Computational strategies for surface fitting using thin plate spline finite element methods

Daryl M. Kempthorne[1] Ian W. Turner[2]
John A. Belward[3]

(Received 2 November 2012; revised 27 March 2013; corrected 2 July 2013)

Abstract

Thin plate spline finite element methods are used to fit a surface to an irregularly scattered dataset. The computational bottleneck for this algorithm is the solution of large, ill-conditioned systems of linear equations at each step of a generalised cross validation algorithm. Preconditioning techniques are investigated to accelerate the convergence of the solution of these systems using Krylov subspace methods. The preconditioners under consideration are block diagonal, block triangular and constraint preconditioners. The effectiveness of each of these preconditioners is examined on a sample dataset taken from a known surface. From our numerical investigation, constraint preconditioners

http://journal.austms.org.au/ojs/index.php/ANZIAMJ/article/view/6337
 Published July 3, 2013, as part of the Proceedings of the 16th Biennial Computational Techniques and Applications Conference. ISSN 1446-8735. (Print two pages per sheet of paper.)

appear to provide improved convergence for this surface fitting problem compared to block preconditioners.
Subject class: 65F08
Keywords: preconditioning, krylov subspace, thin plate splines, surface fitting

Contents

1 Introduction

The thin plate spline finite element method, as proposed by Roberts et al. [9], fits a surface defined by a set of m basis functions on an arbitrary domain Ω to a set of n data points $(x_i, y_i)_{i=1,\dots,n}$. The fitted surface is obtained by minimising a linear combination of the residual of the estimated surface at the data points and a measure of the smoothness of the surface. The

weight of each term is varied by the smoothing parameter, $\alpha > 0$. The inclusion of a smoothing term is to allow a unique surface to be reconstructed from the scanned dataset. In the case of virtualising plant leaves, error is introduced into the data points, which varies with the particular scanning device used to capture the dataset. Due to the presence of measurement error, generalised cross validation (GCV) is used to determine the optimal smoothing parameter [13]. The result of this process is that a number of linear systems of the form $(A + \alpha BB^T)\mathbf{u} = \mathbf{b}$, where A is positive semidefinite and sparse, must be solved for each GCV function evaluation.

The solution of these linear systems is a computational bottleneck for this problem when m is large. Each linear system is of the form of a saddle point problem and preconditioning this problem type is the subject of substantial research [2, 1, 3, 4, 5, 7, 6, 12, e.g.]. Block diagonal, block triangular and constraint preconditioners [2] are investigated to accelerate the convergence of the iterative method applied to the saddle point problem. The linear systems have shifted coefficient matrices, due to the smoothing parameter α, with two different right hand side (RHS) vectors. The efficient solution of a single linear system for a single value of α is the focus of this paper.

The thin plate spline smoother algorithm is outlined in Section 2 and the different preconditioning methods are detailed in Section 3. The computational statistics obtained from the iterative algorithm using these preconditioners is presented in Section 3 for a sample problem based on the `peaks` function in Matlab. Conclusions and recommended future work are outlined in Section 5.

2 Thin plate spline smoother

The thin plate spline smoother uses the analogy that the points lie on a thin metal sheet, which is twisted and bent to fit the data. The quality of the surface is measured in terms of the error between the fitted surface and the known value at the data points, as well as a smoothing term, which

is introduced to control the amount of bending and twisting of the plate. The functional form of this surface on the domain $\Omega \subset \mathbb{R}^2$ is the solution $s(\mathbf{x}) \in H^2(\Omega)$ that minimises the functional

$$\min_{s \in H^2(\Omega)} \bar{J}_\alpha(s, \mathbf{y}) := \|s(\mathbf{x}) - \mathbf{y}\|_n^2 + \alpha |s|^2_{H^2(\Omega)}\,, \tag{1}$$

where

$$\begin{aligned} |s|^2_{H^2(\Omega)} &= \int_\Omega \left[\left(\frac{\partial^2 s}{\partial x_1^2} \right)^2 + 2 \left(\frac{\partial^2 s}{\partial x_1 \partial x_2} \right)^2 + \left(\frac{\partial^2 s}{\partial x_2^2} \right)^2 \right] d\mathbf{x}\,, \\ \langle \mathbf{u}, \mathbf{v} \rangle_n &= n^{-1} \mathbf{u}^T \mathbf{v}\,, \\ \|\mathbf{u}\|_n^2 &= \langle \mathbf{u}, \mathbf{u} \rangle_n\,. \end{aligned}$$

Wahba [13] showed that the optimal value of α depends on the noise in the data and can be determined using GCV.

Roberts et al. [9] reformulated (1) to a $H^1(\Omega)$ minimisation problem by solving for $\mathbf{u}$, defined as $\mathbf{u} := \nabla s$. However, this condition is generally only satisfied in the weak sense $(\nabla s, \nabla v) = (\mathbf{u}, \nabla v)$ for all $v \in H^1(\Omega)$ for arbitrary functions $u_1, u_2 \in H^1(\Omega)$, where $\mathbf{u} = [u_1, u_2]^T$. This formulation is equivalent to the original formulation when the condition $\operatorname{curl}(\mathbf{u}) = \mathbf{0}$ is enforced. Roberts et al. [9] recommend dropping this condition to simplify the solution process.

The Neumann boundary value problem

$$\begin{aligned} \Delta s &= \nabla \cdot \mathbf{u} \quad \text{in } \Omega\,, \\ \nabla s \cdot \mathbf{n} &= \mathbf{u} \cdot \mathbf{n} \quad \text{on } \delta\Omega\,. \end{aligned} \tag{2}$$

is also satisfied by s. The constraint

$$\langle s(\mathbf{x}), \mathbf{e} \rangle_n = \langle \mathbf{y}, \mathbf{e} \rangle_n\,,$$

is imposed to ensure a unique solution of (2), where $\mathbf{e}$ is a vector of all ones. The solution s of (2) is denoted $\Phi(\mathbf{u})$.

The reformulation determines $\mathbf{u}$ such that

$$\mathbf{u}(\mathbf{x}) = \underset{H^1(\Omega)^2}{\arg\min} \|\Phi(\mathbf{u}) - \mathbf{y}\|_n^2 + \alpha \left(|u_1|^2_{H^1(\Omega)} + |u_2|^2_{H^1(\Omega)} \right) , \tag{3}$$

where, for example,

$$|u_1|^2_{H^1(\Omega)} = \int_\Omega \left[\left(\frac{\partial u_1}{\partial x_1} \right)^2 + \left(\frac{\partial u_1}{\partial x_2} \right)^2 \right] d\mathbf{x} .$$

A discretisation of the domain Ω is required, providing a set of m nodes and a triangular mesh. A set of basis functions $\mathbf{h}(\mathbf{x}) \in H^1(\Omega)^m$ is defined to discretise the problem, which gives

$$s(\mathbf{x}) = \mathbf{h}(\mathbf{x})^T \mathbf{c} , \quad u_1(\mathbf{x}) = \mathbf{h}(\mathbf{x})^T \mathbf{g}_1 , \quad u_2(\mathbf{x}) = \mathbf{h}(\mathbf{x})^T \mathbf{g}_2 .$$

The basis functions are chosen as piecewise linear elements, satisfying $h_i(\mathbf{x}_j) = \delta_{ij}$, with δ_{ij} the Kronecker delta. The finite element discretisation of (3) and (2) for fixed α yields the minimisation problem

$$\min_{\mathbf{c},\mathbf{g}_1,\mathbf{g}_2} \left(\left\| \tilde{\mathbf{y}} - H^T \mathbf{c} \right\|_n^2 + \alpha \mathbf{g}_1^T L \mathbf{g}_1 + \alpha \mathbf{g}_2^T L \mathbf{g}_2 \right) ,$$

subject to

$$\mathbf{c} = L^\dagger \left(G_1 \mathbf{g}_1 + G_2 \mathbf{g}_2 \right) ,$$

where $H_{ij} = \mathbf{h}_i(\mathbf{x}_j)$ is a matrix containing the basis functions evaluated at the data points

$$L_{ij} = (\nabla h_i, \nabla h_j)_{L^2(\Omega)} , \quad G_{1_{ij}} = (\partial_{x_1} h_i, h_j)_{L^2(\Omega)} , \quad G_{2_{ij}} = (\partial_{x_2} h_i, h_j)_{L^2(\Omega)} . \tag{4}$$

Operator $L^\dagger$ is a generalised inverse of L satisfying $L^\dagger H \mathbf{e} = \mathbf{0}$ and $\tilde{\mathbf{y}} = \mathbf{y} - \langle \mathbf{y}, \mathbf{e} \rangle_n \mathbf{e}$. The inner product is defined as $(u, v)_{L^2(\Omega)} = \int_\Omega uv \, d\mathbf{x}$. This minimisation problem is equivalent to the equality constrained quadratic programming problem

$$\min_{\mathbf{v}} \mathbf{v}^T A \mathbf{v} - \mathbf{v}^T \mathbf{d} , \quad \text{such that } B\mathbf{v} = \mathbf{0} , \tag{5}$$

where

$$A = \begin{bmatrix} HH^T/n & 0 & 0 \\ 0 & \alpha L & 0 \\ 0 & 0 & \alpha L \end{bmatrix}, \quad \mathbf{d} = \begin{bmatrix} Hy/n \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix},$$
$$B = \begin{bmatrix} I & -L^\dagger G_1 & -L^\dagger G_2 \end{bmatrix}, \quad \mathbf{v} = \begin{bmatrix} \mathbf{c} \\ \mathbf{g}_1 \\ \mathbf{g}_2 \end{bmatrix}. \tag{6}$$

The use of Lagrange multipliers provide an efficient method of obtaining a solution of (5) [14] and results in the need to solve, for a given α, the system of linear equations

$$\mathcal{A}\mathbf{x} = \begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \mathbf{v} \\ \mathbf{w} \end{bmatrix} = \begin{bmatrix} \mathbf{d} \\ \mathbf{0} \end{bmatrix} = \mathbf{b}. \tag{7}$$

Problems of this form are *saddle point problems* [2]. The aim of this work is to investigate efficient solution techniques for systems of linear equations of the form (7).

Benzi et al. [2] reviewed approaches for efficiently solving saddle point problems, focusing on large and sparse linear systems. They remarked that classical methods for solving saddle point problems include null space methods, which were originally used by Roberts et al. [9]. They, amongst others [1, 3, 4, 5, 7, 6, 12], also discussed preconditioned Krylov subspace methods and concluded that effective preconditioners are under development for many classes of linear systems [2, p. 108].

The solution of these linear systems is the primary computational bottleneck for the algorithm of these typically large linear systems. In such cases, direct solution techniques are unable to obtain a solution in reasonable time, necessitating the use of iterative methods. Preconditioning approaches are investigated in Section 3 to accelerate the convergence of these iterative methods.

3 Solution approaches

Efficient solution methods for saddle point problems have been the subject of substantial research due to their regular occurrence in a variety of problems [2, 1, 3, 4, 5, 6, 7]. Benzi et al. [2] provided an overview of ideal preconditioners for saddle point problems. Two types of preconditioners are investigated here, namely block preconditioners and constraint preconditioners.

3.1 Block preconditioners

Two different forms of block preconditioners are under consideration, block diagonal $\mathcal{P}_D$, and block triangular $\mathcal{P}_T$, which are

$$\mathcal{P}_D^{-1} = \begin{bmatrix} A^{-1} & 0 \\ 0 & -S^{-1} \end{bmatrix}, \quad \mathcal{P}_T^{-1} = \begin{bmatrix} A^{-1} & 0 \\ -S^{-1}BA^{-1} & S^{-1} \end{bmatrix}, \tag{8}$$

respectively. Here, S is the Schur complement, $S = -BA^{-1}B$. Both these block preconditioners are applied on the left of the linear system.

De Sturler and Liesen [12] gave the complete eigendecomposition of $\mathcal{P}_D^{-1}\mathcal{A}$. Interestingly, the eigenvalues cluster around the three points 1, $\frac{1}{2}(1+\sqrt{5})$ and $\frac{1}{2}(1-\sqrt{5})$ [7, e.g.]. Benzi et al. [2] showed that $\mathcal{P}_T^{-1}\mathcal{A}$ has 1 as its distinct eigenvalue.

Both block preconditioners require that A is nonsingular. However, our problem has nullity$(A) \geqslant 2$, indicating that the matrix is singular. One common approach taken to overcome the singular nature is to use an augmented Lagrangian formulation [5], which replaces A with $A_W = A + B^T W B$, where W is a symmetric positive semidefinite matrix. Greif et al. [4] analysed the choice $W = \gamma I$ and stated that the choice $\gamma = \|A\|_2/\|B\|_2^2$ is shown experimentally to be effective. The presence of $L^\dagger$ in B causes A_W to be predominantly dense, thus losing the block diagonal structure of A. Greif et al. [4] stated that, in practise, an approximation to A_W is often used, which will not be considered in this paper.

3.2 Constraint preconditioners

The second type of preconditioning strategy used is a constraint preconditioner [2]. This type of preconditioner takes the form

$$\mathcal{P}_C = \begin{bmatrix} Z & B^T \\ B & 0 \end{bmatrix},$$

which is the same as $\mathcal{A}$ with the $(1,1)$ block modified. Generally, Z is chosen implicitly based on the Schilders factorisation [1], namely

$$\mathcal{P}_C = \begin{bmatrix} B_1^T & 0 & M_1 \\ B_2^T & M_2 & E \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} D_1 & 0 & I \\ 0 & D_2 & 0 \\ I & 0 & 0 \end{bmatrix} \begin{bmatrix} B_1 & B_2 & 0 \\ 0 & M_2^T & 0 \\ M_1^T & E^T & I \end{bmatrix}. \tag{9}$$

Choosing the components of this factorisation to match

$$\mathcal{A} = \begin{bmatrix} A_{11} & A_{12} & B_1^T \\ A_{21} & A_{22} & B_2^T \\ B_1 & B_2 & 0 \end{bmatrix},$$

and noting that $A_{11} = HH^T/n$, $A_{12} = A_{21}^T = 0$, $A_{22} = \operatorname{diag}(\alpha L, \alpha L)$, $B_1 = I$ and $B_2 = [-L^\dagger G_1, -L^\dagger G_2]$, gives

$$\begin{aligned} D_1 &= A_{11} - M_1^T - M_1, \\ D_2 &= M_2^{-1}\left(A_{22} + B_2^T A_{11} B_2\right) M_2^{-T}, \\ E &= -B_2^T\left(A_{11} - M_1\right), \end{aligned} \tag{10}$$

where M_1 is any matrix and M_2 is any nonsingular matrix. Benzi and Wathen [1] claimed that any choice of D_1, E and M_1, and any nonsingular choice of D_2 and M_2, provide a suitable preconditioner. Furthermore, the conjugate gradient algorithm (CG) [10] is applicable because this preconditioner, in conjunction with the linear system (7), results in the elimination of the constraints [1, p. 203].

4 Results

The sample function used to assess the effectiveness of the preconditioners was the `peaks` function of Matlab. The function was sampled at 10 000 random points uniformly distributed on the unit square. The nodes were chosen to be on each corner and at 496 randomly distributed points in the interior of the unit square (500 nodes in total). The resulting linear system had dimension 2 000 and `GMRES` [11] was used to solve the linear system with the block preconditioners because the preconditioned system is not symmetric. The Hestenes–Stiefel conjugate gradient method was used with the constraint preconditioner. The two values chosen for the smoothing parameter α are 10^{-10} and 10^{-2}. The condition number for the coefficient matrices in (7) for these choices of α are approximately 5×10^{10} and 7×10^{5}, respectively. The desired convergence tolerance is $1 \times 10^{-8} \|\mathsf{b}\|$, with $\|\mathsf{b}\| \approx 122.866$.

4.1 Block preconditioners

The results of the block preconditioners are compared using the exactly formed $\mathcal{P}_D$ and $\mathcal{P}_T$ in (8) using the augmented Lagrangian formulation in Matlab. The choice $W = \gamma I$ is made, with $\gamma = \|A\|_2 / \|B\|_2^2$. The effect of using these preconditioners is shown in Table 1.

Using the exact form of the preconditioners results in extremely rapid convergence, as expected by the eigenvalue decomposition of the preconditioned matrices [12]. However, the use of the exact preconditioners is impractical for solutions of large linear systems, due to the time required to construct the matrices and the memory required to store them. By means of comparison, applying $\mathcal{P}_D$ and $\mathcal{P}_T$ through solving a linear system $\mathsf{P}\mathbf{x} = \mathbf{z}$ with a Krylov subspace solver, for some $\mathbf{z}$, produces extraordinary computation times. This is primarily due to solving linear systems with S as the coefficient matrix, whereby multiplying a vector by S requires the inversion of A_W. This will produce unreasonable times for even moderately sized linear systems. For

Table 1: Comparison of the preconditioners using the augmented Lagrangian formulation, using exact forms of the block preconditioners.

Preconditioner	α	Iterations	Residual norm	Time (s)
None	10^{-10}	1390	4.8×10^{-8}	46.9
	10^{-2}	537	1.3×10^{-9}	13.4
Triangular	10^{-10}	4	8.8×10^{-8}	2.7
	10^{-2}	3	2.6×10^{-8}	2.3
Diagonal	10^{-10}	6	1.1×10^{-4}	3.0
	10^{-2}	3	3.0×10^{-8}	2.2

fitting surfaces with a small number of nodes using thin plate splines, this finite element method is highly applicable if the matrices are explicitly formed and stored in memory. In situations where this is not possible, approximation methods for the block preconditioners must be utilised to achieve any improvement in solving the linear system.

4.2 Constraint preconditioners

The constraint preconditioner used is the Schilders factorisation (9) with elements described in (10). The matrices M_1 and M_2 are both chosen as the identity matrix. With these choices, to apply this preconditioner only linear systems of the form $D_2\mathbf{x} = \mathbf{b}$ need be solved. This linear system was solved inexactly using `MINRES` [8] with the convergence tolerance τ varied to determine the effect that the solution of this linear system has on the overall performance of the preconditioner.

Tables 2 and 3 show that the convergence tolerance used to solve the inner linear system within the preconditioner has an impact on the overall performance of the iterative method. The use of a tolerance larger than $\tau > 1 \times 10^{-5}$ caused the preconditioner to stagnate, as opposed to converging

Table 2: Effect of the iterative scheme for the $D_2\mathbf{x} = \mathbf{b}$ linear system with $\alpha = 10^{-2}$ for varying tolerance τ.

Tolerance	Iterations	Residual Norm	Termination	Time (s)
1×10^{-10}	2	7.7×10^{-8}	Converged	23.3
1×10^{-9}	2	9.9×10^{-8}	Converged	22.1
1×10^{-8}	2	1.3×10^{-7}	Converged	21.6
1×10^{-7}	2	9.2×10^{-8}	Converged	20.7
1×10^{-6}	2	1.1×10^{-6}	Converged	19.6
5×10^{-6}	3	4.7×10^{-6}	Converged	20.6
1×10^{-5}	3	8.5×10^{-6}	Converged	20.7
1×10^{-5}	—	5.4×10^{-5}	Stagnation	75.1

Table 3: Effect of the iterative scheme for the $D_2\mathbf{x} = \mathbf{b}$ linear system with $\alpha = 10^{-10}$ for varying tolerance τ.

Tolerance	Iterations	Residual Norm	Termination	Time (s)
1×10^{-10}	5	5.3×10^{-6}	Converged	58.4
1×10^{-9}	5	6.7×10^{-6}	Converged	56.4
1×10^{-8}	5	3.7×10^{-6}	Converged	54.5
1×10^{-7}	6	2.2×10^{-6}	Converged	53.1
1×10^{-6}	7	1.3×10^{-6}	Converged	50.9
5×10^{-6}	17	9.6×10^{-6}	Converged	88.2
1×10^{-5}	40	9.0×10^{-6}	Converged	138.5
5×10^{-5}	—	4.3×10^{-3}	Stagnation	125.2

Table 4: Summary statistics for 150 test datasets to assess the effect of the inner linear system convergence tolerance on the overall iterative scheme.

Tolerance τ	Mean wall clock time (s)	Average inner iterations $\alpha = 1 \times 10^{-2}$	Average inner iterations $\alpha = 1 \times 10^{-10}$
1×10^{-10}	40.7	315	506
1×10^{-9}	40.0	302	484
1×10^{-8}	37.1	289	454
1×10^{-7}	34.3	274	398
1×10^{-6}	34.1	250	262
5×10^{-6}	56.4	174	140
1×10^{-5}	81.2	130	93
5×10^{-5}	100.8	43	63

to the solution of the linear system. This sudden change in convergence may be due to the preconditioner no longer exactly satisfying the constraint conditions, thus causing the preconditioned system to not be symmetric and positive definite, resulting in the failure of `CG`. On the other hand, the use of a tolerance too small causes over-solving of the inner linear system, achieving no further reduction in the number of outer iterations required to converge to the solution.

In order to determine an appropriate tolerance τ, the effect of the individual dataset must be removed. Table 4 shows summary statistics for 150 sample datasets, generated using the method described at the beginning of Section 4. Analysis of variance shows that there is a statistical difference between the mean wall clock times averaged over α for different tolerance levels ($F = 1888$, $d.f. = 7$). However, there is no statistical difference between the mean wall clock times for $\tau = 1 \times 10^{-6}$ and $\tau = 1 \times 10^{-7}$ averaged over α. In light of $\tau = 1 \times 10^{-6}$ requiring fewer inner iterations (on average) than $\tau = 1 \times 10^{-7}$, this value for the inner convergence tolerance provides the best trade off between solving the inner linear system exactly and the total time taken.

5 Conclusion

Block preconditioners and constraint preconditioners were investigated to determine their effectiveness for accelerating convergence to the solution of linear systems in the evaluation of the GCV function from the thin plate spline smoother. The use of block preconditioners is impractical because the approximations made to reduce computational requirements cause these preconditioners to be ineffective, also, the loss of symmetry of the coefficient matrix requires `GMRES` to be utilised. The use of a constraint preconditioner accelerated the convergence to the solution of the linear system by the conjugate gradient method. The effectiveness of the constraint preconditioner is also determined by solving the inner linear system $\mathsf{D}_2\mathbf{x} = \mathbf{b}$, with tolerance 1×10^{-6} being most efficient for the size of problems studied here.

Future work will involve the use of block conjugate gradient methods to solve multiple linear systems with the same coefficient matrices at the same time and preconditioning methods for alternative choices of M_2 to improve the rate of convergence of the inner linear system.

Acknowledgements We acknowledge support from the Australian Research Council through the Linkage Project LP100200476 and its industry partners Syngenta, Dow AgroSciences, Croplands/NuFarm, Plant Protection Chemistry NZ Ltd. and Bill Gordon Enterprises.

We thank Professor Valeria Simoncini for many helpful discussions on the efficient solution of saddle point problems. We also thank the anonymous reviewers for their thorough reading of the manuscript and pertinent comments.

References

[1] M. Benzi and A. J. Wathen. Some preconditioning techniques for saddle point problems. In *Model Order Reduction: Theory, Research Aspects and Applications*, volume 13 of *Mathematics in Industry*, pages 195–211. Springer Berlin Heidelberg, 2008. doi:10.1007/978-3-540-78841-6_10. C58, C61, C62, C63

[2] Michele Benzi, Gene H. Golub, and J. Liesen. Numerical solution of saddle point problems. *Acta Numer.*, 14:1–137, 2005. doi:10.1017/S0962492904000212. C58, C61, C62, C63

[3] H. S. Dollar, N. I. M. Gould, M. Stoll, and A. J. Wathen. Preconditioning Saddle-point Systems with Applications in Optimisation. *SIAM J. Sci. Comput.*, 31(1):249–270, 2010. doi:10.1137/080727129. C58, C61, C62

[4] C. Greif, G. Golub, and J. Varah. Augmented Lagrangian Techniques for Solving Saddle Point Linear Systems. *SIAM J. Matrix Anal. Appl.*, 2004. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.10.942. C58, C61, C62

[5] C. Greif and D. Schötzau. Preconditioners for saddle point linear systems with highly singular (1,1) blocks. *Electron. Trans. Numer. Anal.*, 22:114–121, 2006. http://www.emis.ams.org/journals/ETNA/vol.22.2006/pp114-121.dir/pp114-121.pdf. C58, C61, C62

[6] Sabine Le Borne and Che Ngufor. An implicit approximate inverse preconditioner for saddle point problems. *Electron. Trans. Numer. Anal.*, 37:173–188, 2012. http://www.emis.ams.org/journals/ETNA/vol.37.2010/pp173-188.dir/pp173-188.pdf. C58, C61, C62

[7] Malcolm F. Murphy, Gene H. Golub, and Andrew J. Wathen. A note on preconditioning for indefinite linear systems. *SIAM J. Sci. Comput.*,

21(6):1969–1972, December 1999. doi:10.1137/S1064827599355153. C58, C61, C62

[8] C. C. Paige and M. A. Saunders. Solution of Sparse Indefinite Systems of Linear Equations. *SIAM J. Numer. Anal.*, 12 (4):617–629, 1975. doi:10.1137/0712047. C65

[9] S. Roberts, M. Hegland, and I. Altas. Approximation of a Thin Plate Spline Smoother using Continuous Piecewise Polynomial Functions. *SIAM J. Numer. Anal.*, 1:208–234, 2003. doi:10.1137/S0036142901383296. C57, C59, C61

[10] Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, 2nd edition, 2003. C63

[11] Youcef Saad and Martin H Schultz. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7(3):856–869, July 1986. doi:10.1137/0907058. C64

[12] Eric de Sturler and Jörg Liesen. Block-diagonal and constraint preconditioners for nonsymmetric indefinite linear systems. part i: Theory. *SIAM J. Sci. Comput.*, 26(5):1598–1619, May 2005. doi:10.1137/S1064827502411006. C58, C61, C62, C64

[13] G. Wahba. *Spline Models for Observational Data*. SIAM, 1990. C58, C59

[14] W. L. Winston. *Operations Research: Applications and Algorithms*. Thompson Brooks/Cole, 4th edition, 2004. C61

Author addresses

1. **Daryl M. Kempthorne**, Queensland University of Technology
   mailto:d.kempthorne@student.qut.edu.au
2. **Ian W. Turner**, Queensland University of Technology

mailto:i.turner@qut.edu.au

3. **John A. Belward**, Queensland University of Technology
mailto:j.belward@qut.edu.au