Journal of Computational Physics 228 (2009) 5592–5619

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

A singularity-avoiding moving least squares scheme for two-dimensional unstructured meshes

Samuel K.M. Chenoweth [a,*], Julio Soria [b], Andrew Ooi [a]

[a] Department of Mechanical Engineering, University of Melbourne, Victoria 3010, Australia
[b] Laboratory for Turbulence Research in Aerospace and Combustion, Department of Mechanical Engineering, Monash University, Victoria 3800, Australia

ARTICLE INFO

Article history:
Received 4 August 2008
Received in revised form 23 April 2009
Accepted 23 April 2009
Available online 7 May 2009

PACS:
02.60.Ed
02.60.Lj
47.10.ad
92.60.hk

Keywords:
Moving least squares
Interpolation
Singularities
Convection
Diffusion
Navier–Stokes

ABSTRACT

Moving least squares interpolation schemes are in widespread use as a tool for numerical analysis on scattered data. In particular, they are often employed when solving partial differential equations on unstructured meshes, which are typically needed when the geometry defining the domain is complex. It is known that such schemes can be singular if the data points in the stencil happen to be in certain special geometric arrangements, however little research has specifically addressed this issue. In this paper, a moving least squares scheme is presented which is an appropriate tool for use when solving partial differential equations in two dimensions, and the precise conditions under which singularities occur are identified. The theory is used to develop a stencil building algorithm which automatically detects singular stencils and corrects them in an efficient manner, while attempting to maintain stencil symmetry as closely as possible. Finally, the scheme is applied in a convection–diffusion equation solver and an incompressible Navier–Stokes solver, and the results are shown to compare favourably with known analytical solutions and previously published results.

1. Introduction

When conducting numerical simulations that operate on data defined on an unstructured mesh, it is necessary to employ interpolation techniques in order to evaluate the quantities at arbitrary points in the domain. Calculation of the spatial derivatives of a quantity, even at a point where the value of that quantity has been specified, also require the use of such methods. A variety of techniques for interpolation of a scalar function based on values at scattered points have been described in the literature, such as the Shepard method [17], moving least squares [12], radial basis functions [8] and an unstructured version of bilinear interpolation [11]. In addition, some of these methods have been extended to allow direct calculation of spatial derivatives using radial basis functions [19], moving least squares [10] and unstructured bilinear interpolation [11].

The moving least squares scheme allows a function and its spatial derivatives to be approximated at a point (here termed the interpolation point) based on scattered values of the function (or its spatial derivative in some direction) at a number of other points (here termed the data points). At each location in the domain where an interpolation is required (e.g. at each edge midpoint), a stencil of local data points is selected and it is this set of data points which is used to approximate the value at the interpolation point. The justification for basing the approximation only on local data points, as opposed to

* Corresponding author. Tel.: +61 3 8344 8102.
E-mail address: s.chenoweth@pgrad.unimelb.edu.au (S.K.M. Chenoweth).

0021-9991/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2009.04.036

the complete set of data points in the domain, is that the value of the function is likely to be more strongly influenced by close data points than ones further away.

The moving least squares method comes in two main varieties, interpolating moving least squares and standard (or approximating) moving least squares. For standard moving least squares, a surface formed from some basis function is fitted to the data points so that the sum of the squares of the errors at each data point is minimised (see Section 2). (A weighting function is also frequently used so that errors are considered to be less significant if they occur on data points located further from the interpolation point.) It is understood and accepted that the surface obtained will not reproduce the values at the data points exactly, although the errors are likely to be minimal if the data is smooth. Interpolating moving least squares, on the other hand, attempts to form a surface using a weighting function which is inherently singular at the data points. The values at the data points are then reproduced exactly, but smoothness can be an issue [13]. Thus the standard moving least squares scheme is properly called an approximation rather than interpolation scheme. The interpolating moving least squares scheme, on the other hand, is truly interpolating. This paper focusses on the standard moving least squares scheme (henceforward referred to as simply moving least squares), as its smoothness properties make it generally useful for solving partial differential equations not possessing discontinuities (such as shocks) in their solutions. Since the values at the data points are reproduced with a high degree of accuracy when using standard moving least squares to approximate smooth data, the term 'interpolation' will be used henceforward when referring to this operation.

The (standard) moving least squares scheme has been widely used for both pure interpolation/ approximation problems (e.g. in image processing applications [13]) and in solving partial differential equations [3]. It has been widely reported in the literature that the moving least squares scheme can suffer from singularities, where, in certain situations, the matrix requiring inversion in the solution procedure is not invertible. Desimone et al. [3], for instance, identify the singularity problem and note that it tends to occur when the number of data points supporting each weight function is "not sufficiently greater than" the number of linearly independent functions making up their basis function (which is the theoretical minimum required). However, the meaning of "sufficiently" is not made precise, and their solution for avoiding the problem is acknowledged to be computationally expensive, and numerically poor if the number of data points included is not "considerably" greater than the theoretical minimum. A useful clue to the cause of singularities is provided by the work of Bodin et al. [2], who note that singularities occur when the data points are arranged in a "degenerate pattern", citing an arrangement along a straight line as an example. However, no further discussion is provided of the general conditions under which singularities occur. The singularity problem for moving least squares has also been noted by Netuzhylov [14] and Prax et al. [15], and for a similar problem by Schoenauer and Adolph [16], who also observed the occurrence of singularities when the data points lie along straight lines. However, to the best knowledge of the author, a general theory for predicting the occurrence of singularities in moving least squares has never been published. This problem is addressed in this paper.

Moreover, the lack of a precise understanding of the cause of singularities has meant that the strategies proposed to overcome them are often vague and uncertain, such as adding extra data points to the stencil (without any rigourous theory predicting which data points will be the best to add or how many new data points may be required) or, in the case of meshless methods, randomly moving each data point a small distance [15]. Using the methodology presented in this paper, it is possible to devise more reliable algorithms for detecting singular stencils and correcting them efficiently.

2. Overview of moving least squares

The interpolation scheme presented in this section is suitable for use in finite volume solvers of two-dimensional partial differential equations on unstructured meshes. (The extension of these methods to three dimensions is straightforward, but is not considered here.) This scheme is intended to allow the calculation of a function f and its spatial derivatives based upon the values and/or directional derivatives of that function at various scattered points in the local region. This much flexibility is necessary for handling the kinds of boundary conditions that are typically required when solving partial differential equations. For instance, when solving the heat equation, the boundary conditions may be specified in terms of the temperature (for a boundary with controlled temperature), the spatial derivative of the temperature in a direction normal to the boundary (for a boundary with controlled heat flux) or a linear combination of the two (for a boundary exposed to forced convection). In order to allow for such general boundary conditions, the following linear relationship is specified for each data point j:

$$a_j f(x_j, y_j) + L_j n_{x,j} \left.\frac{\partial f}{\partial x}\right|_{(x_j, y_j)} + L_j n_{y,j} \left.\frac{\partial f}{\partial y}\right|_{(x_j, y_j)} = C_j, \tag{1}$$

where x and y are the dimensional spatial coordinates, a_j is a dimensionless constant that is equal to 1 if the value of the function is involved in the specification or 0 otherwise, L_j is a length scale constant, $n_{x,j}$ and $n_{y,j}$ are the dimensionless components of the unit vector in which the directional derivative is to be specified and C_j is a prescribed (possibly time varying) value. The factor L_j is necessary to ensure that all the terms in this expression have the same units.

In order to avoid numerical issues, such as disparities in magnitude between terms of different order in the basis function, it is desirable for the interpolations to be carried out in non-dimensional space. To this end, it is necessary to define the local scale of the mesh at each edge i and use this local scale for non-dimensionalising and re-dimensionalising all length based quantities involved in interpolations at this edge. A convenient definition for the local mesh scale Δ_i near edge i is

$$\Delta_i = \begin{cases} \sqrt{A_{i,1}} & \text{when edge } i \text{ is on a boundary,} \\ \sqrt{\sqrt{A_{i,1}}\sqrt{A_{i,2}}} & \text{otherwise,} \end{cases} \tag{2}$$

where $A_{i,1}$ and $A_{i,2}$ are the areas of the element(s) adjacent to edge i. Because this length parameter is based on area, the same length scale will apply to both the long and short sides of an element. The spatial coordinates used for doing the interpolation at the midpoint of edge i are non-dimensionalised using an origin (x_c, y_c), also located at the midpoint of edge i, giving

$$\hat{x} = \frac{x - x_c}{\Delta_i} \tag{3}$$

and

$$\hat{y} = \frac{y - y_c}{\Delta_i}. \tag{4}$$

The spatial derivatives of f can be expressed in terms of non-dimensional spatial coordinates as

$$\frac{\partial f}{\partial \hat{x}} = \Delta_i \frac{\partial f}{\partial x} \tag{5}$$

and

$$\frac{\partial f}{\partial \hat{y}} = \Delta_i \frac{\partial f}{\partial y}. \tag{6}$$

The length parameter used in (1) can be non-dimensionalised thus, for each data point j which is used in the stencil for interpolating on edge i

$$\widehat{L}_j = \frac{L_j}{\Delta_i}. \tag{7}$$

The data point specification (1) can then be expressed in terms of non-dimensional spatial variables as

$$a_j f(x_j, y_j) + \widehat{L}_j n_{x,j} \left.\frac{\partial f}{\partial \hat{x}}\right|_{(x_j,y_j)} + \widehat{L}_j n_{y,j} \left.\frac{\partial f}{\partial \hat{y}}\right|_{(x_j,y_j)} = C_j. \tag{8}$$

The moving least squares scheme requires that a basis function be chosen, which will be fitted to the data points of each stencil. One of the more common choices for a basis function is an nth order polynomial, which can be expressed in non-dimensional vector form as

$$f^*(\hat{x}, \hat{y}) = \mathbf{b}(\hat{x}, \hat{y})\mathbf{p}, \tag{9}$$

where $\mathbf{p} = \left(p_1, p_2, \ldots, p_{\frac{1}{2}(n+1)(n+2)-1}, p_{\frac{1}{2}(n+1)(n+2)}\right)^T$, the column vector of basis polynomial coefficients (T indicates the transpose), and $\mathbf{b}(\hat{x}, \hat{y}) = \left(1, \hat{x}, \hat{y}, \hat{x}^2, \hat{x}\hat{y}, \hat{y}^2, \ldots, \hat{x}^n, \hat{x}^{n-1}\hat{y}, \ldots, \hat{x}\hat{y}^{n-1}, \hat{y}^n\right)$, the row vector of basis polynomial terms.

The partial derivatives of the polynomial basis function can be expressed in vector form as

$$f^*_{\hat{x}}(\hat{x}, \hat{y}) = \mathbf{b}_{\hat{x}}(\hat{x}, \hat{y})\mathbf{p}, \tag{10}$$

$$f^*_{\hat{y}}(\hat{x}, \hat{y}) = \mathbf{b}_{\hat{y}}(\hat{x}, \hat{y})\mathbf{p}, \tag{11}$$

where $\mathbf{b}_{\hat{x}}(\hat{x}, \hat{y}) = \left(0, 1, 0, 2\hat{x}, \hat{y}, 0, \cdots n\hat{x}^{n-1}, (n-1)\hat{x}^{n-2}\hat{y}, \ldots, 2\hat{x}\hat{y}^{n-2}, \hat{y}^{n-1}, 0\right)$ and $\mathbf{b}_{\hat{y}}(\hat{x}, \hat{y}) = \left(0, 0, 1, 0, \hat{x}, 2\hat{y}, \ldots, 0, \hat{x}^{n-1}, 2\hat{x}^{n-2}\ \hat{y}, \ldots, (n-1)\hat{x}\hat{y}^{n-2}, n\hat{y}^{n-1}\right)$.

The constraint which is specified for each data point j is given by (8). In order to fit an appropriate basis polynomial, therefore, it is necessary to compare the values of C_j with values predicted by the basis polynomial, C_j^*. To this end an expression for C_j^* in terms of the basis polynomial must be obtained, which is

$$C_j^* = a_j f^*(\hat{x}_j, \hat{y}_j) + \widehat{L}_j n_{x,j} f^*_{\hat{x}}(\hat{x}_j, \hat{y}_j) + \widehat{L}_j n_{y,j} f^*_{\hat{y}}(\hat{x}_j, \hat{y}_j) = (a_j \mathbf{b}(\hat{x}_j, \hat{y}_j) + \widehat{L}_j n_{x,j} \mathbf{b}_{\hat{x}}(\hat{x}_j, \hat{y}_j) + \widehat{L}_j n_{y,j} \mathbf{b}_{\hat{y}}(\hat{x}_j, \hat{y}_j))\mathbf{p}. \tag{12}$$

Finally, a column vector $\mathbf{C}^*$ containing the predicted C_j^* values for all the m data points can be expressed as

$$\mathbf{C}^* = \mathbf{B}\mathbf{p}, \tag{13}$$

where $\mathbf{B}$ is an m by $\frac{1}{2}(n+1)(n+2)$ matrix, defined by

$$\mathbf{B} = \begin{bmatrix} a_1 \mathbf{b}(\hat{x}_1, \hat{y}_1) + \widehat{L}_1 n_{x,1} \mathbf{b}_{\hat{x}}(\hat{x}_1, \hat{y}_1) + \widehat{L}_1 n_{y,1} \mathbf{b}_{\hat{y}}(\hat{x}_1, \hat{y}_1) \\ a_2 \mathbf{b}(\hat{x}_2, \hat{y}_2) + \widehat{L}_2 n_{x,2} \mathbf{b}_{\hat{x}}(\hat{x}_2, \hat{y}_2) + \widehat{L}_2 n_{y,2} \mathbf{b}_{\hat{y}}(\hat{x}_2, \hat{y}_2) \\ \vdots \\ a_m \mathbf{b}(\hat{x}_m, \hat{y}_m) + \widehat{L}_m n_{x,m} \mathbf{b}_{\hat{x}}(\hat{x}_m, \hat{y}_m) + \widehat{L}_m n_{y,m} \mathbf{b}_{\hat{y}}(\hat{x}_m, \hat{y}_m) \end{bmatrix}. \tag{14}$$

The goal of the interpolation scheme, then, is to select $\mathbf{p}$ such that $\mathbf{C}^*$ is the best possible approximation to $\mathbf{C}$, where $\mathbf{C} = (C_1, C_2, \ldots, C_m)^T$.

One of the simplest ways to find a basis function which is a good approximation to the given data is to select $\mathbf{p}$ such that $\sum_{j=1}^{m} \left(C_j^* - C_j\right)^2$ is minimised. However, this will mean that errors in the fit will be equally weighted for all data points in the

stencil, regardless of their relative distance from the interpolation point. Intuitively, it would make more sense if distant points were weighted less. For this reason, it is useful to define a weight function w_j associated with each data point in the stencil, which will depend on the spatial relationship between data point j and the interpolation point (x_c, y_c). The weight function is applied to the least squares expression, so that the aim is to select $\mathbf{p}$ such that $\sum_{j=1}^{m}\left[w_j\left(C_j^* - C_j\right)\right]^2$ is minimised. A weight function should be a monotonically decreasing function of the (normalised) radial distance between the data point and the interpolation point, $\hat{r}_j = \sqrt{\hat{x}_j^2 + \hat{y}_j^2}$. Singularities at $\hat{r}_j = 0$ are to be avoided, and the weight function must be non-zero at all stencil data points, since a zero weight at a data point is effectively non-inclusion of the data point in the stencil. One suitable weight function is the Gaussian function, defined by

$$w_j = e^{-\frac{\hat{r}_j^2}{\varepsilon^2}}, \tag{15}$$

where ε is a dimensionless shape parameter. This was the weight function used for the simulations in this paper. In general, a small value of ε results in a sharp penalty surface, i.e. a weight function which penalises errors close to the interpolation point much more heavily than errors further away. A large value results in a flatter penalty surface. From numerical experiments based on the accuracy in reproducing test functions (see Section 4.1), it was found that $\varepsilon = 1.4$ was an appropriate choice; this value was used for all the simulations in this paper. Note that ε is a constant, and so the same weight function is used for all stencils and data points.

Recall that the aim of the moving least squares method (with a weight function) is to minimise $\sum_{j=1}^{m}\left[w_j\left(C_j^* - C_j\right)\right]^2$. This is equivalent to minimising $\sum_{j=1}^{m}\left(w_jC_j^* - w_jC_j\right)^2$. Let a weight matrix $\mathbf{w}$ be defined by

$$\mathbf{w} = \begin{bmatrix} w_1 & 0 & 0 & \cdots & 0 \\ 0 & w_2 & 0 & \cdots & 0 \\ 0 & 0 & w_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & w_m \end{bmatrix}, \tag{16}$$

so that

$$\begin{Bmatrix} w_1C_1^* - w_1C_1 \\ w_2C_2^* - w_2C_2 \\ \vdots \\ w_mC_m^* - w_mC_m \end{Bmatrix} = \mathbf{wC}^* - \mathbf{wC}. \tag{17}$$

Therefore $\sum_{j=1}^{m}\left(w_jC_j^* - w_jC_j\right)^2$ can be minimised by minimising $|\mathbf{wC}^* - \mathbf{wC}|$. From (13), this is the same as minimising $|(\mathbf{wB})\mathbf{p} - (\mathbf{wC})|$.

According to linear algebra theory (see [7, pp. 417–426]), $|(\mathbf{wB})\mathbf{p} - (\mathbf{wC})|$ will be minimised when

$$\mathbf{p} = ((\mathbf{wB})^T(\mathbf{wB}))^{-1}(\mathbf{wB})^T(\mathbf{wC}) = \mathbf{DC}, \tag{18}$$

where $\mathbf{D} = ((\mathbf{wB})^T(\mathbf{wB}))^{-1}(\mathbf{wB})^T\mathbf{w}$. Note that $\mathbf{D}$ is a rectangular matrix of $\frac{1}{2}(n+1)(n+2)$ rows by m columns.

The polynomial coefficients thus calculated can be used for interpolation of the function and its derivatives. These are given by

$$f^*(\hat{x},\hat{y}) = \mathbf{b}(\hat{x},\hat{y})\mathbf{p} = \mathbf{b}(\hat{x},\hat{y})\mathbf{DC}, \tag{19}$$

$$f_{\hat{x}}^*(\hat{x},\hat{y}) = \mathbf{b}_{\hat{x}}(\hat{x},\hat{y})\mathbf{p} = \mathbf{b}_{\hat{x}}(\hat{x},\hat{y})\mathbf{DC}, \tag{20}$$

$$f_{\hat{y}}^*(\hat{x},\hat{y}) = \mathbf{b}_{\hat{y}}(\hat{x},\hat{y})\mathbf{p} = \mathbf{b}_{\hat{y}}(\hat{x},\hat{y})\mathbf{DC}. \tag{21}$$

Since the spatial derivatives obtained are taken with respect to $\hat{x}$ and $\hat{y}$, it is necessary to re-dimensionalise these in order to obtain the dimensional versions of these quantities. This can be done using (5) and (6).

Note that the row vectors $\mathbf{b}(\hat{x},\hat{y})\mathbf{D}$, $\mathbf{b}_{\hat{x}}(\hat{x},\hat{y})\mathbf{D}$ and $\mathbf{b}_{\hat{y}}(\hat{x},\hat{y})\mathbf{D}$ are dependent only on the stencil structure and the weighting function w. Hence these row vectors can be pre-computed and stored, then reused as many times as required for time varying data in the $\mathbf{C}$ vector.

3. Singularities

It has been noted [3,2,14,15] that singular cases can exist when using moving least squares, where the matrix requiring inversion for a given stencil is not invertible. In particular, it has been observed that when using a polynomial basis, singularities tend to occur when the data points lie along straight lines, and that these singularities correspond to situations where insufficient information is available to uniquely define the polynomial. In this section, a general criterion for predicting singularities will be derived.

3.1. Singularity conditions for this formulation

The calculation procedure for finding a unique polynomial which best fits a given set of values at a stencil's data points could, in principle, fail for two separate reasons: it would fail if there were no best fit polynomial or if there were multiple best fit polynomials. But a best fit polynomial must exist, since an arbitrarily chosen polynomial will fit the data points with some total error obtained. If no other polynomial can be found which fits with a smaller total error, then the original polynomial gives a best fit. Otherwise, if any other polynomials can be found which provide a better fit, then the best one of those must give a best fit. Therefore, a moving least squares stencil will be singular if and only if multiple best fit polynomials exist.

From (18), it can be seen that having multiple values of the **p** vector (i.e. the best fit polynomial coefficients) corresponding to the same **C** vector (the values at the data points) is possible if and only if there can be multiple values of the **D** matrix. Moving least squares stencils are therefore singular if and only if multiple **D** matrices exist for the stencil. Now the **D** matrix depends only on the **w** and **B** matrices, so the singularity properties of moving least squares stencils must therefore depend only on the weighting function chosen, the stencil geometry and the types of the data points making up the stencil. Most importantly, the **D** matrix does not depend on **C**, and so the singularity properties of moving least squares stencils are not dependent on the actual values specified at the data points. Thus any argument concerning the stability of the moving least squares scheme that can be advanced for one particular set of values at the data points must apply also to all possible sets of data point values.

As an aside, it should also be noted that, providing that none of the weights are equal to zero, the **w** matrix has no effect on stencil singularity properties. This is because, in the expression requiring inversion in (18), the **w** matrix only exists as a pre-multiplication of the **B** matrix, and the rank of the **B** matrix cannot be affected by pre-multiplication by a diagonal matrix possessing only non-zero values on the diagonal.

Consider the process of finding an nth order polynomial of best fit through a stencil of m data points, each one specifying a C value equal to zero (in the sense of (1)). In view of the above argument about the stability of stencils being independent of the data point values, the rest of this discussion may focus on this homogenous case, without any loss of applicability to the more general case of an arbitrary set of values. For this homogenous case, an exact fit (and therefore the best fit) will be obtained if

$$\mathbf{Bp} = \mathbf{0}. \tag{22}$$

Clearly, one set of polynomial coefficients which always satisfies this equation (and thus produces an exact fit) is the trivial solution $\mathbf{p} = \mathbf{0}$. If there is any other solution, $\mathbf{p_0}$, then $k\mathbf{p_0}$ (where k is any real number) must also be a solution and so there will be infinitely many sets of polynomial coefficients which fit the data points exactly; in this case, a unique best fit polynomial cannot be obtained, and so the stencil must be singular. On the other hand, if the trivial solution is the only solution to (22), then by definition that is the unique best fit polynomial, and so the stencil is non-singular. In other words, a moving least squares stencil will be singular for homogenous data point values if and only if the **B** matrix for the stencil possesses a non-trivial null space. Since the singularity properties of stencils are known to be independent of the values at the data points, then it is also true that a moving least squares stencil will be singular for arbitrary data point values if and only if the **B** matrix for the stencil possesses a non-trivial null space.

This somewhat abstract result can be extended, to provide a numerically reliable test for detecting singular stencils and greater insight into the effects of stencil geometry. One way in which this may be done is by means of the singular value decomposition of **B**. The singular value decomposition allows **B** to be expressed as

$$\mathbf{B} = \mathbf{U}\begin{bmatrix} D_1 & 0 & \cdots & 0 \\ 0 & D_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & D_{\frac{1}{2}(n+1)(n+2)} \end{bmatrix}\begin{bmatrix} \mathbf{V}_1^T & \mathbf{V}_2^T & \cdots & \mathbf{V}_{\frac{1}{2}(n+1)(n+2)}^T \end{bmatrix}^T, \tag{23}$$

where **U** is a square matrix with m rows and $1/2(n+1)(n+2)$ orthogonal columns (so that $\mathbf{U}\mathbf{U}^T = \mathbf{U}^T\mathbf{U} = \mathbf{I}$) and the **V** vectors are row vectors with $\frac{1}{2}(n+1)(n+2)$ columns (each vector being orthogonal to and linearly independent from the rest). The null space of **B** can thus be found by finding the null space of a much simpler matrix $\mathbf{\Psi}$, i.e. solving

$$\mathbf{\Psi p_0} = \mathbf{0} \tag{24}$$

for $\mathbf{p_0}$, where

$$\mathbf{\Psi} = \begin{bmatrix} D_1\mathbf{V}_1 \\ D_2\mathbf{V}_2 \\ \vdots \\ D_{\frac{1}{2}(n+1)(n+2)}\mathbf{V}_{\frac{1}{2}(n+1)(n+2)} \end{bmatrix}. \tag{25}$$

It should be noted that, since the **V** vectors are linearly independent, the **Ψ** matrix will be full rank if and only if all the D values are non-zero (or, from a practical numerical point of view, have magnitudes which are all above some small threshold value). In this case the only solution for $\mathbf{p_0}$ is the null vector and the stencil is non-singular. If one or more of the D values are zero, then **Ψ** must be less than full rank, and so there will be an infinite number of possible solutions for $\mathbf{p_0}$ and the stencil will be singular. This, then, is a reliable test for determining whether or not a stencil is singular.

The null space of the **B** matrix (which is the same as the null space of the **Ψ** matrix) can be thought of as a set of polynomial coefficients which generate C^* values of zero at all stencil data points. If $\mathbf{p_0}$ is a non-trivial member of the null space of **B**, therefore, the relation $(a_j\mathbf{b}(\hat{x}_j,\hat{y}_j)+\widehat{L}_j n_{xj}\mathbf{b}_{\hat{x}}(\hat{x}_j,\hat{y}_j)+\widehat{L}_j n_{yj}\mathbf{b}_{\hat{y}}(\hat{x}_j,\hat{y}_j))\mathbf{p_0}=0$ defines an algebraic constraint which is obeyed at each data point j in the stencil. The set of linearly independent $\mathbf{p_0}$ vectors in the null space of **B** thus provides the complete basis to the set of algebraic constraints of nth order obeyed by the coordinates of the stencil's data points. The fact that the stencil coordinates obey any such nth order algebraic constraint at all can be thought of as the geometrical cause of the stencil's singularity. These constraint equations will be termed 'spanning polynomials', for reasons which will become apparent when the special case of stencils consisting only of function specified data points is considered.

It is a well known fact of linear algebra that the vectors in the columns of the right hand matrix (**V**) of a matrix's singular value decomposition which correspond to singular values equal to zero in the diagonal matrix form a complete basis for the null space of the original matrix [6]. This means the coefficients for the family of polynomials which span a stencil can be generated by taking a linear combination of the **V** vectors corresponding to the D values which are equal to zero. Since the ordering of the rows in (25) is arbitrary, it is reasonable to assume that the rows in this equation are ordered such that the singular (D) values are ordered from smallest magnitude to largest magnitude. (If the singular value decomposition algorithm does not produce output with this property, then the rows of (25) may be sorted.) Suppose that there are s singular values equal to zero, where $1\leqslant s\leqslant\frac{1}{2}(n+1)(n+2)$. (In the case when $s=0$ the stencil is not singular and so the only spanning polynomial is the trivial polynomial, with all coefficients equal to zero.) In view of the assumed ordering, the zero singular values will then be $D_1, D_2, \ldots D_s$, and their corresponding **V** vectors will be $\mathbf{V}_1, \mathbf{V}_2, \ldots \mathbf{V}_s$. The family of spanning polynomials can then be expressed as

$$\mathbf{p_0}=\eta_1\mathbf{V}_1^T+\eta_2\mathbf{V}_2^T+\cdots+\eta_s\mathbf{V}_s^T, \tag{26}$$

where $\eta_1,\eta_2,\ldots,\eta_s$ are real parameters which may be varied arbitrarily to generate the family of spanning polynomials. Note that the transpose operators are necessary because $\mathbf{p_0}$ is defined as a column vector whereas the **V** vectors are defined as row vectors. The parameter s can be thought of as the number of dimensions in the space of spanning polynomials.

3.2. Generalisation to arbitrary basis function

The results obtained for the polynomial based moving least squares scheme can easily be extended to a scheme with an arbitrary basis function. Let the basis function chosen consist of n linearly independent components, each an arbitrary function of $\hat{x}$ and $\hat{y}$, so that the basis vector may be redefined as

$$\mathbf{b}(\hat{x},\hat{y})=(g_1(\hat{x},\hat{y}),g_2(\hat{x},\hat{y}),\ldots,g_n(\hat{x},\hat{y})). \tag{27}$$

The spatial derivative vectors are then

$$\mathbf{b}_{\hat{x}}(\hat{x},\hat{y})=\left(\left.\frac{\partial g_1}{\partial\hat{x}}\right|_{(\hat{x},\hat{y})},\left.\frac{\partial g_2}{\partial\hat{x}}\right|_{(\hat{x},\hat{y})},\ldots,\left.\frac{\partial g_n}{\partial\hat{x}}\right|_{(\hat{x},\hat{y})}\right) \tag{28}$$

and

$$\mathbf{b}_{\hat{y}}(\hat{x},\hat{y})=\left(\left.\frac{\partial g_1}{\partial\hat{y}}\right|_{(\hat{x},\hat{y})},\left.\frac{\partial g_2}{\partial\hat{y}}\right|_{(\hat{x},\hat{y})},\ldots,\left.\frac{\partial g_n}{\partial\hat{y}}\right|_{(\hat{x},\hat{y})}\right) \tag{29}$$

and so (14) can then be used to define the **B** matrix. If this matrix possesses a non-trivial null space, then the stencil is singular; otherwise, it is non-singular. The rest of the analysis from the previous subsection proceeds in much the same manner, except that now there are n singular values and **V** vectors (not $\frac{1}{2}(n+1)(n+2)$) and the **p** vector now represents the coefficients of the arbitrary basis function components.

It should be noted that singularities in moving least squares stencils are highly basis function dependent. It is quite possible for a stencil which is singular when using a polynomial basis function not to be singular when using some other basis, and vice versa.

At this point, it is instructive to consider what happens when an arbitrary basis function is extended to include additional terms. Suppose that a given stencil is singular when using a basis function $\mathbf{b}(\hat{x},\hat{y})$, as defined by (27). This means that the corresponding **B** matrix given by (14) possesses a non-trivial null space, one member of which is $(p_1,p_2,\ldots,p_n)^T$. If an extended basis function were used instead, namely

$$\mathbf{b}_{\text{ext}}(\hat{x},\hat{y})=\big(g_1(\hat{x},\hat{y}),g_2(\hat{x},\hat{y}),\ldots,g_n(\hat{x},\hat{y}),g_{n+1}(\hat{x},\hat{y})\big), \tag{30}$$

then a **B** matrix may be obtained for the stencil using the extended basis, which is

$$\mathbf{B}_{\text{ext}} = \begin{bmatrix} a_1\mathbf{b}_{\text{ext}}(\hat{x}_1,\hat{y}_1) + \widehat{L}_1 n_{x,1}\mathbf{b}_{\text{ext},\hat{x}}(\hat{x}_1,\hat{y}_1) + \widehat{L}_1 n_{y,1}\mathbf{b}_{\text{ext},\hat{y}}(\hat{x}_1,\hat{y}_1) \\ a_2\mathbf{b}_{\text{ext}}(\hat{x}_2,\hat{y}_2) + \widehat{L}_2 n_{x,2}\mathbf{b}_{\text{ext},\hat{x}}(\hat{x}_2,\hat{y}_2) + \widehat{L}_2 n_{y,2}\mathbf{b}_{\text{ext},\hat{y}}(\hat{x}_2,\hat{y}_2) \\ \vdots \\ a_m\mathbf{b}_{\text{ext}}(\hat{x}_m,\hat{y}_m) + \widehat{L}_m n_{x,m}\mathbf{b}_{\text{ext},\hat{x}}(\hat{x}_m,\hat{y}_m) + \widehat{L}_m n_{y,m}\mathbf{b}_{\text{ext},\hat{y}}(\hat{x}_m,\hat{y}_m) \end{bmatrix}. \tag{31}$$

$\mathbf{B}_{\text{ext}}$ must also possess a non-trivial null space, since the vector $(p_1, p_2, \ldots, p_n, 0)^T$ is a member of the null space of $\mathbf{B}_{\text{ext}}$, and is clearly non-trivial. This proves that a singular stencil remains singular when additional terms are added to the basis function. An important corollary of this is that a stencil which is singular when fitting an nth order polynomial must also be singular when fitting a polynomial with order greater than n, since this entails the addition of terms to the basis function.

3.3. The special case of function-specified data points

In the special case where a stencil consists exclusively of function specified data points (i.e. data points with $a = 1, \widehat{L} = 0$), the singularity theory presented previously becomes a great deal simpler and more intuitive. For a start, the **B** matrix (when using an nth order polynomial basis) is reduced to a Vandermonde matrix, namely

$$\mathbf{B} = \begin{bmatrix} 1 & \hat{x}_1 & \hat{y}_1 & \hat{x}_1^2 & \hat{x}_1\hat{y}_1 & \hat{y}_1^2 & \cdots & \hat{x}_1^n & \hat{x}_1^{n-1}\hat{y}_1 & \cdots & \hat{x}_1\hat{y}_1^{n-1} & \hat{y}_1^n \\ 1 & \hat{x}_2 & \hat{y}_2 & \hat{x}_2^2 & \hat{x}_2\hat{y}_2 & \hat{y}_2^2 & \cdots & \hat{x}_2^n & \hat{x}_2^{n-1}\hat{y}_2 & \cdots & \hat{x}_2\hat{y}_2^{n-1} & \hat{y}_2^n \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & \hat{x}_m & \hat{y}_m & \hat{x}_m^2 & \hat{x}_m\hat{y}_m & \hat{y}_m^2 & \cdots & \hat{x}_m^n & \hat{x}_m^{n-1}\hat{y}_m & \cdots & \hat{x}_m\hat{y}_m^{n-1} & \hat{y}_m^n \end{bmatrix}. \tag{32}$$

It can easily be seen that the null space of this matrix is the set of all polynomial coefficients $\mathbf{p_0}$ such that $\left(1, \hat{x}_j, \hat{y}_j, \hat{x}_j^2, \hat{x}_j\hat{y}_j, \hat{y}_j^2, \ldots, \hat{x}_j^n, \hat{x}_j^{n-1}\hat{y}_j, \ldots, \hat{x}_j\hat{y}_j^{n-1}, \hat{y}_j^n\right)\mathbf{p_0} = 0$ for all data points j in the stencil. In effect, this means the null space of the **B** matrix (for nth order polynomial based moving least squares) generated from a stencil consisting only of function specified data points is the set of all nth (or lower) order polynomial relationships equating to zero which sweep out all of the stencil's data points. This provides an alternative way of looking at singular stencils: function specified stencils (for nth order polynomial based moving least squares) are singular if and only if there exists an nth (or lower) order polynomial relationship equating to zero which sweeps out all of the stencil's data points.

An important corollary of this is that an nth order polynomial surface cannot be uniquely fitted to a stencil of (function specified) data points, if it is possible to draw n straight lines that between them pass through all the data points in the stencil. The proof for this is simple. A single straight line is the set of all $(\hat{x}, \hat{y})$ points satisfying the general equation $a_1\hat{x} + b_1\hat{y} + c_1 = 0$, where at least one of a_1, b_1 is non-zero. The general equation for n straight lines is then the nth order polynomial $(a_1\hat{x} + b_1\hat{y} + c_1)(a_2\hat{x} + b_2\hat{y} + c_2)\cdots(a_n\hat{x} + b_n\hat{y} + c_n) = 0$, where at least one of a_i, b_i is non-zero for $i \in 1, \ldots, n$. If all the data points can be swept out by n straight lines, then there is an nth order polynomial in $\hat{x}$ and $\hat{y}$ which equates to zero and passes through all the data points in the stencil, and therefore it will be impossible to fit a unique nth order polynomial to the stencil. This result explains the observations made by Schoenauer and Adolph [16] and Bodin [2]. However, it is important to note that n straight lines are not the only possible nth order polynomials, and therefore there are many other configurations of the data points that also result in singularities.

Fig. 1, for example, shows a stencil of function specified data points (shown as hollow circles) which is singular when fitting a third order polynomial in $\hat{x}$ and $\hat{y}$. This stencil is singular because it is possible to sweep out all its data points with a third order polynomial relation in $\hat{x}$ and $\hat{y}$. In fact there are many such relations, and the entire family of spanning polynomials may be expressed as

$$a\hat{y} + 4b\hat{x}^2\hat{y} - (a+b)\hat{y}^3 = 0, \tag{33}$$

where a and b are arbitrary parameters that may be varied in order to generate the family. The number of arbitrary parameters may be reduced to 1 by dividing through by b, as long as the special case of $b = 0$ is taken into account separately. Fig. 1 also shows the spanning polynomial for a number of values of the parameter $\frac{a}{b}$, as well as for $b = 0$. It can be seen that there are 6 fundamentally different cases in the family, namely:

- Three horizontal lines;
- An ellipse and one horizontal line;
- Two vertical lines and one horizontal line;
- A hyperbola and one horizontal line;
- Two skew lines and one horizontal line;
- An inverted hyperbola and one horizontal line.

The important conclusion that may be drawn from this example is that, since this family of spanning polynomials (between them) sweeps out the entire plane, adding one more data point is guaranteed to be insufficient to resolve the

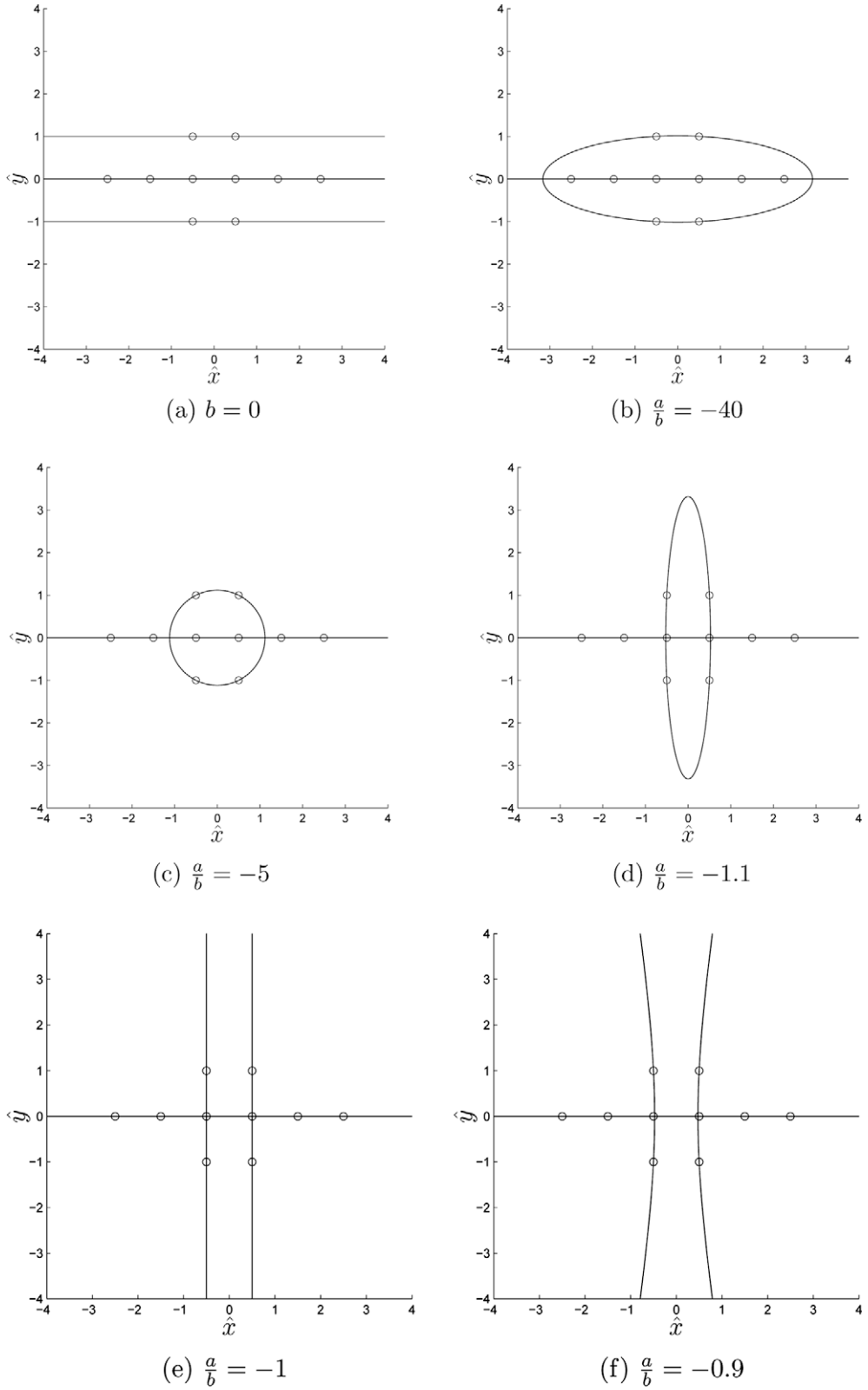
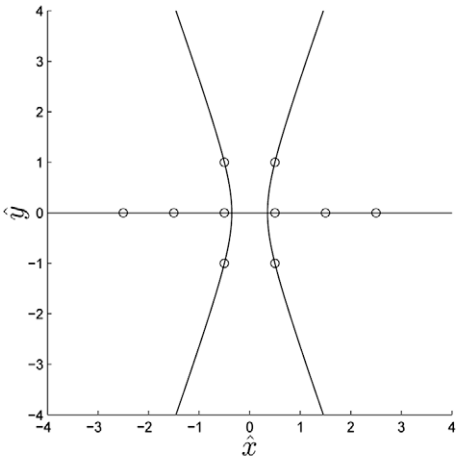
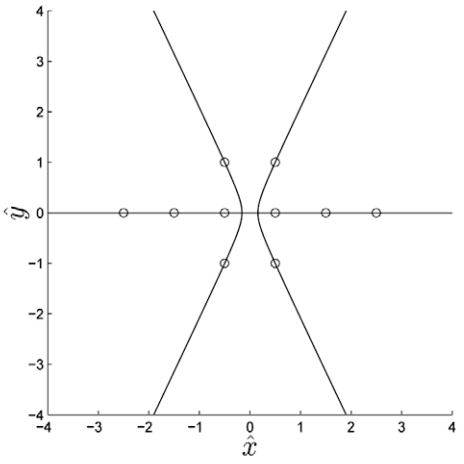


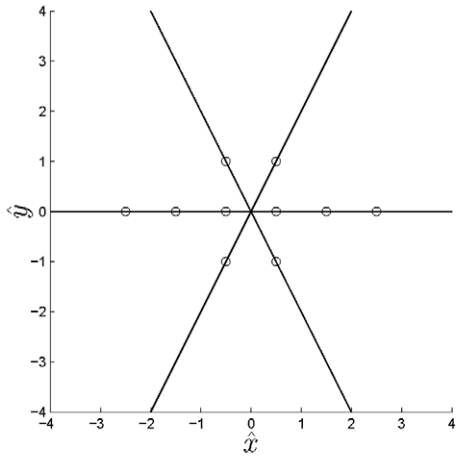
Fig. 1. A third order singular stencil, with two independent null space parameters, *a* and *b*. The sub-figures show the family of third order polynomials in $\hat{x}$ and $\hat{y}$ that fit the data points. All data points are function specified, and are shown as hollow dots.



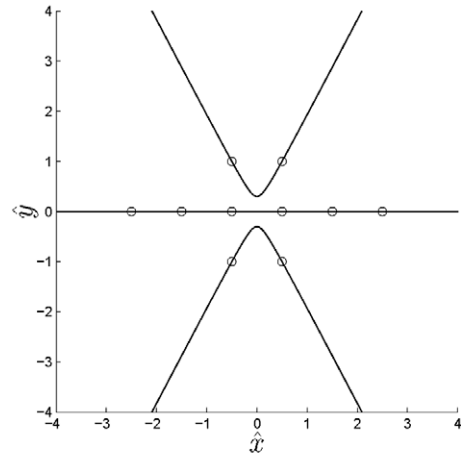
(g) $\frac{a}{b} = -0.5$



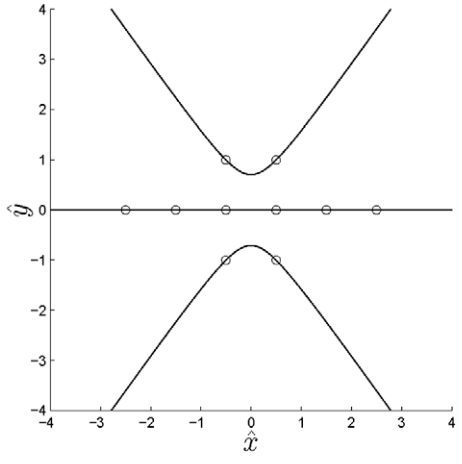
(h) $\frac{a}{b} = -0.1$



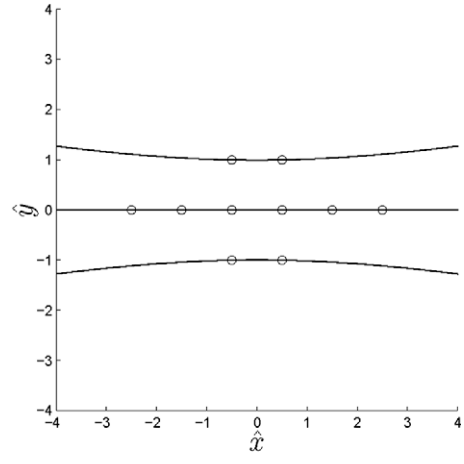
(i) $\frac{a}{b} = 0$



(j) $\frac{a}{b} = 0.1$



(k) $\frac{a}{b} = 1$



(ℓ) $\frac{a}{b} = 100$

Fig. 1 (*continued*)

singularity. This is because, wherever the new data point is located, there exists a member of the family of spanning polynomials which spans the new data point also. In fact, at least two new data points must be added in order to resolve the singularity in this example. The theory behind this is explored in more detail in Section 3.5. It is also worth noting that the type of data points present in a stencil can have a very significant impact on its singularity properties. Fig. 2(a), for instance, is a stencil consisting of two function specified data points (solid dots) and one derivative specified data point (hollow dot, with arrow indicating direction of spatial derivative). This stencil is non-singular when fitting a first order polynomial. However, this stencil would become singular if the derivative specified data point were replaced with a function specified data point or if, instead, the direction of the spatial derivative were made parallel to the line that the data points lie on. The converse can also occur: Fig. 2(b) also has two function specified data points and one derivative specified data point, but is singular (when using a first order polynomial basis) because the direction of the spatial derivative is parallel to the line that the two function specified data points lie on. If the derivative specified data point were replaced with a function specified data point, however, then the stencil would become non-singular. The existence of such subtle behaviour highlights the importance of using the general theory of singularities when using a mixture of data point types.

3.4. Addition of data points to singular stencils

Before designing an algorithm for building non-singular stencils, it is necessary to consider what would happen to the family of spanning polynomials if a new data point were added to a singular stencil. Suppose that the new data point is located at $(\hat{x}_N, \hat{y}_N)$ and has data point properties of $a, \widehat{L}, n_x$ and n_y (as defined in (8)). The spanning polynomial $\mathbf{p_0}$ of the original singular stencil will also span the new data point when

$$\left[a\mathbf{b}(\hat{x}_N,\hat{y}_N) + \widehat{L}n_x\mathbf{b}_{\hat{x}}(\hat{x}_N,\hat{y}_N) + \widehat{L}n_y\mathbf{b}_{\hat{y}}(\hat{x}_N,\hat{y}_N)\right]\mathbf{p_0} = 0. \tag{34}$$

If this equation is combined with (26), then it may be written as

$$\kappa_1\eta_1 + \kappa_2\eta_2 + \cdots + \kappa_s\eta_s = 0, \tag{35}$$

where $\kappa_i = [a\mathbf{b}(\hat{x}_N,\hat{y}_N) + \widehat{L}n_x\mathbf{b}_{\hat{x}}(\hat{x}_N,\hat{y}_N) + \widehat{L}n_y\mathbf{b}_{\hat{y}}(\hat{x}_N,\hat{y}_N)]\mathbf{V}_i^T$.

There are two possible outcomes that could occur when a new data point is added to a singular stencil. One possibility is that the new data point happens to be spanned by all of the polynomials defined by vectors $\mathbf{V}_1, \mathbf{V}_2, \ldots, \mathbf{V}_s$ (which form a complete basis to the family of spanning polynomials for the original stencil). In this case, the new data point will be spanned by every polynomial in the family. Moreover, all of the κ values in (35) will be equal to zero, and so that equation will provide no information concerning the η values or the relationship between them. Thus, the new data point does not help to fix the singularity, since the family of spanning polynomials has not been reduced in any way.

The other possible outcome of adding the new data point is that at least one of the polynomials defined by vectors $\mathbf{V}_1, \mathbf{V}_2, \ldots, \mathbf{V}_s$ does not span the new data point. This means that at least one of the κ values (κ_j) will be non-zero. It is then possible to solve for η_j in terms of the other η parameters, obtaining the expression

$$\eta_j = -\frac{1}{\kappa_j}\sum_{i=1,\ldots,s,i\neq j}\kappa_i\eta_i. \tag{36}$$

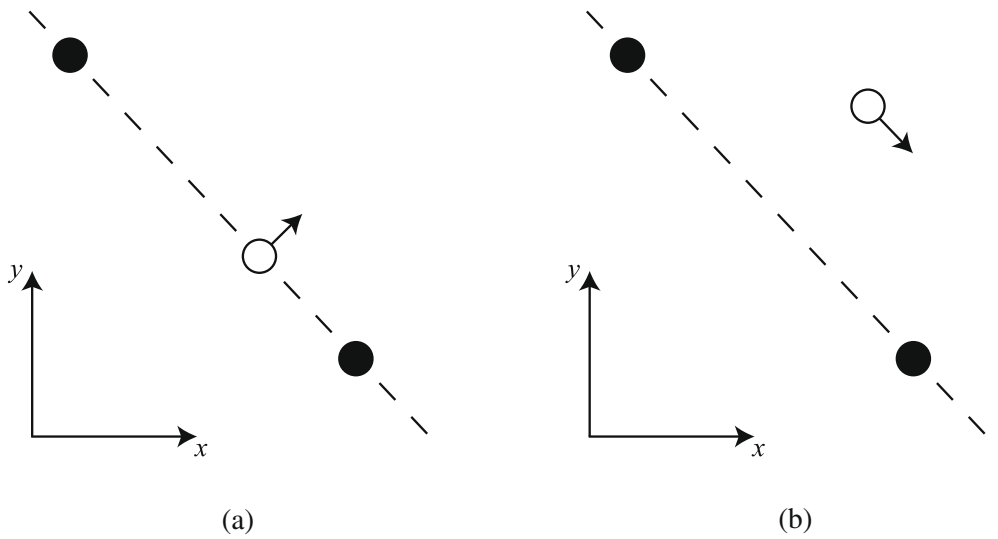


Fig. 2. Examples of stencils which are (a) non-singular and (b) singular when using a first order polynomial basis. Function specified data points are shown as solid, black dots. Hollow circles represent the derivative specified data point, with the arrow indicating the direction in which the spatial derivative is specified.

This means that the family of polynomials which spans both the original stencil and the new data point can be written as a vector space with $s-1$ dimensions, i.e. one dimension less than the family of polynomials spanning just the original stencil. Thus it can be seen that the addition of a new data point helps to reduce the size of the family of spanning polynomials, as long as at least one of the polynomials defined by the $\mathbf{V}$ vectors of the original stencil does not span the new data point. Moreover, the dimensions of the family of spanning polynomials will be reduced by exactly one regardless of whether there was only one $\mathbf{V}$ vector polynomial that did not span the new data point or there were any number up to and including s.

There is a special case that also deserves a mention, and that is when the family of polynomials spanning the original stencil is one dimensional, i.e. when $s=1$. The family is then simply

$$\mathbf{p_0} = \eta_1 \mathbf{V}_1^T. \tag{37}$$

If the spanning polynomial family is also to span a new data point, then it must be the case that

$$\kappa_1 \eta_1 = 0, \tag{38}$$

where $\kappa_1 = [a\mathbf{b}(\hat{x}_N,\hat{y}_N) + \widehat{L}n_x\mathbf{b}_{\hat{x}}(\hat{x}_N,\hat{y}_N) + \widehat{L}n_y\mathbf{b}_{\hat{y}}(\hat{x}_N,\hat{y}_N)]\mathbf{V}_1^T$. If the new data point is spanned by the polynomial corresponding to $\mathbf{V}_1$, then $\kappa_1 = 0$ and so η_1 could have any value whatsoever; hence the new data point will be completely useless. However, if the new data point is not spanned by the polynomial corresponding to $\mathbf{V}_1$, then $\kappa_1 \neq 0$ and so $\eta_1 = 0$; hence, the only polynomial spanning both the original stencil and the new data point is the trivial polynomial (all coefficients equal to zero). If this is the case, adding the new data point removes the singularity and makes the stencil stable.

In all cases, adding a new data point to a singular stencil is effective at reducing the singularity if and only if it is not spanned by all the spanning polynomials of the original stencil.

3.5. Algorithm for building non-singular stencils

Stencil building algorithms can employ the theory presented in this paper in the following ways:

- A singular stencil may be detected by computing the singular value decomposition of the stencil's $\mathbf{B}$ matrix. If all the D values are greater in magnitude than some small threshold value, then the stencil is considered to be non-singular; therefore, the stencil building procedure is complete. If one or more of the D values is smaller in magnitude than the threshold, then the stencil is considered to be singular. In this case, the stencil requires more data points. A value of 0.2 was found to be an appropriate threshold for this test, and was used in all the simulations discussed in this paper.
- For a singular stencil, the $\mathbf{V}$ vector corresponding to the smallest magnitude D value is the polynomial that most accurately spans the data points of the stencil. This is thus the best choice for use in spanning tests. (In view of the aforementioned ordering convention of right singular vectors, this vector is $\mathbf{V}_1$.) A new data point will be more effective at removing the singularity if it lies further off this spanning polynomial, so the best choice is the data point for which $|(a\mathbf{b}(\hat{x}_N,\hat{y}_N) + \widehat{L}n_x\mathbf{b}_{\hat{x}}(\hat{x}_N,\hat{y}_N) + \widehat{L}n_y\mathbf{b}_{\hat{y}}(\hat{x}_N,\hat{y}_N))\mathbf{V}_1|$ is maximised. This principle was applied in all the simulations discussed in this paper.
- If the null space of the $\mathbf{B}$ matrix has more than one dimension, then adding a single data point cannot remove the singularity by itself. However, it may reduce the dimensions of the null space by one. Subsequent data point additions must be used to complete the singularity removal.

It is also desirable that stencil building algorithms take the following considerations into account. Note that algorithms which do this can often end up building larger stencils than necessary, but the trade off may be worthwhile in many applications:

- The data points included in a stencil should, as much as possible, be local to the interpolation point. This may be achieved by requiring that a new data point can only be added to a stencil if it is adjacent (in terms of the mesh) to an existing data point in the stencil. A stronger requirement may also be imposed, which is that all the data points at a given connectivity depth to the interpolation point must be added to the stencil before any data point with a higher connectivity depth will be considered for addition. This more stringent locality requirement was applied in all the simulations discussed in this paper.
- It is well known in the literature that better spectral performance may be obtained from finite difference schemes when symmetric stencils are employed. (In one spatial dimension, these are the popular central difference schemes.) Similarly, it may be shown that moving least squares stencils provide better spectral performance when they are as close as possible to being symmetric about axes parallel and normal to the interpolation edge (both of which pass through the interpolation point itself). Stencil building algorithms should therefore attempt to build balanced stencils, by adding mirror data points for each new data point chosen according to the singularity theory. The stencil building algorithm used for simulations presented in this paper does this.

When starting to build a stencil, an initial core of data points must be selected for the stencil, so that a $\mathbf{B}$ matrix may be constructed and used to determine which data point should be added to the stencil next. Ideally, this core should be as small as possible, so that inclusion of redundant data points is avoided. Also, it is desirable for the data points in the core stencil to

be very close to the interpolation point. If the edge on which an interpolation is required is on a domain boundary, then the interpolation point overlaps with a data point available from the boundary conditions; this data point is an appropriate choice for the core stencil. Otherwise, if the interpolation edge is internal to the domain, then its closest data points are located at the centroids of the two elements adjacent to it; these data points are an appropriate choice for the core stencil. These choices were used for all the simulations in this paper.

The stencil building algorithm used in this paper is outlined in Algorithm 1, with *SingularityTolerance* = 0.2. Typical stencils generated by this algorithm for third order moving least squares are shown in Figs. 3–5.

Algorithm 1. A non-singular stencil building algorithm

Require: *Edge* = the edge requiring a stencil to be built around it, *SingularityTolerance* = the largest magnitude a singular value may have while still being considered equal to zero
Ensure: *Stencil* = the list of all data points in the finished stencil
if *Edge* is a boundary edge **then**
  Stencil ← the midpoint of *Edge*
else [*Edge* is an internal edge]
  Stencil ← the centroids of the two elements adjacent to *Edge*
endif
loop
  Candidates ← the set of data points which are adjacent to the current stencil, but which are not already part of it.
  while there are still data points in *Candidates* **do**
    Find the *SingularValues* and *RightSingularVectors* of the current stencil's **B** matrix, using the singular value decomposition
    if there are no *SingularValues* with a magnitude less than *SingularityTolerance* **then**
      return
    endif
    Choice ← the candidate data point for which $|(a\mathbf{b}(\hat{x}_N,\hat{y}_N)+\widehat{L}n_x\mathbf{b}_{\hat{x}}(\hat{x}_N,\hat{y}_N)+\widehat{L}n_y\mathbf{b}_{\hat{y}}(\hat{x}_N,\hat{y}_N))\mathbf{V}_1|$ is maximum.
    Candidates ← *Candidates* \ *Choice*
    Stencil ← *Stencil* ∪ *Choice*
    Set *Mirror*1, *Mirror*2 and *Mirror*3 to be the locations of the mirror images of *Choice* about axes parallel and normal to *Edge*.
    if there is a data point in *Candidates* which is closer to *Mirror*1 than any data point in *Stencil* is **then**
      ExtraChoice ← the closest data point to *Mirror*1 in *Candidates*
      Candidates ← *Candidates* \ *ExtraChoice*
      Stencil ← *Stencil* ∪ *ExtraChoice*
    endif
    if there is a data point in *Candidates* which is closer to *Mirror*2 than any data point in *Stencil* is **then**
      ExtraChoice ← the closest data point to *Mirror*2 in *Candidates*
      Candidates ← *Candidates* \ *ExtraChoice*
      Stencil ← *Stencil* ∪ *ExtraChoice*
    endif
    if there is a data point in *Candidates* which is closer to *Mirror*3 than any data point in *Stencil* is
      ExtraChoice ← the closest data point to *Mirror*3 in *Candidates*
      Candidates ← *Candidates* \ *ExtraChoice*
      Stencil ← *Stencil* ∪ *ExtraChoice*
    end if
  end while
end loop

4. Verification of scheme

4.1. Verification using test functions

4.1.1. General approach

A very simple way of testing the accuracy of an interpolation scheme on a given mesh is to make up an analytical function f and evaluate this test function at the element centroids (and boundary edge midpoints) of the mesh. These values may be used as inputs to the interpolation scheme, from which estimates of the value of f (and its spatial derivatives) at the edge midpoints may be obtained. Finally, the accuracy of the interpolation scheme may be assessed by comparing the interpolated values at the edge midpoints with values obtained from the analytical function.

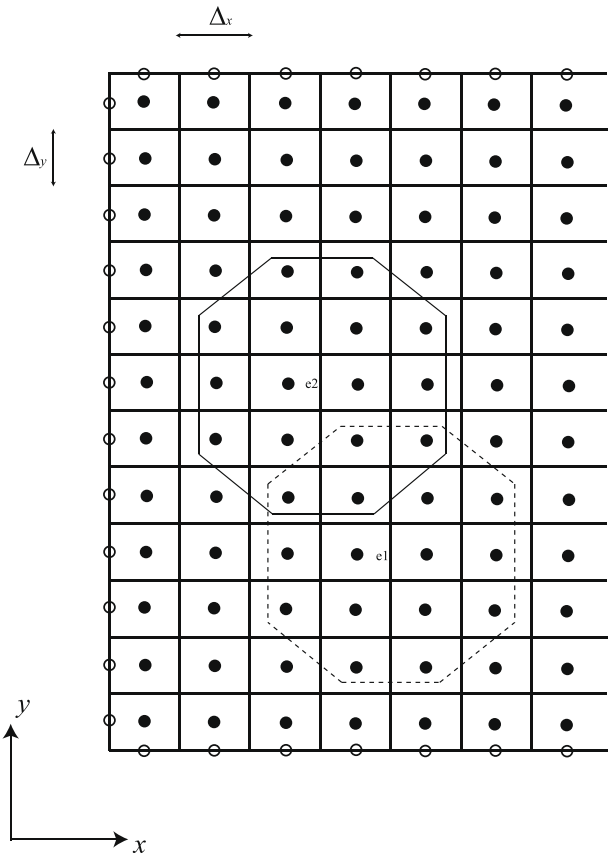


Fig. 3. Example stencils for moving least squares on a regular mesh. The third order moving least squares stencil for interpolating on the edge marked e1 is enclosed by the dashed line, while the stencil for interpolating on the edge marked e2 is enclosed by the solid line. The hollow black dots are data points from boundary conditions and the solid black dots are data points at the element centroids.

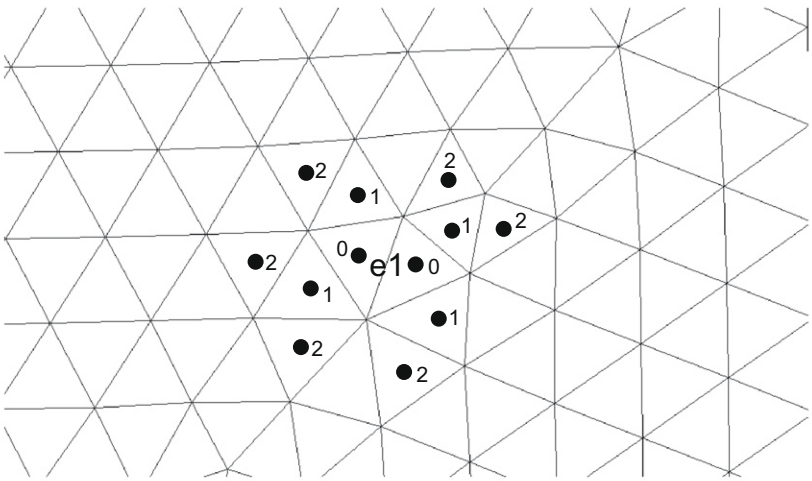


Fig. 4. Example stencil for moving least squares on an irregular mesh. The third order moving least squares stencil for interpolating on the edge marked e1 is shown. This stencil consists only of element centroid data points, which are shown as black dots. The numbers indicate the connectivity depth of the data points to the two initial data points (which are marked with zeros).

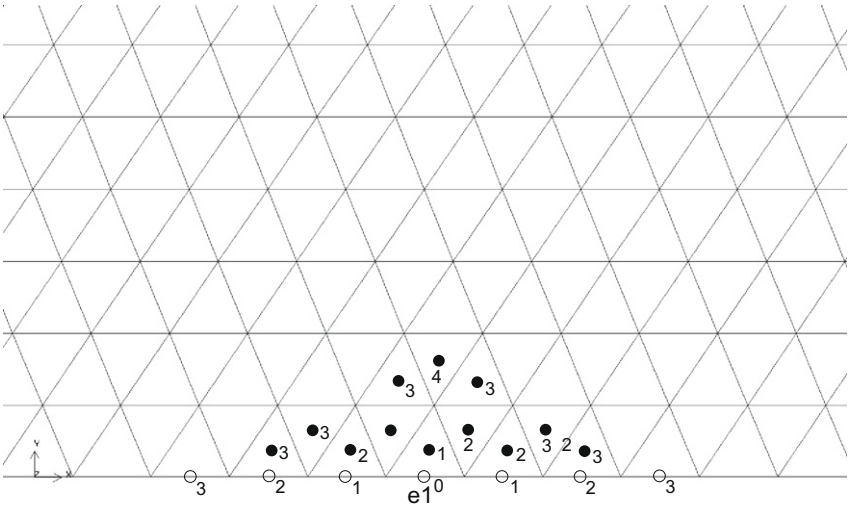


Fig. 5. Example boundary edge stencil for moving least squares on a locally regular mesh, when the value of f is specified on the boundaries. The third order moving least squares stencil for interpolating on the edge marked e1 is shown. This stencil consists of element centroid data points (solid black dots) and boundary edge midpoint data points (hollow dots). The numbers indicate the connectivity depth of the data points to the one initial data point (which is marked with a zero).

Specifically, the overall f interpolation accuracy may be gauged by computing

$$E_f = \frac{\sqrt{\sum_{i\in \text{ edges}} \left(f_i^* - f_i\right)^2}}{(f_{\max} - f_{\min})\sqrt{n_{\text{edges}}}}, \tag{39}$$

where f_i^* is the interpolated value of f on edge i, f_i is the analytical value of f on edge i, n_{edges} is the number of edges in the mesh and $f_{\max}$ and $f_{\min}$ represent the extreme values taken by f over the domain. This quantity is the root mean square interpolation error in f, normalised by the range of the test function. Likewise, the error in the interpolation of the x direction spatial derivative may be gauged using

$$E_{df/dx} = \frac{\sqrt{\sum_{i\in \text{ edges}} \left(\left.\frac{df^*}{dx}\right|_i - \left.\frac{df}{dx}\right|_i\right)^2}}{\left(\left.\frac{df}{dx}\right|_{\max} - \left.\frac{df}{dx}\right|_{\min}\right)\sqrt{n_{\text{edges}}}}, \tag{40}$$

while the error in the interpolation of the y direction spatial derivative may be gauged using $E_{df/dy}$, which is given by a similar expression.

For certain purposes, it is more meaningful to study the worst interpolation errors, rather than the overall interpolation accuracy. The worst f interpolation errors may be gauged using

$$M_f = \frac{\max\limits_{i\in\text{edges}} \left|f_i^* - f_i\right|}{(f_{\max} - f_{\min})}, \tag{41}$$

while the worst spatial derivative errors may be gauged using $M_{df/dx}$ and $M_{df/dy}$, which are given by similar expressions.

An appropriate choice for f is a sinusoidal function in both x and y, since this is an infinite order function, varies with both x and y and has the same smoothness properties throughout the domain. In contrast, finite order polynomials would not be appropriate, since they are of finite order and can be reproduced exactly with a moving least squares scheme of the same order.

Experience and common sense suggests that test function values will be easier to replicate accurately using interpolation if the test function is smooth at the scale of the mesh elements. For the analytical values of a sinusoidal test function to be reproduced accurately by interpolation, there need to be several samples per period of the wave. In other words, the period of a sinusoidal test function needs to be several times the size of a typical mesh element if accurate reproduction is to be obtained. The number of mesh elements N per test function period may then be taken as a measure of the difficulty of the test function. The performance of a given interpolation scheme may be gauged by using a range of values of N, in order to find out how quickly the scheme's performance degenerates as N is reduced towards 2, the Nyquist limit.

Consider a regular rectangular mesh with a mesh spacing of Δ_x in the x direction and Δ_y in the y direction. In view of the previous discussion, a suitable test function for this class of mesh is

$$f(x,y) = \cos\left(\frac{2\pi}{N\Delta_x}x + \frac{2\pi}{N\Delta_y}y\right), \tag{42}$$

where N is a parameter which determines the number of data points per period of the wave along each coordinate axis. N does not need to be an integer, but since $N = 2$ corresponds to the Nyquist limit, values of N greater than 2 must be selected.

The test function may be extended to irregular meshes if some rough approximations are applied. To this end, a general mesh will be represented by a regular rectangular mesh, which is in some sense equivalent to the original mesh. Let element i have a maximum x value (i.e. at its rightmost corner node) of $x_{i,\max}$, minimum x value of $x_{i,\min}$, maximum y value of $y_{i,\max}$ and minimum y value of $y_{i,\min}$. Further, let A_i be the area of element i and M be the number of elements in the mesh. The aspect ratio of the rectangular elements in the equivalent mesh should be similar to the aspect ratio of the elements in the original mesh, and so an equivalent aspect ratio R may be defined as

$$R = \frac{\frac{1}{M}\sum_i (y_{i,\max} - y_{i,\min})}{\frac{1}{M}\sum_i (x_{i,\max} - x_{i,\min})} = \frac{\sum_i (y_{i,\max} - y_{i,\min})}{\sum_i (x_{i,\max} - x_{i,\min})}. \tag{43}$$

In effect, R is the ratio between the average y range of the elements to the average x range. This measure gives reasonable results on a range of meshes. Next, a basic mesh scale Δ may be calculated by taking the square root of the average element area, i.e.

$$\Delta = \sqrt{\frac{1}{M}\sum_i A_i}. \tag{44}$$

Finally, the equivalent mesh scales in the x and y directions may be computed using

$$\Delta_x = \frac{\Delta}{\sqrt{R}} \tag{45}$$

and

$$\Delta_y = \Delta\sqrt{R}, \tag{46}$$

so that $\Delta_x\Delta_y = \frac{1}{M}\sum_i A_i$ and $\frac{\Delta_y}{\Delta_x} = R$. Note that if this method is applied to a regular rectangular mesh, the equivalent mesh is the same as the original mesh, as expected. With this approximation, test function (42) may be applied to unstructured meshes of triangles and/or quadrilaterals, producing consistent results.

For example, Fig. 6 shows the graphs of $\log_{10}(E_f)$, $\log_{10}(E_{df/dx})$ and $\log_{10}(E_{df/dy})$ versus $\log_{10}(N)$ for third order moving least squares, when test function (42) is applied in this manner to the unstructured mesh displayed in Fig. 12. Logarithms are used on both axes, in order to clearly represent the shape of the error curves. Similarly, Fig. 7 shows the same for the unstructured mesh displayed in Fig. 17. These curves represent the decay in error as N is increased, and are typical in shape of the error curves which may be obtained on a wide variety of unstructured meshes. Since N is the ratio of the wavelength of the test

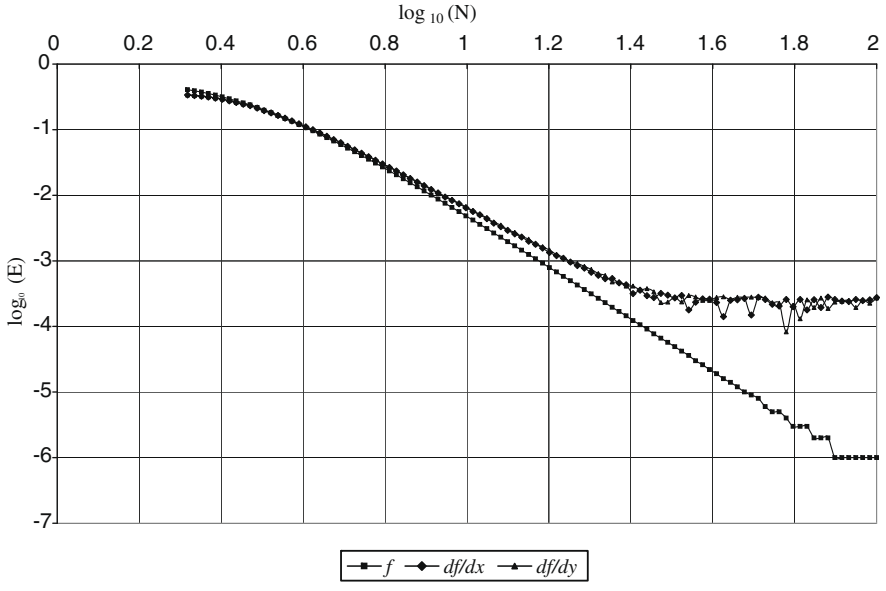


Fig. 6. Interpolation test function results for the mesh in Fig. 12.

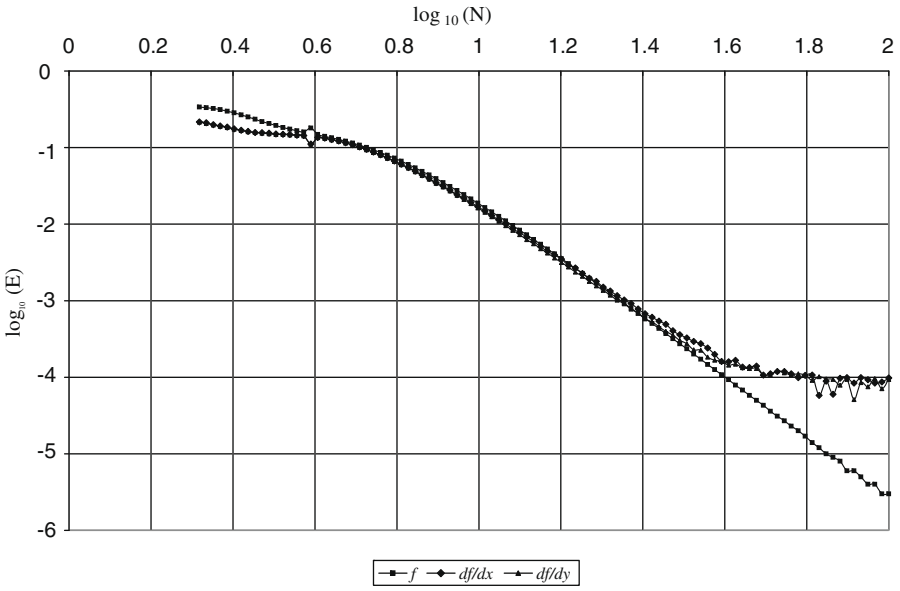


Fig. 7. Interpolation test function results for the mesh in Fig. 17.

function to the scale of the mesh, these graphs were obtained by increasing the wavelength of the test function, while keeping the mesh scale constant. (This is, in principle, equivalent to reducing the mesh scale while keeping the test function wavelength constant.) As expected, the errors decay rapidly when this happens. This demonstrates the accuracy of the moving least squares scheme.

Theory predicts that the third order moving least squares scheme has an f interpolation error proportional to Δ^4, i.e. proportional to N^{-4}. (Informally, this is because the nth order polynomial of best fit is a close approximation to the nth order Taylor representation of f at the interpolation point. The leading terms of the f interpolation error are therefore of order $n+1$. A formal proof along similar lines is known to the authors, but is omitted for brevity.) The figures support the theory, as the curve for $\log_{10}(E_f)$ versus $\log_{10}(N)$ has a straight line section with a gradient of approximately -4 in both cases (-3.98 and -3.93 for Figs. 6 and 7, respectively). Thus the third order moving least squares scheme is fourth order accurate for interpolating f, as expected.

4.1.2. Selection of the value of ϵ

The effect of varying the shape parameter ϵ may be assessed using test functions. For this purpose, (42) was used with $N = 9.94$. This choice is arbitrary, however similar results are obtained using other test functions. Using third order moving least squares (with *SingularityTolerance* $= 0.2$) on the mesh shown in Fig. 12, the normalised maximum errors may be plotted for various values of ϵ. This is shown in Fig. 8. Based on this graph, and similar data for other meshes and test functions, a value of $\epsilon = 1.4$ was chosen. Although this value is slightly higher than the optimum value in this case, it is prudent to make such a selection, as the errors increase much more quickly if ϵ is too small rather than too large, and some allowance must be made for variations in the effects of ϵ when different meshes are used. Note also that this choice may not be anywhere near optimal for moving least squares with a higher order polynomial basis, as the stencils are then larger. The data points on the periphery of the stencil, which are necessary for singularity avoidance, may then be effectively removed from the stencil by being assigned a weight which is very close to zero, if the same value of ϵ were used. It is therefore expected that ϵ will need to be larger for higher order moving least squares schemes, in order to extend the range of non-zero weights.

4.1.3. The importance of removing singularities

The importance of detecting and correcting singularities may be demonstrated by reducing the value of *SingularityTolerance* towards zero, thus allowing stencils to be built which are very close to being singular. It is expected that test functions will then be interpolated poorly. This may also serve as a basis for tuning the *SingularityTolerance* parameter.

To that end, (42) was used with $N = 9.94$. This choice is arbitrary, however similar results are obtained using other test functions. Using third order moving least squares (with $\epsilon = 1.4$) on the mesh shown in Fig. 12, the normalised maximum errors may be plotted for various values of *SingularityTolerance*. This is shown in Fig. 9. It may be seen that the errors increase dramatically as *SingularityTolerance* is reduced, indicating the presence of undetected singular stencils. The errors are so large, that a logarithmic scale was required for plotting the errors in the derivatives. Such enormous errors would spell disas-

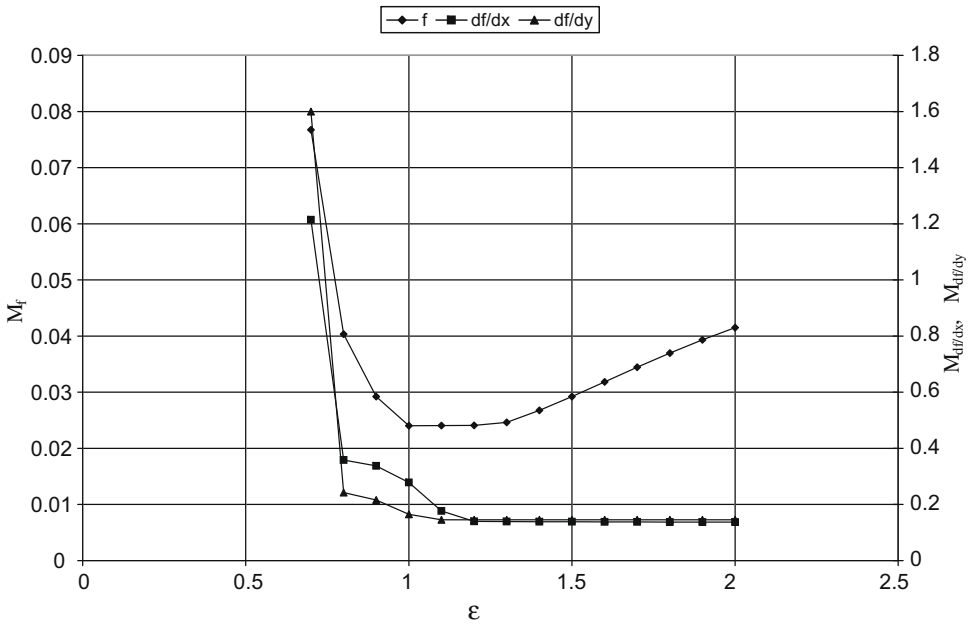


Fig. 8. Effect of shape parameter ϵ on interpolation errors.

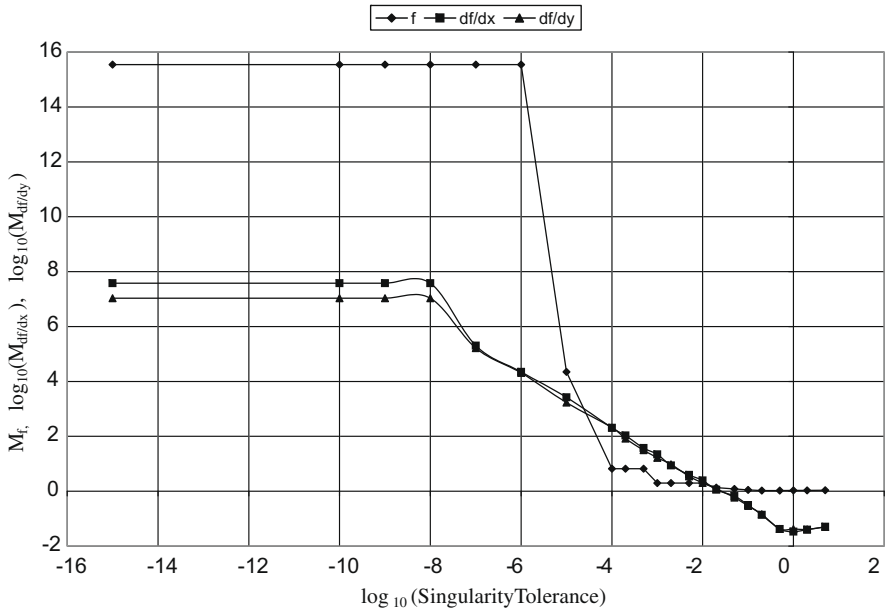


Fig. 9. Effect of *SingularityTolerance* on interpolation errors.

ter for any numerical simulation which uses the moving least squares scheme, most likely causing an instability in the first few time steps. Thus reliable detection of singular stencils is crucial.

Optimising *SingularityTolerance* on the basis of Fig. 9 alone would suggest a value of about 1, since this is where the errors reach a minimum. However, it is also important to consider the sizes of the resulting stencils, since these are strongly affected by the value of the parameter and larger stencils resulting in a more computationally expensive scheme. Fig. 10 shows the effect of *SingularityTolerance* on stencil size, again for the mesh shown in Fig. 12. As expected, increasing the parameter results in larger stencils, since stencils which are further from being singular will be detected as singular and thus expanded. It is clear that the stencil size increases rapidly once the parameter exceeds about 0.2. On this basis, a value of *SingularityTolerance* $= 0.2$ was selected, since this provides a reasonable compromise between reducing the errors and keeping the stencils small.

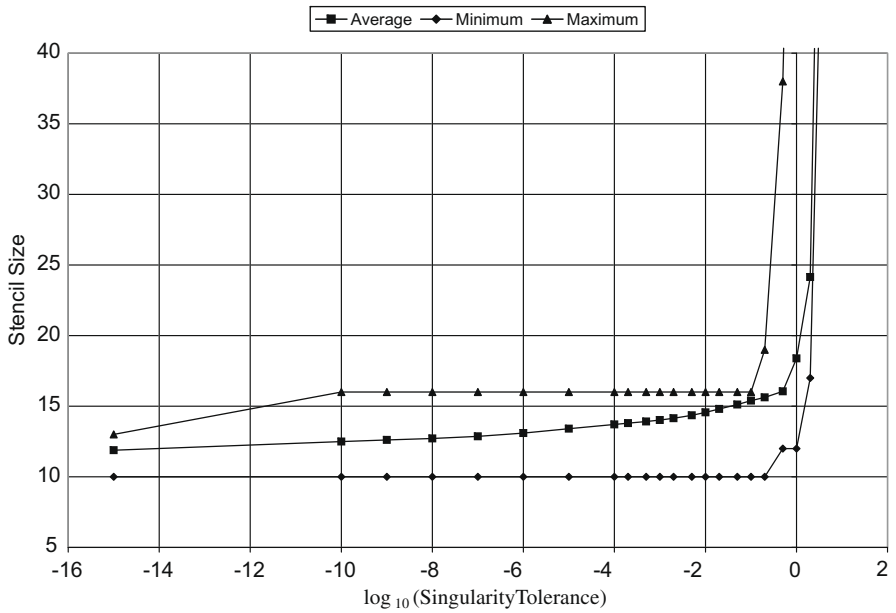


Fig. 10. Effect of *SingularityTolerance* on stencil size.

4.2. The benefits of using the improved stencil building algorithm

An effective stencil building algorithm for moving least squares must reliably detect and correct singularities, whilst building stencils which are small and therefore efficient to use. Existing methods for singularity avoidance, such as setting a uniform stencil size which is significantly larger than the theoretical minimum required, perform poorly on both counts. Reliability is hindered by the uncertainty as to whether a given stencil size will be large enough to avoid singularities at all locations on all possible meshes, regular and irregular. Stencil sizes are excessively large, due to the imposition of the stencil size required for the worst location in a mesh in all locations. The singularity detection theory presented in this paper solves both these problems, since individual stencils may then be expanded until they are known to be non-singular, which may be determined reliably. The value of this approach may be demonstrated by comparing the maximum stencil size obtained to the average; it is not uncommon for the maximum stencil size to be significantly larger than the average, meaning that the imposition of a uniform stencil size would have increased the stencil sizes significantly.

The methodologies outlined in this paper also stipulate how the data points added to the stencil to remove a known singularity may be chosen most efficiently (Algorithm 1). The benefits of this selection method are demonstrated in Table 1, where the optimal data point selection method (Algorithm 1) is compared to random data point selection and anti-optimal data point selection (where Algorithm 1 is modified to make the worst selection instead of the best). Various polynomial orders are used, but in all cases $\epsilon = 1.4$ and singularities were detected with *SingularityTolerance* $= 0.2$. The mesh used is shown in Fig. 12. The results show that, at each polynomial order studied, the average stencil obtained using optimal selection is smaller than that obtained using random selection, which is in turn smaller than that obtained by anti-optimal selection. This shows that Algorithm 1 does indeed result in smaller stencils, although the benefits are small. Note, however, that all the schemes involved in this comparison gain the benefits of reliable singularity detection, as discussed in the last paragraph. Without this, all the stencils in the mesh would need to be as large as the maximum stencil size (for the optimal scheme).

A more significant difference may be obtained when locality enforcement is removed; that is, when it is no longer required that all data points at a given connectivity depth are added before any at a higher connectivity depth are added.

Table 1
Moving least squares stencil sizes on the mesh shown in Fig. 12. Algorithms compared are optimal data point addition (Algorithm 1), random data point addition and anti-optimal data point addition. All stencil building algorithms enforce locality and symmetry.

Stencil order	Theoretical minimum size	Optimal selection			Random selection			Anti-optimal selection		
		Min	Mean	Max	Min	Mean	Max	Min	Mean	Max
1	3	3	5.96	6	3	6.68	8	3	7.81	8
2	6	6	7.97	9	6	7.99	9	6	8.00	9
3	10	10	15.62	19	10	16.33	21	10	17.55	23
4	15	15	18.05	44	15	18.21	45	15	18.34	47

Table 2
Moving least squares stencil sizes on the mesh shown in Fig. 12. Algorithms compared are optimal data point addition (Algorithm 1), random data point addition and anti-optimal data point addition, except that all stencil building algorithms are modified to enforce symmetry but not locality.

Stencil order	Theoretical minimum size	Optimal selection			Random selection			Anti-optimal selection		
		Min	Mean	Max	Min	Mean	Max	Min	Mean	Max
1	3	3	5.97	6	3	6.85	16	3	38.53	153
2	6	6	9.58	10	6	10.11	30	6	51.91	303
3	10	10	14.31	18	10	17.12	46	10	87.55	458
4	15	15	20.99	27	15	23.69	80	15	131.07	618

Table 2 compares the three schemes from Table 1, but each with this modification. It may be seen that, without locality enforcement, the anti-optimal data selection builds extremely large stencils, while the optimal data selection still builds small stencils. This demonstrates that the optimal data point selection method does indeed distinguish correctly between data points which are appropriate and inappropriate for removing singularities. However, most of the benefit of using the optimal data point selection method may also be obtained by the use of locality enforcement on its own.

4.3. Use of the scheme in a convection–diffusion solver

The moving least squares scheme was used in a simulation developed for solving the two-dimensional convection–diffusion equation, which may be written as

$$\frac{\partial f}{\partial t} = -u\frac{\partial f}{\partial x} - v\frac{\partial f}{\partial y} + \mu\left(\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}\right), \tag{47}$$

where u and v are the components of the convection velocity field in the x and y directions and μ is the diffusion coefficient. f is the variable being convected, which could represent temperature, concentration, etc according to the application. A finite volume scheme was used, where the value of f is tracked at the centroid of each mesh element. The equation used to obtain the time derivative of f in each element (with area A_{element} and its centroid at (x_e, y_e)) may be obtained by integrating (47) over the element's area and applying the divergence theorem. This yields

$$A_{\text{element}}\left.\frac{\partial f}{\partial t}\right|_{(x_e,y_e)} = \sum_{p\in\text{element edges}} L_p\left[\mu\left.\frac{\partial f^*}{\partial n}\right|_{(x_p,y_p)} - f^*(x_p,y_p)u_{n,p}\right], \tag{48}$$

where the index p is used as a subscript to denote quantities evaluated at the midpoint of bounding edge p and L_p is the length of edge p. The derivative with respect to n indicates a spatial derivative normal to edge p (outwards from the element). Likewise, $u_{n,p}$ is the projection of the local convection velocity in this direction. The asterisk indicates that the quantity involved needs to be obtained using interpolation (except in the special cases where it is known directly from boundary conditions). All interpolations were done using the third order moving least squares scheme, as described previously (with *SingularityTolerance* = 0.2 unless otherwise noted). This provides a robust test of the scheme's accuracy and applicability.

Time stepping was done using 4th order Runge–Kutta, based on (48). The size of the time increment (Δt) was determined using a combination of Fourier and Courant–Friedrichs–Lewy numbers. Let the distance between data points i and j be $r_{i,j}$, so that

$$r_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}. \tag{49}$$

The time step selected is then

$$\Delta t = \min\left\{\frac{r_{i,j}}{\frac{\sqrt{u^2+v^2}}{Cfl} + \frac{\mu}{Fo\cdot r_{i,j}}}\right\}, \tag{50}$$

where *Cfl* is the Courant–Friedrichs–Lewy number and *Fo* is the Fourier number. For problems involving a spatially varying flow field, u and v are taken as averages of their respective values at data points i and j. It should be noted that when $u = v = 0$, this expression collapses to the usual definition of the Fourier number for pure diffusion problems. Also, when $\mu = 0$, the expression collapses to

$$\Delta t = \min\left\{\frac{Cfl\cdot r_{i,j}}{\sqrt{u^2+v^2}}\right\}, \tag{51}$$

which is a variant on the usual definition of the Courant–Friedrichs–Lewy number. Values of $Cfl = 0.5$ and $Fo = 0.5$ were found to be appropriate for running stable simulations, and were used for the simulations in this paper.

By way of example, the two-dimensional convection–diffusion equation was solved on the domain illustrated in Fig. 11, filled with an irregular mesh of quadrilaterals (see Fig. 12). The applied boundary conditions are shown, with the initial condition given by

$$f = f_0 e^{-\frac{(x-x_0)^2+(y-y_0)^2}{2\sigma_0^2}}, \tag{52}$$

where $x_0 = y_0 = 0.5$ m, $f_0 = 1$ and $\sigma_0 = 0.1$ m in this example. Convection velocities of $u = v = 1$ m s^{-1} were used, while the parameter μ was assigned a value of 3.0×10^{-3} m^2 s^{-1}. This problem has the analytical solution [18]

$$f = \frac{\sigma_0^2 f_0}{\sigma_0^2 + 2\mu t} e^{-\frac{(x-x_0-ut)^2+(y-y_0-vt)^2}{2(\sigma_0^2+2\mu t)}}, \tag{53}$$

which is useful for verification purposes. The numerical and analytical solutions at times $t = 1$ s and $t = 2$ s are shown in Figs. 13 and 14. It may be seen that the numerical results correspond closely to the analytical solution, which indicates that the solver is accurate. Note the presence of a minor amount of spurious noise at time $t = 1$ s, which later disappears. This noise is visible only because the contour level of zero is shown, and the numerical solution dips very slightly negative inside that small circle. This occurs because the simulation was run at a Peclet number which is close to the stability limit. In contrast, Fig. 15 shows the same simulation at time $t = 1$ s, when *SingularityTolerance* $= 0.05$ instead of 0.2. With this value, the stencils built are allowed to be much closer to being singular, and so the numerical error is greatly increased. Consequently, the numerical solution becomes unstable, producing highly inaccurate results in a region containing one or more stencils that are nearly singular.

4.4. Use in an incompressible Navier–Stokes solver

As an additional test of the robustness of the moving least squares scheme, an incompressible Navier–Stokes solver was developed which used the scheme for the necessary interpolations. The two-dimensional, incompressible Navier–Stokes equations are

$$\begin{aligned}
\rho\left(\frac{\partial u}{\partial t} + \frac{\partial(u^2)}{\partial x} + \frac{\partial(uv)}{\partial y}\right) &= -\frac{\partial p}{\partial x} + \mu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right), \\
\rho\left(\frac{\partial v}{\partial t} + \frac{\partial(vu)}{\partial x} + \frac{\partial(v^2)}{\partial y}\right) &= -\frac{\partial p}{\partial y} + \mu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right),
\end{aligned} \tag{54}$$

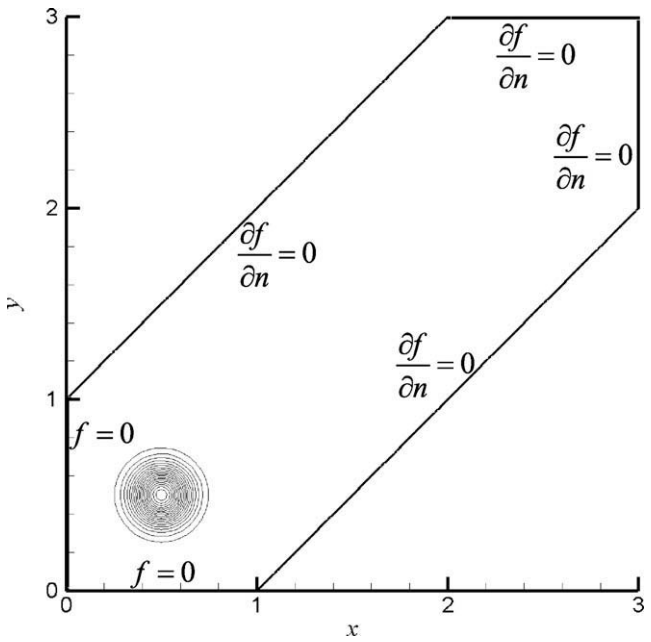


Fig. 11. Initial conditions and boundary conditions for propagation of a Gaussian pulse. The initial conditions are shown as contours, the outermost contour of the pulse being at 0.05 and increasing at intervals of 0.05 inwards.

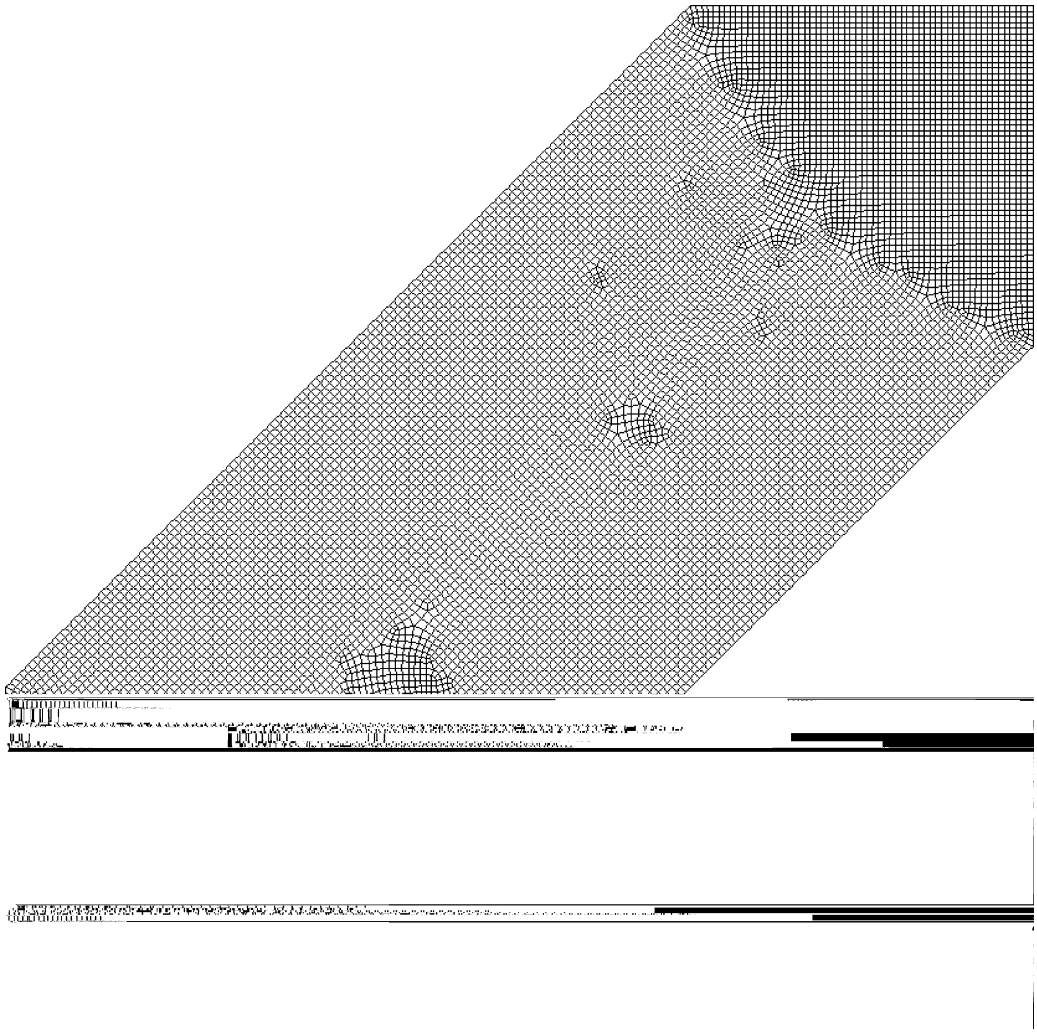


Fig. 12. Mesh used for simulating the convection–diffusion of a Gaussian pulse.

where t is the time, μ is the dynamic viscosity, ρ is the density, p is the pressure and u and v are the velocity components in the x and y spatial directions respectively. The continuity equation is also required, which is

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}=0. \tag{55}$$

Since the continuity equation is difficult to apply in incompressible solvers, the preferred equation to use is the Poisson equation [9]. This is derived from the Navier–Stokes equations, and is

$$\frac{\partial D}{\partial t}+\nabla\cdot[D(u,v)]=\frac{\mu}{\rho}\nabla^2 D-\frac{1}{\rho}\nabla^2 p-\nabla\cdot\left[(u,v)\frac{\partial(u,v)}{\partial(x,y)}\right], \tag{56}$$

where

$$D=\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y} \tag{57}$$

is the divergence and

$$\frac{\partial(u,v)}{\partial(x,y)}=\begin{bmatrix}\frac{\partial u}{\partial x} & \frac{\partial v}{\partial x}\\ \frac{\partial u}{\partial y} & \frac{\partial v}{\partial y}\end{bmatrix} \tag{58}$$

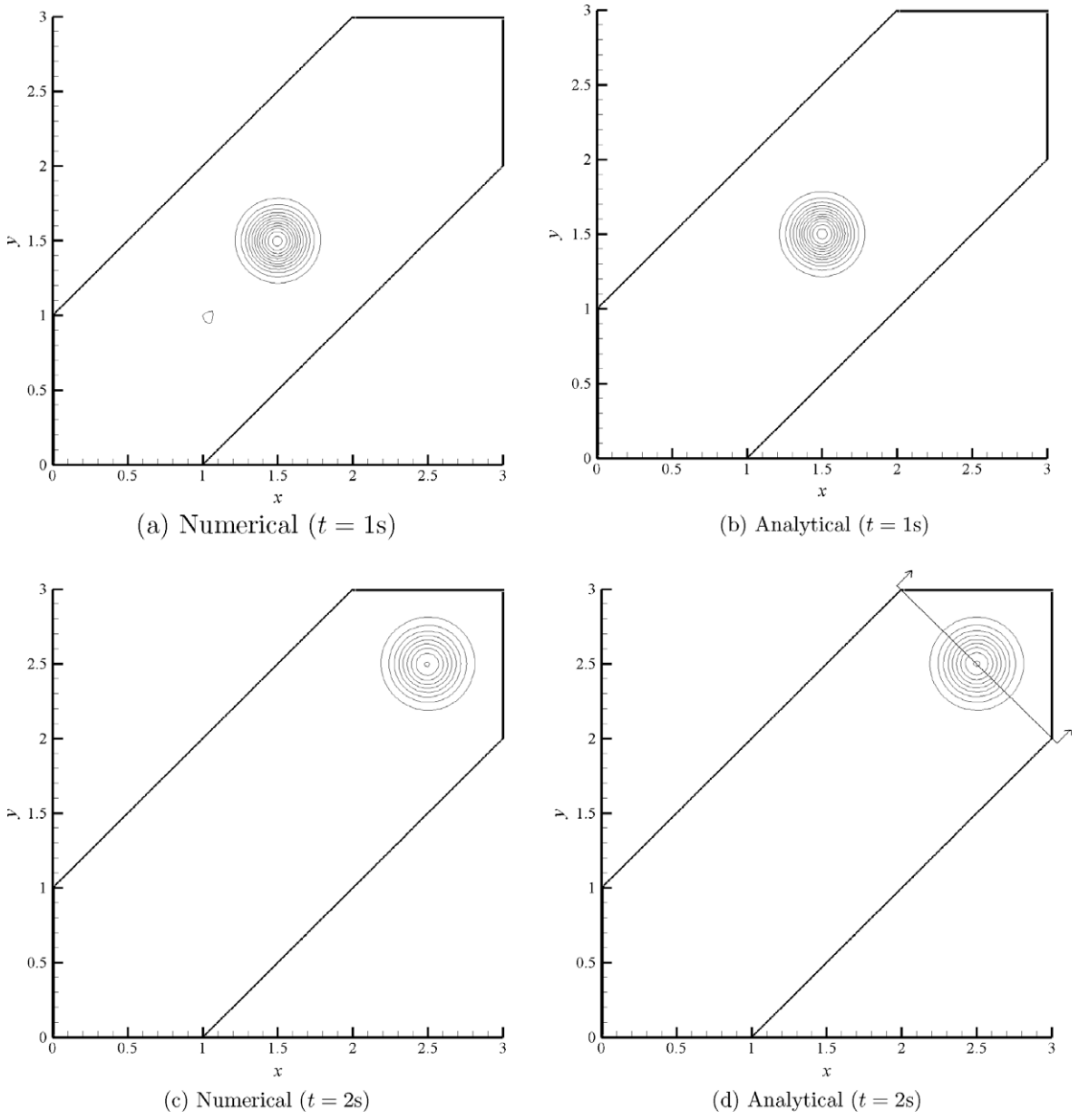


Fig. 13. Propagation of a Gaussian pulse. The outermost contour of the pulses are at 0.05 and the contours increase in intervals of 0.05 inwards. The contour level of 0 appears only in the numerical solution at time $t = 1$ s, as there is a very small level of numerical noise which makes the solution very slightly negative in one small region. The line drawn on Fig. 13(d) indicates the section used for the following figure.

is the transpose of the velocity gradient tensor. (Vector notation is used in the above equation for the sake of brevity.) It should be noted that the above equation is in effect a scalar transport equation for divergence, including terms for the convection, diffusion and production of divergence. In practice many of the terms of this equation are insignificant, and so it may be simplified to

$$\frac{\partial D}{\partial t} = -\frac{1}{\rho}\nabla^2 p. \tag{59}$$

The incompressible Navier–Stokes equations may be solved using the fractional step method. This method allows the velocity and pressure fields to be partially decoupled during a time step. First, an intermediate velocity field $(\hat{u}, \hat{v})$ is obtained implicitly by means of the Crank–Nicholson method applied to the Navier–Stokes equations. Note that the pressure gradient term is not treated implicitly, since this limits the variables which must be solved for to the velocity components. Treated in this manner, the x direction Navier–Stokes equation yields

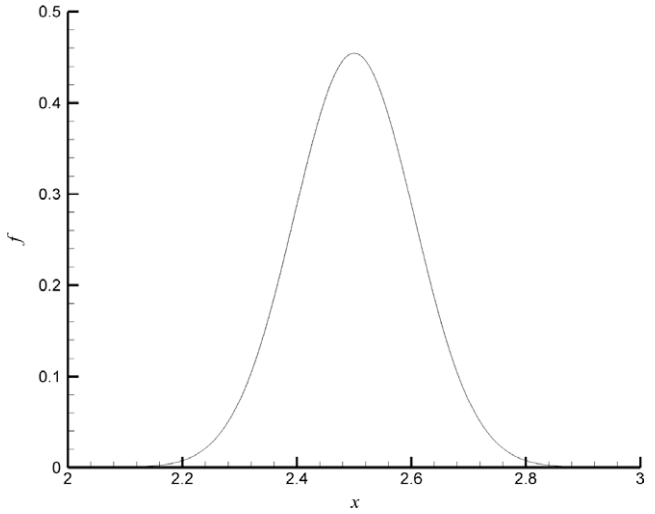


Fig. 14. Propagation of a Gaussian pulse, showing a cross section of the surface at time $t = 2$ s. The dashed line is the solution from the solver while the solid line is the theoretical solution. The two lines lie over the top of each other, and so cannot be distinguished at this scale. The location of the section is shown in Fig. 13(d).

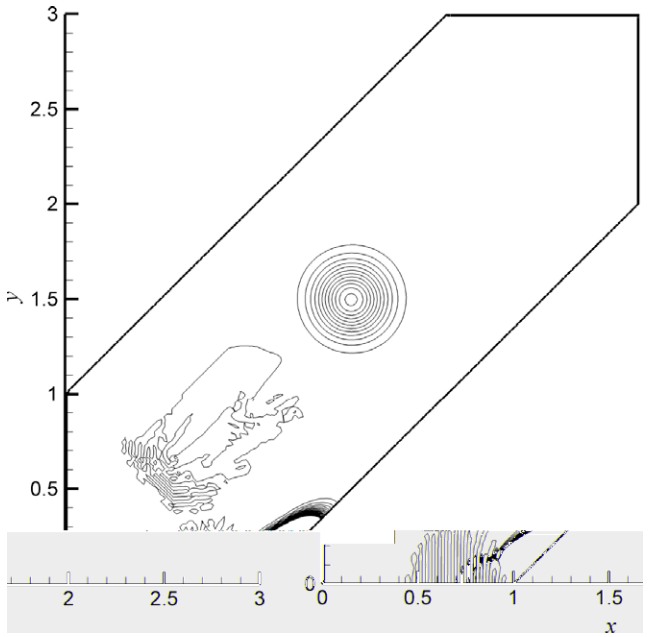


Fig. 15. Unstable propagation of a Gaussian pulse, when *SingularityTolerance* = 0.05, at time $t = 1$ s. The outermost contour of the pulse is at 0.05 and the contours increase in intervals of 0.05 inwards. Although the unstable region contains a wider range of values, for reasons of clarity the smallest contour shown is 0 and the largest is 1.

$$\rho\left(\frac{\hat{u}-u^n}{\Delta t}+\frac{1}{2}\cdot\left(\frac{\partial(\hat{u}\hat{u}+u^n u^n)}{\partial x}+\frac{\partial(\hat{u}\hat{v}+u^n v^n)}{\partial y}\right)\right)=-\frac{\partial p^n}{\partial x}+\frac{\mu}{2}\left(\frac{\partial^2(\hat{u}+u^n)}{\partial x^2}+\frac{\partial^2(\hat{u}+u^n)}{\partial y^2}\right), \tag{60}$$

where u^n, v^n and p^n represent the velocity components and pressure which exist at the start of the time step (which is of length Δt). The non-linear implicit terms prevent this equation being solved directly by a linear solver, so they must be simplified. It may be shown [11] that this equation may be rewritten as

$$\rho\left(\frac{\hat{u}-u^n}{\Delta t}+\frac{1}{2}\cdot\left(\frac{\partial(2\hat{u}u^n)}{\partial x}+\frac{\partial(\hat{u}v^n+u^n\hat{v})}{\partial y}\right)\right)=-\frac{\partial p^n}{\partial x}+\frac{\mu}{2}\left(\frac{\partial^2(\hat{u}+u^n)}{\partial x^2}+\frac{\partial^2(\hat{u}+u^n)}{\partial y^2}\right), \tag{61}$$

with a loss of accuracy which is of order Δt^2. Similarly, the y direction Navier–Stokes equation yields

$$\rho\left(\frac{\hat{v}-v^n}{\Delta t}+\frac{1}{2}\cdot\left(\frac{\partial(\hat{v}u^n+v^n\hat{u})}{\partial x}+\frac{\partial(2\hat{v}v^n)}{\partial y}\right)\right)=-\frac{\partial p^n}{\partial y}+\frac{\mu}{2}\left(\frac{\partial^2(\hat{v}+v^n)}{\partial x^2}+\frac{\partial^2(\hat{v}+v^n)}{\partial y^2}\right). \tag{62}$$

Next the initial pressure field (p^n) is applied in reverse to the $(\hat{u},\hat{v})$ velocity field, to obtain the (u^*,v^*) velocity field.

$$\begin{aligned}\frac{u^*-\hat{u}}{\Delta t}&=\frac{1}{\rho}\cdot\frac{\partial p^n}{\partial x},\\ \frac{v^*-\hat{v}}{\Delta t}&=\frac{1}{\rho}\cdot\frac{\partial p^n}{\partial y}\end{aligned} \tag{63}$$

Next the final pressure field is calculated, so that the pressure gradient is exactly what is required in order to force the divergence of the final velocity field to zero. This is done using the simplified Poisson equation, giving

$$\frac{D^*}{\Delta t}=\frac{1}{\rho}\nabla^2 p^{n+1}, \tag{64}$$

where D^* is the divergence of the (u^*,v^*) velocity field and p^{n+1} is the final pressure field. Lastly, the final pressure field is applied to the (u^*,v^*) velocity field, in order to force the divergence of the velocity field to zero. (Since the pressure field was applied once forwards and once in reverse in the earlier steps, the net effect of all three applications of the pressure field is a single forward application, as required.) Thus

$$\begin{aligned}\frac{u^{n+1}-u^*}{\Delta t}&=-\frac{1}{\rho}\cdot\frac{\partial p^{n+1}}{\partial x},\\ \frac{v^{n+1}-v^*}{\Delta t}&=-\frac{1}{\rho}\cdot\frac{\partial p^{n+1}}{\partial y},\end{aligned} \tag{65}$$

where u^{n+1} and v^{n+1} are the component of the velocity field at the end of the time step. The fractional step method has two implicit steps ((61) and (62) for the velocity field, (64) for the pressure field) and two explicit steps ((63) and (65)).

Since a finite volume solver was developed, it was necessary to use the divergence theorem to convert the fractional step equations into finite volume form, then apply these to a control volume. (In this two-dimensional solver, control volumes are area elements of the mesh.) If this is done, and the resulting integrals discretised for the straight line segments of each element, then (61) to (65) become

$$\frac{\rho A_{\text{C.V.}}}{\Delta t}(\hat{u}_{\text{C.V.}}-u^n_{\text{C.V.}})+\frac{\rho}{2}\sum_{\substack{i\in\text{C.V.}\\ \text{edges}}}\left[(2u_i^n l_{i,x}+v_i^n l_{i,y})\hat{u}_i+u_i^n l_{i,y}\hat{v}_i\right]=-\sum_{\substack{i\in\text{C.V.}\\ \text{edges}}}\left[p_i^n l_{i,x}\right]+\frac{\mu}{2}\sum_{\substack{i\in\text{C.V.}\\ \text{edges}}}\left[\frac{\partial(\hat{u}_i+u_i^n)\partial x}{l}_{i,x}+\frac{\partial(\hat{u}_i+u_i^n)}{\partial y}l_{i,y}\right], \tag{66}$$

$$\frac{\rho A_{\text{C.V.}}}{\Delta t}(\hat{v}_{\text{C.V.}}-v^n_{\text{C.V.}})+\frac{\rho}{2}\sum_{\substack{i\in\text{C.V.}\\ \text{edges}}}\left[v_i^n l_{i,x}\hat{u}_i+(2v_i^n l_{i,y}+u_i^n l_{i,x})\hat{v}_i\right]=-\sum_{\substack{i\in\text{C.V.}\\ \text{edges}}}\left[p_i^n l_{i,y}\right]+\frac{\mu}{2}\sum_{\substack{i\in\text{C.V.}\\ \text{edges}}}\left[\frac{\partial(\hat{v}_i+v_i^n)}{\partial x}l_{i,x}+\frac{\partial(\hat{v}_i+v_i^n)}{\partial y}l_{i,y}\right], \tag{67}$$

$$\begin{aligned}u^*_{\text{C.V.}}&=\hat{u}_{\text{C.V.}}+\frac{\Delta t}{\rho A_{\text{C.V.}}}\sum_{\substack{i\in\text{C.V.}\\ \text{edges}}}\left[p_i^n l_{i,x}\right],\\ v^*_{\text{C.V.}}&=\hat{v}_{\text{C.V.}}+\frac{\Delta t}{\rho A_{\text{C.V.}}}\sum_{\substack{i\in\text{C.V.}\\ \text{edges}}}\left[p_i^n l_{i,y}\right],\end{aligned} \tag{68}$$

$$\sum_{\substack{i\in\text{C.V.}\\ \text{edges}}}\left[\frac{\partial p_i^{n+1}}{\partial x}l_{i,x}+\frac{\partial p_i^{n+1}}{\partial y}l_{i,y}\right]=\frac{\rho}{\Delta t}\sum_{\substack{i\in\text{C.V.}\\ \text{edges}}}\left[u_i^* l_{i,x}+v_i^* l_{i,y}\right], \tag{69}$$

$$\begin{aligned}u^{n+1}_{\text{C.V.}}&=u^*_{\text{C.V.}}-\frac{\Delta t}{\rho A_{\text{C.V.}}}\sum_{\substack{i\in\text{C.V.}\\ \text{edges}}}\left[p_i^{n+1} l_{i,x}\right],\\ v^{n+1}_{\text{C.V.}}&=v^*_{\text{C.V.}}-\frac{\Delta t}{\rho A_{\text{C.V.}}}\sum_{\substack{i\in\text{C.V.}\\ \text{edges}}}\left[p_i^{n+1} l_{i,y}\right],\end{aligned} \tag{70}$$

where $A_{\text{C.V.}}$ is the area of the element being used as a control volume, the C.V. subscript indicates a velocity taken at the centroid of the element and the i subscript indicates a quantity which is taken at the midpoint of one of the element's edges. If $(n_{i,x},n_{i,y})$ is an outward facing unit vector normal to edge i and l_i is the length of edge i, then $l_{i,x}=n_{i,x}l_i$ and $l_{i,y}=n_{i,y}l_i$.

Note that the above equations incorporate the value and spatial derivatives of both pressure and velocity on the edge midpoints. These values and derivatives may be expressed as a weighted sum of the values at the element centroids and known values on the boundaries (from boundary conditions), using the interpolation scheme explained earlier. When this is done, (66) and (67) then involve implicit references only to velocities at the element centroids. Eqs. (66) and (67) may thus be used to build a linear system of the form

$$\mathbf{A}\mathbf{x} = \mathbf{q}, \tag{71}$$

where $\mathbf{A}$ is a column-sparse square matrix and $\mathbf{x}$ is a column vector containing the $\hat{u}$ and $\hat{v}$ values for each element centroid in the domain, which must be solved for. ($\mathbf{q}$ is a column vector of the same length as $\mathbf{x}$.) Likewise, (69) then involves implicit references only to pressures at the element centroids. Eq. (69) may thus be used to build a linear system of form (71), where $\mathbf{A}$ is again a column-sparse square matrix but $\mathbf{x}$ is this time a column vector containing the p^{n+1} values for each element centroid in the domain, which must be solved for.

The two linear systems were solved using the biconjugate gradient stabilised method. [1] In order to conserve memory, the matrices for the linear systems were stored in sparse format. No matrix preconditioner was used. Let r_i be the residual for the ith row of the linear system (i.e. $r_i = \sum_j[(\mathbf{A})_{ij}(\mathbf{x})_j] - (\mathbf{q})_i$). Convergence for the linear system derived from (66) and (67) was determined to occur when

$$\max[|r_i|] \leqslant \varepsilon_M \rho V^2 \sqrt{A_{\min}}, \tag{72}$$

where $\varepsilon_M = 10^{-4}$, V is a parameter specifying the typical velocity scale of the problem and $A_{\min}$ is the area of the smallest element of the mesh. Likewise, convergence for the linear system derived from (69) was determined to occur when

$$\max[|r_i|] \leqslant \frac{\varepsilon_P V \sqrt{A_{\min}}}{\Delta t}, \tag{73}$$

where $\varepsilon_P = 10^{-4}$.

The time step size Δt was selected automatically using

$$\Delta t = \frac{Cfl \cdot D_{\min}}{V}, \tag{74}$$

where $Cfl = 0.2$ is the Courant–Friedrichs–Lewy number and $D_{\min}$ is the minimum distance between data points in the mesh.

The incompressible Navier–Stokes solver was verified using the lid-driven cavity flow problem, which is a standard benchmark. This is illustrated schematically in Fig. 16, which also shows the applied boundary conditions. Fluid parameters of $\rho = 1000$ kg.m^{-3} and $\mu = 1$ kg m^{-1} s^{-1} were specified, giving a Reynolds number of 1000. The mesh used (see Fig. 17) contained both regular and irregular regions, so that the interpolation scheme (third order moving least squares) could be put to the test. The streamlines for the steady state solution are shown in Fig. 18, which also shows the streamlines obtained by Erturk et al. [4] for the same problem. The flow patterns obtained are extremely similar, the only differences between the two plots being in the choice of streamline origins. As a more rigourous comparison, the steady state v velocity along the horizontal centreline and steady state u velocity along the vertical centreline are plotted in Fig. 19. These are compared with the results of simulations by Kim and Choi [11] and Ghia et al. [5] on the same figure, and the results are clearly similar. This demonstrates the accuracy of the solver, which in turn implies that the interpolation scheme used is suitably robust and accurate.

In order to show the importance of singularity avoidance, the above simulation was re-run with *SingularityTolerance* = 0.05 instead of 0.2. This allows stencils to be built which are nearly singular, causing large numerical errors. In turn, this results in instabilities in several places on the mesh, which ultimately grow large enough to prevent successful convergence of the iterative algorithm for solving (71). This occurred in only 58 time steps, which demonstrates just how crucial it is to avoid singularities. The results for this simulation are highly inaccurate, and are omitted for brevity.

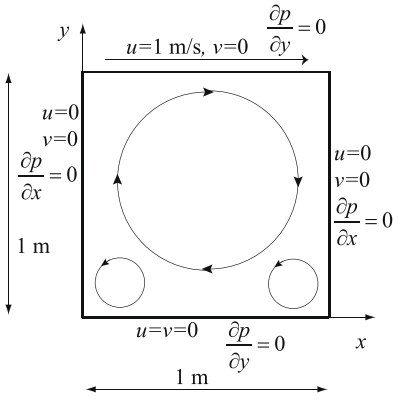


Fig. 16. Schematic of the cavity flow problem, including the boundary conditions which were applied in this study. The vortices shown are those which exist in the steady state solution at $Re = 1000$. Additional vortices exist at higher Reynolds numbers.

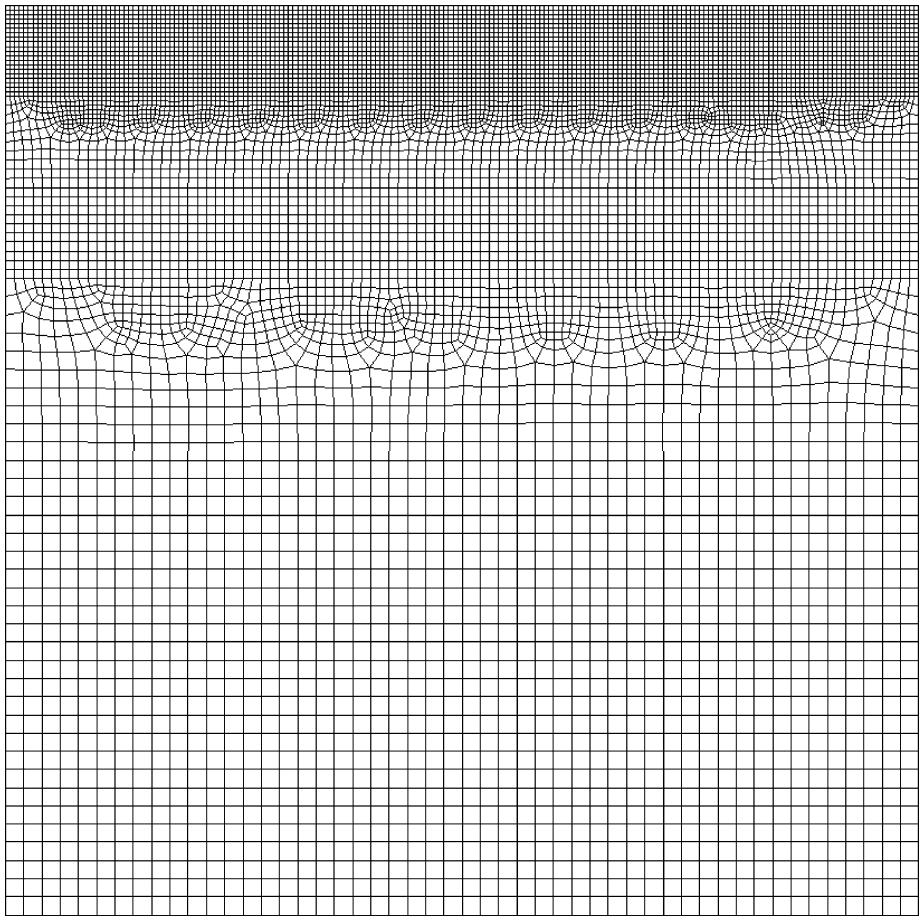


Fig. 17. Mesh used for simulating the cavity flow problem.

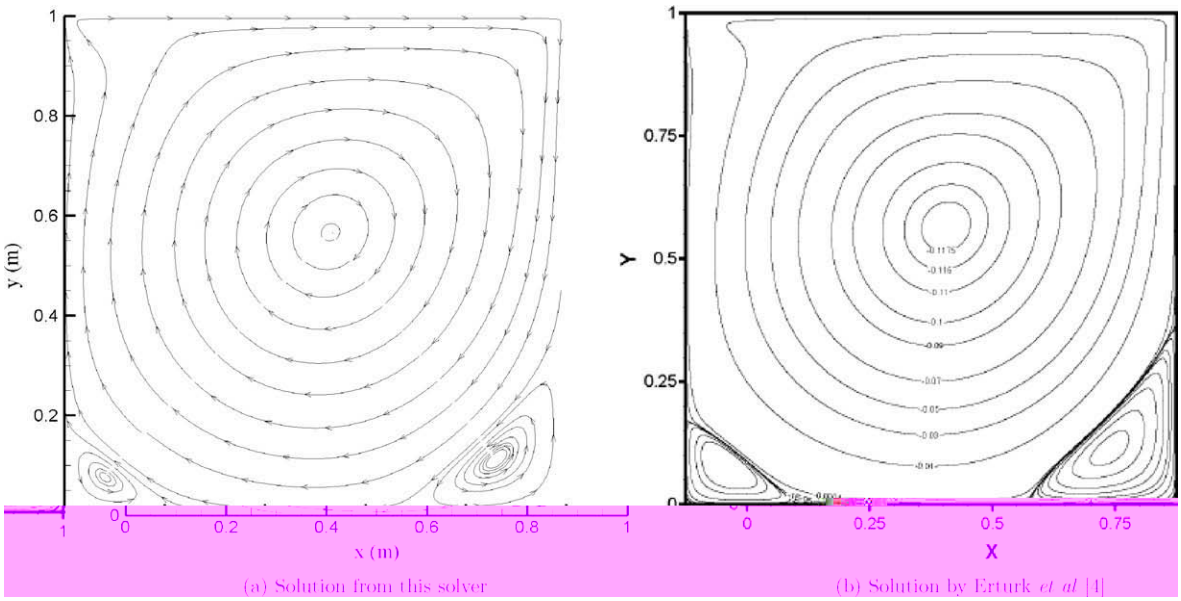
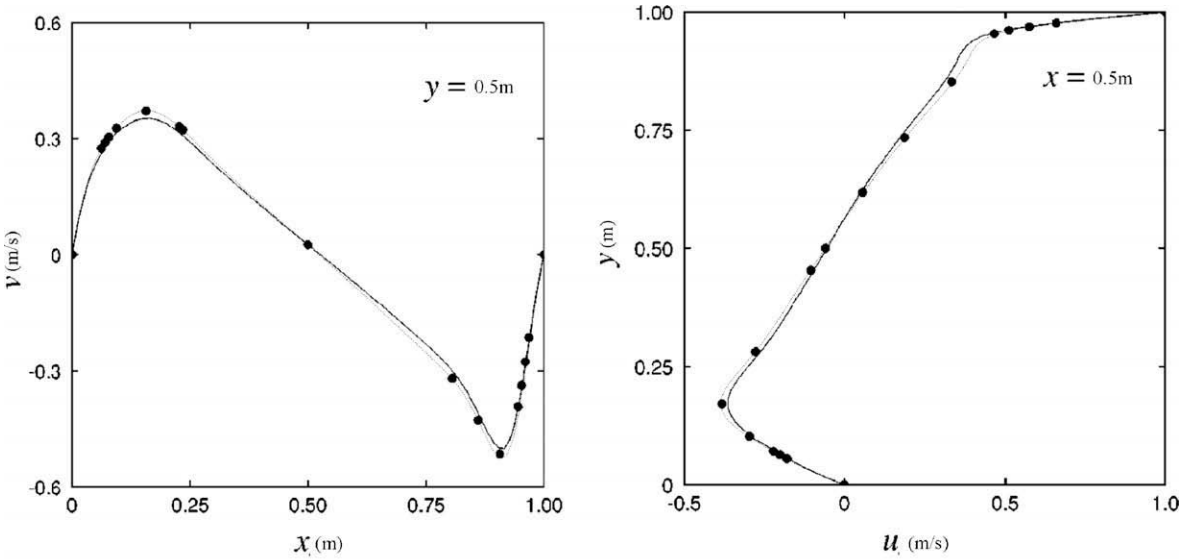


Fig. 18. Streamlines for the steady state solution to the cavity flow problem, at $Re = 1000$. Note that the streamlines for the plots were chosen in order to highlight the flow topology, and are not intended to correspond to regularly spaced values of the stream function.



(a) v along a horizontal cut at $y = 0.5$m (b) u along a vertical cut at $x = 0.5$m

Fig. 19. Velocity sections for the steady state solution to the cavity flow problem, at $Re = 1000$. The dashed lines are the results obtained from this study, while the solid lines are the results of Kim and Choi [11]. The large dots are the results of Ghia et al. [5].

5. Conclusion

In this paper, a moving least squares interpolation scheme was presented, for use with unstructured meshes. The precise conditions under which singularities occur were identified, and a method by which this theory may be extended to more general moving least squares schemes was discussed. Moreover, the singularity theory was used to create an algorithm which may be used to build stencils which are guaranteed to be non-singular (as well as being as close as possible to symmetric). Example stencils produced by this algorithm were given. The moving least squares scheme, based on the stencil building algorithm, was then applied in a convection–diffusion equation solver and incompressible Navier–Stokes solver, and was found to be robust and accurate.

Acknowledgment

The authors would like to acknowledge the support of the Australian Research Council for this research through Grant DP0556098.

References

[1] R. Barrett, M.W. Berry, T.F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, H. van der Vorst, Templates for the solution of linear systems: building blocks for iterative methods, Society for Industrial and Applied Mathematics (1993).
[2] A. Bodin, J. Ma, X.J. Xin, P. Krishnaswami, A meshless integral method based on regularized boundary integral equation, Computer Methods in Applied Mechanics and Engineering 195 (2006) 6258–6286.
[3] H. Desimone, S. Urquiza, H. Arrieta, E. Pardo, Solution of Stokes equations by moving least squares, Communications in Numerical Methods in Engineering 14 (1998) 907–920.
[4] E. Erturk, T.C. Corke, C. Gokcol, Numerical solutions of 2-D steady incompressible driven cavity flow at high Reynolds numbers, International Journal for Numerical Methods in Fluids 48 (2005) 747–774.
[5] U. Ghia, K.N. Ghia, C.T. Shin, High-Re solutions for incompressible flow using the Navier–Stokes equations and a multigrid method, Journal of Computation Physics 48 (1982) 387–411.
[6] G.H. Golub, C.F. Van Loan, Matrix Computations, The John Hopkins University Press, 1996.
[7] S.I. Grossman, Elementary Linear Algebra, Saunders College Publishing, Berlin, 1994.
[8] R.L. Hardy, Multiquadric equations of topography and other irregular surfaces, Journal of Geophysical Research 76 (1971) 1905–1915.
[9] K.A. Hoffmann, Computational fluid dynamics, Engineering Education System (2000).
[10] B. Jiang, On the least squares method, Computer Methods in Applied Mechanics and Engineering 152 (1998) 239–257.
[11] D. Kim, H. Choi, A second-order time-accurate finite volume method for unsteady incompressible flow on hybrid unstructured grids, Journal of Computational Physics 162 (2000) 411–428.
[12] P. Lancaster, K. Salkauskas, Surfaces generated by moving least squares methods, Mathematics of Computation 37 (1981) 141–158.
[13] Y. Lipman, D. Cohen-Or, D. Levin, Data-dependent MLS for faithful surface approximation, Eurographics Symposium on Geometry Processing, 2007.
[14] H. Netuzhylov, Meshfree collocation solution of boundary value problems via interpolating moving least squares, Communications in Numerical Methods in Engineering 22 (2006) 893–899.
[15] C. Prax, H. Sadat, E. Dabboura, Evaluation of high order versions of the diffuse approximate meshless method, Applied Mathematics and Computation 186 (2007) 1040–1053.
[16] W. Schoenauer, T. Adolph, How WE Solve PDEs, Journal of Computational and Applied Mathematics 131 (2001) 473–492.

[17] D. Shepard, A two-dimensional interpolation function for irregularly spaced points, in: Proceedings of the 23rd National Conference, ACM, 1968, pp. 517–523.
[18] O. Shipilova, H. Haario, A. Smolianski, Particle transport method for convection problems with reaction and diffusion, International Journal for Numerical Methods in Fluids 54 (2007) 1215–1238.
[19] G.B. Wright, B.F. Fornberg, Scattered node compact finite difference-type formulas generated from radial basis functions, Journal of Computational Physics 212 (2006) 99–123.