16th Australasian Fluid Mechanics Conference
Crown Plaza, Gold Coast, Australia
2-7 December 2007

A Singularity-Avoiding Moving Least Squares Scheme for Two Dimensional Unstructured Meshes

S. K. M. Chenoweth[1] J. Soria[2] and A. Ooi[1]

[1]Department of Mechanical and Manufacturing Engineering
University of Melbourne, Victoria, 3010 AUSTRALIA

[2]Laboratory for Turbulence Research in Aerospace
& Laboratory for Turbulence Research in Aerospace and Combustion (LTRAC)
Department of Mechanical Engineering
Monash University, Victoria, 3800 AUSTRALIA

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 addressed this issue specifically. 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 then applied in the form of 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 used in a convection-diffusion equation solver, and the results of a number of simulations are presented.

Introduction

When creating numerical simulations that operate on data defined on an unstructured mesh, it is necessary to employ interpolation techniques in order to evaluate the functions at arbitrary points in the domain. Calculation of the spatial derivatives of a function, even at a point where the value of that function 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, however one of the more common ones is the moving least squares scheme.

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 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, rather than ones further away. Thus the stencil moves in order to be local to the interpolation point it belongs to; this is the reason for the word 'moving'.

In the standard moving least squares scheme, 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. 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. Thus the standard moving least squares scheme is properly called an approximation rather than interpolation scheme. However, since the errors of fit are known to be minimal for smooth data, the term 'interpolation' is used even though 'approximation' is more strictly correct.

The moving least squares scheme has been widely used for both pure interpolation / approximation problems (e.g. in image processing applications [1]) and in solving partial differential equations [2]. 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 in fact not invertible. Desimone et al [2], 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 [3], 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 [4] and Prax et al [5], and for a similar problem by Schoenauer and Adolph [6], who also observed the occurrence of singularities when the data points lie along straight lines. However, to the best knowledge of the authors, a general theory for predicting the occurrence of singularities in moving least squares has never been published.

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 way of predicting in advance which data points will be the best to add and how many new data points may be required) or, in the case of meshless methods, randomly moving each data point a small distance [5]. Using the methodology presented in this paper, it is possible to devise more reliable algorithms for detecting singular stencils and correcting them efficiently (i.e. with a minimal number of new data points added).

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. 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 (or a linear combination of these) at various scattered points in the local region. This amount of flexibility is necessary for handling the kinds of boundary conditions that are typically required when solving partial differential equations such as the convection diffusion equation [7]. In order to allow for such general boundary conditions, the following linear relationship is specified for each data point j

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

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 (possibly time varying) value. The factor L_j is necessary in order for all the terms in this expression to have the same units (which are the same as the units of f itself). With appropriate choices for the constants, this expression allows the interpolation to be based on a variety of different data point types.

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

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

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, which is useful in a mesh with a high aspect ratio. 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

$$\hat{x}=\frac{x-x_c}{\Delta_i} \quad (3)$$

and

$$\hat{y}=\frac{y-y_c}{\Delta_i}. \quad (4)$$

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

$$\frac{\partial f}{\partial \hat{x}}=\Delta_i\frac{\partial f}{\partial x} \quad (5)$$

and

$$\frac{\partial f}{\partial \hat{y}}=\Delta_i\frac{\partial f}{\partial y}. \quad (6)$$

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

$$\hat{L}_j=\frac{L_j}{\Delta_i}. \quad (7)$$

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

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

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

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

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

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

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

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

where $\mathbf{b}_{\hat{x}}\left(\hat{x},\hat{y}\right)=\frac{\partial}{\partial x}\mathbf{b}\left(\hat{x},\hat{y}\right)$ and $\mathbf{b}_{\hat{y}}\left(\hat{x},\hat{y}\right)=\frac{\partial}{\partial y}\mathbf{b}\left(\hat{x},\hat{y}\right)$.

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

$$\begin{aligned}C^*_j&=a_j f^*\left(\hat{x}_j,\hat{y}_j\right)+\hat{L}_j n_{x,j} f^*_{\hat{x}}\left(\hat{x}_j,\hat{y}_j\right)+\hat{L}_j n_{y,j} f^*_{\hat{y}}\left(\hat{x}_j,\hat{y}_j\right)\\&=\left(a_j\mathbf{b}\left(\hat{x}_j,\hat{y}_j\right)+\hat{L}_j n_{x,j}\mathbf{b}_{\hat{x}}\left(\hat{x}_j,\hat{y}_j\right)+\hat{L}_j n_{y,j}\mathbf{b}_{\hat{y}}\left(\hat{x}_j,\hat{y}_j\right)\right)\mathbf{p}.\end{aligned} \quad (12)$$

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

$$\mathbf{C}^*=\mathbf{B}\mathbf{p}, \quad (13)$$

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

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

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 $\mathbf{C}=\left(C_1,C_2,\cdots C_m\right)^T$.

Note that, if $m<\frac{1}{2}(n+1)(n+2)$, then an interpolant can be obtained which fits the data points exactly. However, in this case there are infinitely many basis functions which can interpolate the data and no way to distinguish between them. Many of the possible basis functions may behave wildly in between the data points, and so it is unlikely that an arbitrarily chosen one will be useful. If $m=\frac{1}{2}(n+1)(n+2)$, then a unique interpolant may exist which fits the data points exactly. However, this basis function may not interpolate smoothly in between the data points. (Moreover, for certain geometrical arrangements of the data points a singularity can occur, which may only be overcome by adding more data points to the stencil.) If $m>\frac{1}{2}(n+1)(n+2)$, then a unique approximant can

be obtained which minimises the sum of the squares of the errors in fitting the basis function to the data points. This allows a smoother interpolation in between the data points to be obtained, at the expense of including additional redundant data points in the stencil. However, it will be assumed in subsequent discussion that $m \geq \frac{1}{2}(n+1)(n+2)$, so that a unique interpolant or approximant can always be found.

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, therefore, 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 now 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 good weight function will 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, since it is common for the interpolation point to be coincident with one of the data points. Also, the weighting 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. There are infinitely many possible choices for the weight function, but a few are listed in table 1. Note that the dimensionless shape parameter, ε, is required for some of these functions; the smaller this value is, the more heavily the weight function discriminates between data points on the basis of distance. From numerical experiments based on the accuracy in reproducing test functions, it was found that the Gaussian weight function with $\varepsilon = 1.4$ was an appropriate choice.

Table 1: Some possible choices of the weight function for the moving least squares interpolation scheme

Weight Function	Formula
Inverse	$w_j = \frac{1}{1+\frac{\hat{r}_j}{\varepsilon}}$
Inverse Quadratic	$w_j = \frac{1}{1+\frac{\hat{r}_j^2}{\varepsilon^2}}$
Exponential	$w_j = e^{-\frac{\hat{r}_j}{\varepsilon}}$
Gaussian	$w_j = e^{-\frac{\hat{r}_j^2}{\varepsilon^2}}$

Let a weight matrix $\mathbf{w}$ be defined by

$$\mathbf{w} = \begin{bmatrix} w_1 & 0 & 0 & \cdots & 0 \\ 0 & w_2 & 0 & \cdots & 0 \\ 0 & 0 & w_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & w_m \end{bmatrix}. \tag{15}$$

It may then be shown that $\sum_{j=1}^{m}\left[w_j\left(C_j^* - C_j\right)\right]^2$ is minimised when

$$\mathbf{p} = \mathbf{DC}, \tag{16}$$

where $\mathbf{D} = \left((\mathbf{wB})^T(\mathbf{wB})\right)^{-1}(\mathbf{wB})^T\mathbf{w}$. This defines the polynomial of best fit (in a weighted least-squares sense) to the data points of the stencil.

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

$$\begin{aligned} f^*(\hat{x},\hat{y}) &= \mathbf{b}(\hat{x},\hat{y})\mathbf{p} \\ &= \mathbf{b}(\hat{x},\hat{y})\mathbf{DC} \end{aligned} \tag{17}$$

$$\begin{aligned} f_{\hat{x}}^*(\hat{x},\hat{y}) &= \mathbf{b}_{\hat{x}}(\hat{x},\hat{y})\mathbf{p} \\ &= \mathbf{b}_{\hat{x}}(\hat{x},\hat{y})\mathbf{DC} \end{aligned} \tag{18}$$

$$\begin{aligned} f_{\hat{y}}^*(\hat{x},\hat{y}) &= \mathbf{b}_{\hat{y}}(\hat{x},\hat{y})\mathbf{p} \\ &= \mathbf{b}_{\hat{y}}(\hat{x},\hat{y})\mathbf{DC}. \end{aligned} \tag{19}$$

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. This is essential if this method is to be used efficiently in a time stepping finite volume solver.

Singularities

It is well known in the literature that the moving least squares scheme may fail if the data points of the stencil are in a special spatial arrangement. Such stencils are termed singular stencils. It may be shown that a given moving least squares stencil will be singular if and only if the $\mathbf{B}$ matrix for the stencil possesses a non-trivial null space. In other words, if there are any $\mathbf{p}$ vectors (other than the zero vector) satisfying

$$\mathbf{Bp} = \mathbf{0}, \tag{20}$$

then the stencil is singular. If, on the other hand, the only solution to (20) is the zero vector, then the stencil is non-singular.

Using the singular value decomposition, the $\mathbf{B}$ matrix may be expressed as

$$\mathbf{B} = \mathbf{UDV}^T, \tag{21}$$

where $\mathbf{D} = \begin{bmatrix} D_1 & 0 & \cdots & 0 \\ 0 & D_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & D_{\frac{1}{2}(n+1)(n+2)} \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}$ is a diagonal matrix of the singular values and $\mathbf{V} = \begin{bmatrix} \mathbf{V}_1^T & \mathbf{V}_2^T & \cdots & \mathbf{V}_{\frac{1}{2}(n+1)(n+2)}^T \end{bmatrix}$. Note that $\mathbf{U}$ is a square matrix with m rows and m orthogonal columns (so that $\mathbf{UU}^T = \mathbf{U}^T\mathbf{U} = \mathbf{I}$) and the $\mathbf{V}_i$ 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 $\mathbf{B}$ can thus be found by finding the null space of a much simpler matrix Ψ, i.e. solving

$$\Psi\mathbf{p_0} = \mathbf{0} \tag{22}$$

for $\mathbf{p_0}$, where

$$\Psi = \begin{bmatrix} D_1\mathbf{V}_1 \\ D_2\mathbf{V}_2 \\ \vdots \\ D_{\frac{1}{2}(n+1)(n+2)}\mathbf{V}_{\frac{1}{2}(n+1)(n+2)} \end{bmatrix}. \quad (23)$$

It should be noted that, since the $\mathbf{V}$ 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 *SingularityTolerance*). 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 $\mathbf{B}$ 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 $\mathbf{B}$, therefore, the relation $\left(a_j\mathbf{b}\left(\hat{x}_j,\hat{y}_j\right)+\hat{L}_j n_{x,j}\mathbf{b}_{\hat{x}}\left(\hat{x}_j,\hat{y}_j\right)+\hat{L}_j n_{y,j}\mathbf{b}_{\hat{y}}\left(\hat{x}_j,\hat{y}_j\right)\right)\mathbf{p_0}=0$ 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 $\mathbf{B}$ 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'.

Moreover, it is known from linear algebra theory that the $\mathbf{V}$ vectors corresponding to the D values that are equal to zero form a complete basis to the null space of the original matrix [8]. In this case, therefore, they also provide a complete basis to the set of spanning polynomial coefficients. Since the ordering of the rows in (23) 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 (23) may be sorted.) Suppose that there are s singular values equal to zero, where $1 \le s \le \frac{1}{2}(n+1)(n+2)$. (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, ... D_s, and their corresponding $\mathbf{V}$ vectors will be $\mathbf{V}_1$, $\mathbf{V}_2$, ... $\mathbf{V}_s$. The family of spanning polynomials can then be expressed as

$$\mathbf{p_0} = \eta_1\mathbf{V}_1^T + \eta_2\mathbf{V}_2^T + \ldots + \eta_s\mathbf{V}_s^T \quad (24)$$

where η_1, η_2, ... η_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 $\mathbf{V}$ vectors are defined as row vectors. The parameter s can be thought of as the number of dimensions in the space of spanning polynomials.

This information is incredibly valuable, since it allows the singularity-causing geometric constraints obeyed by the data points of a singular stencil to be identified. Moreover, if the singularity is to be removed by adding new data points to the stencil, then the effectiveness of any particular candidate data point may be assessed by the degree to which it disobeys the singularity-causing constraints. Suppose that the new data point is located at $(\hat{x}_N,\hat{y}_N)$ and has data point properties of a, $\hat{L}$, n_x and n_y (as defined in (8)). A given spanning polynomial $\mathbf{p_0}$ of the original stencil will also span the new data point when

$$\left[a\mathbf{b}\left(\hat{x}_N,\hat{y}_N\right)+\hat{L}n_x\mathbf{b}_{\hat{x}}\left(\hat{x}_N,\hat{y}_N\right)+\hat{L}n_y\mathbf{b}_{\hat{y}}\left(\hat{x}_N,\hat{y}_N\right)\right]\mathbf{p_0}=0. \quad (25)$$

If this equation is combined with (24), then it may be written as

$$\kappa_1\eta_1+\kappa_2\eta_2+\ldots+\kappa_s\eta_s=0, \quad (26)$$

where $\kappa_i=\left[a\mathbf{b}\left(\hat{x}_N,\hat{y}_N\right)+\hat{L}n_x\mathbf{b}_{\hat{x}}\left(\hat{x}_N,\hat{y}_N\right)+\hat{L}n_y\mathbf{b}_{\hat{y}}\left(\hat{x}_N,\hat{y}_N\right)\right]\mathbf{V}_i^T$. Now if the new data point is spanned by all of the polynomials $\mathbf{V}_1^T$, $\mathbf{V}_2^T$ ... $\mathbf{V}_s^T$, then the entire family of spanning polynomials of the original stencil also span the new data point; the new data point therefore does not help to remove the singularity. If, however, there is (at least) one polynomial $\mathbf{V}_j^T$ which does not span the new data point, then $\kappa_j \neq 0$ and so η_j may be expressed in terms of the other parameter as

$$\eta_j = -\frac{1}{\kappa_j}\sum_{i=1..s,\, i\neq j}\kappa_i\eta_i, \quad (27)$$

providing that $s>1$. 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.

For the special case when $s=1$, there is only one (linearly independent) polynomial that spans the original stencil. Adding a new data point which is spanned by that polynomial will be useless, whereas adding a new data point which is not spanned by that polynomial will remove the singularity.

Based on this singularity theory, an algorithm for detecting and correcting singular stencils has been developed. This algorithm has been found to provide reliable detection and robust correction on a wide variety of unstructured meshes, both regular and irregular. Moreover, the number of additional data points required to do this was found to be reasonable (on average), meaning that the algorithm is also an efficient way of solving the singularity problem.

Use in a Convection-Diffusion Solver

Fundamental Equations

The two dimensional convection-diffusion equation can be written as

$$\frac{\partial f}{\partial t} = -u\frac{\partial f}{\partial x} - v\frac{\partial f}{\partial y} + \mu\left(\frac{\partial^2 f}{\partial x^2}+\frac{\partial^2 f}{\partial y^2}\right), \quad (28)$$

where u and v are the components of the uniform 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. In finite volume form, this may be written in terms of a line integral around the boundary of a control area, giving

$$A_{\text{C.A.}}\left.\frac{\partial f}{\partial t}\right|_{\substack{\text{C.A.}\\ \text{centroid}}} = -\oint_{\text{C.A.B.}} f u_n dl + \mu\oint_{\text{C.A.B.}}\frac{\partial f}{\partial n}dl, \quad (29)$$

where $\begin{pmatrix} n_x & n_y \end{pmatrix}$ is a unit vector which is locally normal to the boundary and points outwards from the control area, dl is

a length increment along the boundary, u_n is the component of the convection velocity in the direction of $\begin{pmatrix} n_x & n_y \end{pmatrix}$ and $\frac{\partial f}{\partial n}$ is the spatial derivative of f in that direction. $A_{\text{C.A.}}$ is the total area of the control area.

In the special case where the control volume in question is a mesh element bounded only by straight edges, with area A_{element} and its centroid at (x_c, y_c), the finite volume form of the convection-diffusion equation can be represented (approximately) as

$$A_{\text{element}} \left.\frac{\partial f}{\partial t}\right|_{(x_c,y_c)} = \sum_{\substack{p \in \text{element} \\ \text{edges}}} L_p \left[\mu \left.\frac{\partial f^*}{\partial n}\right|_{(x_p,y_p)} - f^*\left(x_p, y_p\right) u_{n,p} \right] \tag{30}$$

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 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).

Interpolation Scheme

In order to simulate the convection-diffusion equation on an unstructured mesh, it is necessary to use some kind of interpolation scheme to find approximate values for $\frac{\partial f^*}{\partial n}$ and f^* at each edge midpoint, based on the prescribed boundary conditions and known values of f at element centroids. To this end, a moving least squares scheme with a third order polynomial (in $\hat{x}$ and $\hat{y}$) basis was used. The weighting function chosen (see table 1) was Gaussian, with $\varepsilon = 1.4$. The stencils required for this scheme were selected using the algorithm for building non-singular stencils (with $SingularityTolerance = 0.8$).

Time Stepping Scheme

(30) provides an expression for the time derivative of f at the centroid of each element, and this expression thus allows the values of f at the element centroids to be time advanced using any of the explicit time stepping schemes. In this simulation, fourth order Runge-Kutta was chosen for this purpose.

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

$$r_{i,j} = \sqrt{\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2}. \tag{31}$$

The time step selected is then

$$\Delta t = \min\left\{ \frac{r_{i,j}}{\frac{\sqrt{u^2+v^2}}{Cfl} + \frac{\mu}{Fo \cdot r_{i,j}}} \right\}, \tag{32}$$

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

$$\Delta t = \min\left\{ \frac{Cfl \cdot r_{i,j}}{\sqrt{u^2+v^2}} \right\}, \tag{33}$$

which is a variant on the usual definition of the Courant-Friedrichs-Lewy number.

This time step selection scheme is not intended to determine the boundary of stability precisely, but merely to provide an approximate measure of the time scale on which the solution is expected to evolve. Most importantly, the time step scales automatically with the problem parameters, so that a combination of Fourier and Courant-Friedrichs-Lewy numbers may be determined which allows stable operation for a wide range of different problems. Stable operation was found to occur when using $Cfl = 0.5$ and $Fo = 0.5$.

Initial Conditions

Initial conditions must be specified in order to define the convection-diffusion problem completely. If $f(x,y,t)$ is the solution function, then it is necessary to specify the initial conditions as

$$f(x,y,0) = f_0(x,y), \tag{34}$$

where $f_0(x,y)$ is the scalar field defining the initial distribution of f. This initial condition may be specified arbitrarily, according to the requirements of the specific problem being simulated.

Boundary Conditions

The choice of boundary conditions for mixed convection-diffusion problems is slightly subtle, and is highly dependent on the physics of the specific problem being solved. In general, both convection and diffusion processes will act to transport heat across a given point on the boundary, and the process which dominates will determine which boundary conditions are more appropriate to use.

Consider the example domain shown in figure 1. Boundaries ***a*** and ***b*** are inflow boundaries. It is necessary to specify the value of f on these boundaries, since this represents the temperature of the fluid flowing into the domain from outside. Boundary ***c*** is parallel to the convection velocity, and so there is no convection across it; boundary conditions may therefore be chosen according to the thermal controls on this wall (i.e. specify f for a temperature controlled wall or specify $\frac{\partial f}{\partial n}$ for a heat flux controlled wall). Boundaries ***d*** and ***e*** are outflow boundaries and are thus allowed to float, with nothing specified there at all. However, on a slightly inclined boundary such as ***e***, it will sometimes be necessary to specify a non-floating boundary condition for purely numerical reasons. This is because the extrapolation required for a floating boundary condition introduces a small error into the solution in the region near the boundary, and this error may grow in magnitude as the solution propagates alongside it. It is the authors' experience that this issue may be resolved by specifying the spatial derivative of f (normal to either the boundary or the convection velocity) to be equal to zero, with minimal impact on the overall solution.

Results

Propagation of a Gaussian Pulse

The two dimensional convection-diffusion equation was solved on an irregularly shaped domain, filled with an irregular mesh of quadrilaterals. (The shape of the domain is indicated by the coloured region in figure 2.) Boundary conditions of $f = 0$ were applied to the 'west' and 'south' boundaries, while $\frac{\partial f}{\partial n} = 0$ was applied to the remaining boundaries. The initial conditions were given by

$$f = f_0 e^{-\frac{(x-x_0)^2+(y-y_0)^2}{2\sigma_0^2}}, \tag{35}$$

where $x_0 = y_0 = 0.5\text{m}$, $f_0 = 1$ and $\sigma_0 = 0.1\text{m}$ in this example. Convection velocities of $u = v = 1\text{m.s}^{-1}$ were used, while the parameter μ was assigned a value of $1.0 \times 10^{-3}\text{m}^2\text{.s}^{-1}$. This

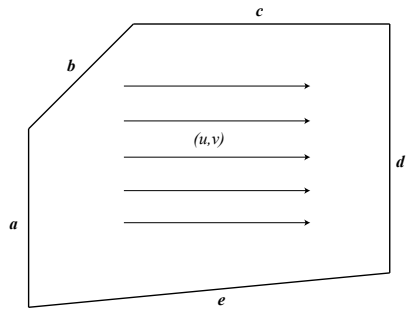


Figure 1: Example domain for the convection-diffusion problem. The arrows indicate the direction of the convection velocity.

problem has the analytical solution [9]

$$f = \frac{\sigma_0 f_0}{\sqrt{\sigma_0^2 + 2\mu t}} e^{-\frac{(x-x_0-ut)^2+(y-y_0-vt)^2}{2(\sigma_0^2+2\mu t)}}, \tag{36}$$

which is useful for verification purposes. The numerical and analytical solutions at times $t = 0$s, $t = 1$s and $t = 2$s are shown in figure 2. It may be seen that the numerical results correspond closely in form to the analytical solution, which indicates that the solver is accurate. In particular, there is very little dispersion error. Close inspection of the contours, however, shows that the peak value of the pulse in the numerical solution is slightly lower than the peak value predicted by the analytical solution. This indicates the presence of numerical diffusion, in addition to the diffusion specified for the problem.

Mixing

The two dimensional convection-diffusion equation was solved on a circular domain, where the fluid in the top region of the domain was initially set to a hot temperature of $f = 1$, while the fluid in the bottom region was initially set to a cold temperature of $f = -1$. In order for the boundary between the two regions to be sharply defined, the two regions of the domain were meshed separately using unstructured triangular meshes. Mixing was simulated through the combined effects of diffusion (using $\mu = 5 \times 10^{-4}\text{m}^2.\text{s}^{-1}$) and convection by a potential vortex. The imposed velocity field is given by

$$u = -\frac{\Gamma}{2\pi} \cdot \frac{y}{\sqrt{x^2+y^2+\delta^2}} \tag{37}$$

and

$$v = \frac{\Gamma}{2\pi} \cdot \frac{x}{\sqrt{x^2+y^2+\delta^2}}, \tag{38}$$

where $\Gamma = 1.0\text{m}.\text{s}^{-1}$ is the strength of the potential vortex and $\delta = 0.05$m is a term included in order to avoid a velocity singularity at the origin. The applied boundary condition of $\frac{\partial f}{\partial n} = 0$ ensures that heat cannot enter or leave the domain, and so the system is expected to converge towards a steady state where $f = 0$ (the initial average temperature) everywhere.

The time evolution of the numerical solution is shown in figure 3. While no analytical solution is available for verification, it can be seen that the numerical solution nevertheless behaves in the expected manner: the convection process causes fingers of interleaved hot and cold fluid to form in a vortex structure, while the diffusion process acts to reduce the temperature gradient between these fingers. As time advances, the temperature field becomes closer to being a uniform temperature of $f = 0$, which is the expected steady state condition.

Conclusions

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 the singularity theory was used to create an algorithm which may be used to build stencils which are guaranteed to be non-singular. The algorithm was used in a moving least squares solution of the convection-diffusion equation. Results were obtained for the propagation of a Gaussian pulse, and these were compared to the known analytical solution; they were found to be accurate, except for the excess numerical diffusion. The success of the scheme at solving this partial differential equation demonstrates its reliability. Most importantly, the algorithm introduced in part 1 for building non-singular stencils is clearly effective, as stable simulations would not be possible if any singular stencils remained.

Acknowledgements

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

References

[1] Y. Lipman, D. Cohen-Or, D. Levin, Data-Dependent MLS for Faithful Surface Approximation, Eurographics Symp. Geom. Proc. (2007).

[2] H. Desimone, S. Urquiza, H. Arrieta, E. Pardo, Solution of Stokes Equations by Moving Least Squares, Commun. Numer. Meth. En. 14 (1998) 907–920.

[3] A. Bodin, J. Ma, X. J. Xin, P. Krishnaswami, A meshless integral method based on regularized boundary integral equation, Comput. Method Appl. M. 195 (2006) 6258–6286.

[4] H. Netuzhylov, Meshfree collocation solution of boundary value problems via interpolating moving least squares, Commun. Numer. Meth. En. 22 (2006) 893-899.

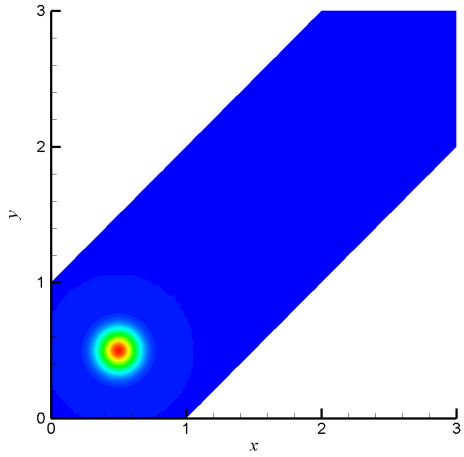
[5] C. Prax, H. Sadat, E. Dabboura, Evaluation of high order versions of the diffuse approximate meshless method, Appl. Math. Comput. 186 (2007) 1040–1053.

[6] W. Schoenauer and T. Adolph, How WE Solve PDEs, J. Comput. Appl. Math. 131 (2001) 473–492.

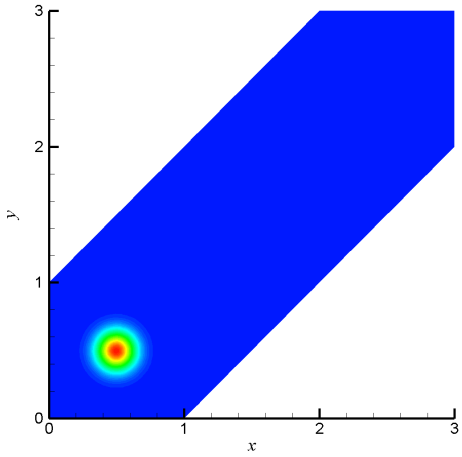
[7] H. Wang, H. K. Dahle, R. E. Ewing, M. S. Espedal, R. C. Sharpley, S. Man, An ELLAM Scheme for Advection-Diffusion Equations in Two Dimensions, SIAM J. Sci. Comput. 20-6 (1999) 2160–2194.

[8] G. H. Golub and C. F. Van Loan, Matrix Computations, The John Hopkins University Press (1996).

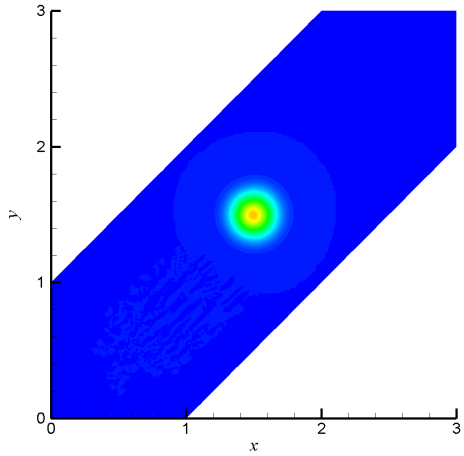
[9] O. Shipilova, H. Haario and A. Smolianski, Particle transport method for convection problems with reaction and diffusion, Int. J. Numer. Meth. Fl. 54 (2007) 1215–1238.



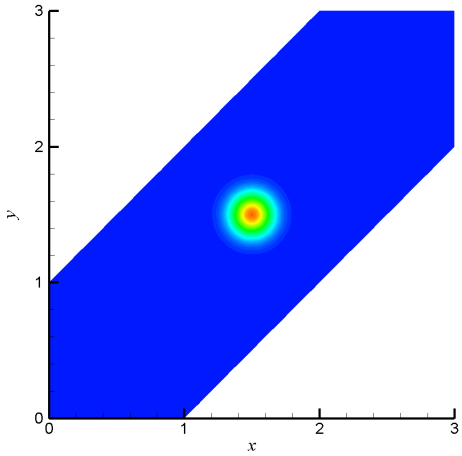
(a) $t = 0$s (numerical)



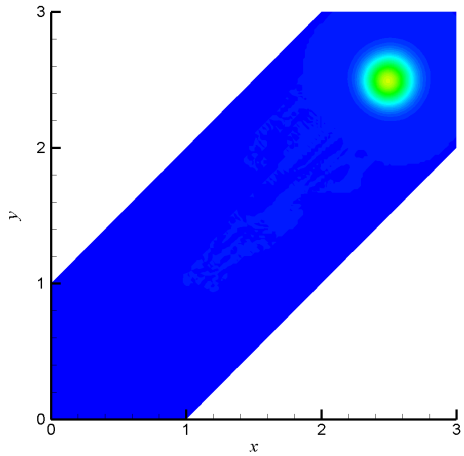
(b) $t = 0$s (theoretical)



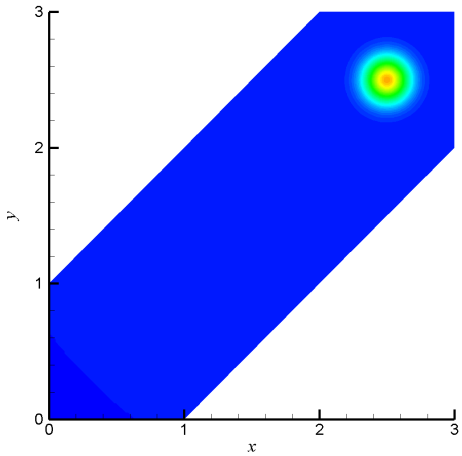
(c) $t = 1$s (numerical)



(d) $t = 1$s (theoretical)

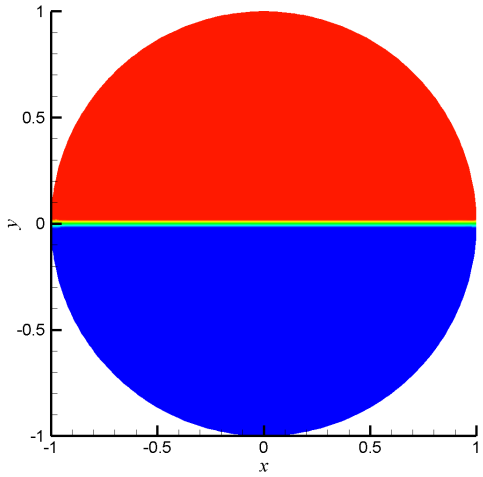


(e) $t = 2$s (numerical)

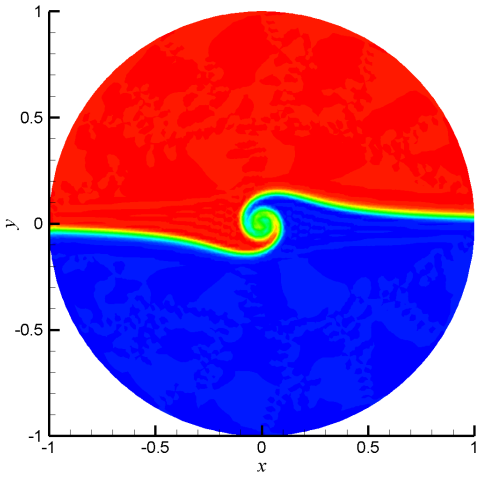


(f) $t = 2$s (theoretical)

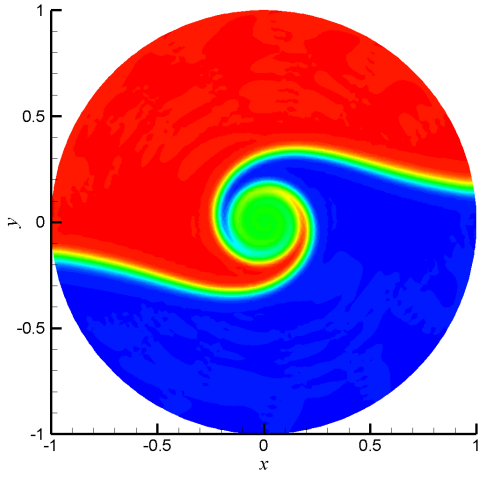
Figure 2: Convection-diffusion of a Gaussian pulse



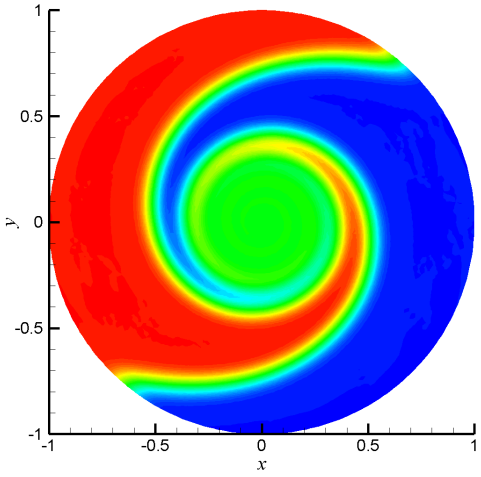
(a) $t = 0$s



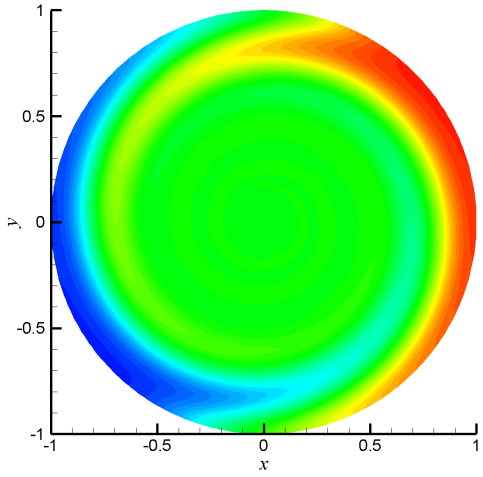
(b) $t = 0.2$s



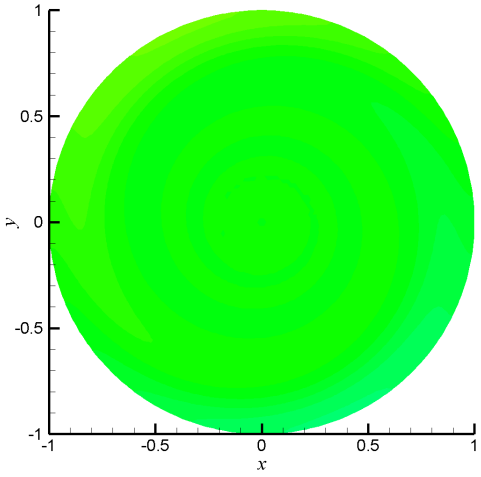
(c) $t = 0.99$s



(d) $t = 5.15$s



(e) $t = 25.14$s



(f) $t = 99.96$s