1998. Available as Mathematical Research Report MRR 057-97, School of mathematical Sciences, Australian National University. C649

[7] S. Roberts, M. Hegland, and I. Altas. Approximation of a thin plate spline smoother using continuous piecewise polynomial functions, 2001. *SIAM J. Numer. Anal.*, 41(1):208–234, 2003. [Online] http://epubs.siam.org/sam-bin/dbq/article/38329 C649, C650

[8] K. Rohr, M. Fornefett amd H. S. Stiehl. Approximating thin-plate splines for elastic registration: integration of landmark errors and orientation attributes. In A. Kuba, M. Šámal and A. Todd-Pokropek, editors, *Lecture Notes in Computer Science*, Vol. 1613, pages 252–265. 1999. [Online] http://www.i-u.de/schools/rohr/publ.htm. C648

[9] R. Sibson and G. Stone. Computation of thin plate splines. *SIAM J. Sci. Stat. Comput.*, 12:1304–1313, 1991. C648

[10] L. Stals. *Parallel Multigrid on Unstructured Grids Using Adaptive Finite Element Methods.* PhD thesis, Department Of Mathematics, Australian National University, Australia, 1995. C652

[11] Z. Strakoš and P. Tichý. On error estimation in the conjugate gradient method and why it works in finite precision computations. *Electronic Transaction on Numerical Algebra (ETNA)*, 13:56–80, 2002. C652

[12] G. Wahba. *Spline models for observational data.* CBMS-NSF Regional Conference Series in Applied Mathematics, Vol. 59. SIAM, 1990. C647, C648