The Meshfree Finite Volume Method with application to multi-phase porous media models

Brody H. Foy\textsuperscript{a, *}, Patrick Perré\textsuperscript{b}, Ian Turner\textsuperscript{a,c}

\textsuperscript{a} School of Mathematical Sciences, Queensland University of Technology, QLD, 4000, Australia
\textsuperscript{b} LCMM, CentraleSupélec, Université Paris-Saclay, Grande Voie des Vignes, 92290 Châtenay-Malabry, France
\textsuperscript{c} Australian Research Council Centre of Excellence for Mathematical and Statistical Frontiers, Queensland University of Technology, Brisbane, Queensland, Australia

\textbf{A R T I C L E   I N F O}

Article history:
Received 21 June 2016
Received in revised form 14 October 2016
Accepted 21 December 2016
Available online 28 December 2016

Keywords:
Mesh-free
Finite-volume
Finite-element
Newton-Krylov
Multi-phase

\textbf{A B S T R A C T}

Numerical methods form a cornerstone of the analysis and investigation of mathematical models for physical processes. Many classical numerical schemes rely on the application of strict meshing structures to generate accurate solutions, which in some applications are an infeasible constraint. Within this paper we outline a new meshfree numerical scheme, which we call the Meshfree Finite Volume Method (MFVM). The MFVM uses interpolants to approximate fluxes in a disjoint finite volume scheme, allowing for the accurate solution of strong-form PDEs. We present a derivation of the MFVM, and give error bounds on the spatial and temporal approximations used within the scheme. We present a wide variety of applications of the method, showing key features, and advantages over traditional meshed techniques. We close with an application of the method to a non-linear multi-phase wood drying model, showing the potential for solving numerically challenging problems.

\textcopyright 2016 Elsevier Inc. All rights reserved.

1. Introduction

Over the last 40 years a great deal of work has been undertaken to develop techniques for numerically approximating the solution of partial differential equations (PDEs). Typically these techniques involve the discretisation of a problem domain into a series of nodal points connected by an underlying element structure, referred to as a mesh. Traditionally weak-form PDEs have been discretised using finite element methods (FEM), while strong-form PDEs have been discretised using finite different methods (FDM) or finite volume methods \cite{1}. All of these techniques can be implemented effectively in models for many different physical scenarios, however they have the shared limitation of requiring a well-structured mesh for basic implementation \cite{2}. This requirement can present difficulties in many practical applications, as often the use of a static or well-structured mesh is impractical, infeasible or computationally expensive.

To overcome these limitations, a significant amount of research has been undertaken towards the development of meshless methods; a class of numerical schemes that aim to avoid mesh generation, or reduce reliance on well-structured meshes. In application, few meshless methods are truly meshless, instead using a background meshing structure to perform calculations such as integrations \cite{2}. The formulation of meshless methods can vary greatly, however all techniques have the aim to more accurately and efficiently handle complex domain shapes and nodal arrangements.
Most major meshless method developments have occurred across three different streams. The first, led by Gingold [3] and Monaghan [4], have been integral representation techniques. These techniques approximate the unknown function value at each node as a definite integral around the node’s neighbourhood, relative to a weight function called the kernel. Gingold’s original motivation was to develop methods for efficiently modelling three-dimensional astronomical processes that lacked well-defined boundaries, such as exploding star systems and dust clouds. This led to a technique called smoothed particle hydrodynamics. Significant work was undertaken by Liu et al. in refining integral representation techniques. In particular, the development of reproducing kernel particle methods (RKPM) represented a significant step forward [5]. While integral representation techniques have been effectively applied to many problems in the literature, they often involve significant computational expense. Equally, the integral approximation acts as a smoothing operator, and thus can dampen and remove local information, unless care is taken in limiting neighbourhood sizes [2].

The second area of development has been through differentiation techniques. These involve the approximation of an unknown function value through its derivatives. This was commonly performed through the application of high-ordered Taylor-series expansions. In the early stages of development of these techniques, Jensen used this concept to create an unstructured mesh finite difference scheme [6]. The work was later expanded upon by Liszka and Orkisz [7], who improved stability through segmentation. While quite efficient, differentiation techniques have been shown to be unstable in applications. This is because the derivative operator amplifies local information, and can often cause errors to propagate throughout a simulation. Due to this amplification, differentiation techniques have rarely been applied on their own in modern literature, instead being used in hybrid settings [8].

The third major area of development has been through interpolation techniques. These techniques approximate function values through the construction of data interpolants around the neighbourhood of a node. Initial work in the field was led by Nayroles [9] and Belytschko [10], who used moving least squares (MLS) interpolators to create robust meshless methods. Soon after, the work of Belytschko was generalised into a family of techniques called partitions of unity, of which MLS interpolants are a subcategory [11,12]. More recent work by Liu and Gu used exact polynomial interpolation instead of regression fittings, creating a series of point interpolation methods (PIM) [13]. These were shown to be more efficient than regression interpolation methods, but also unstable under certain node placements. Stability issues with PIM were partially addressed through the use of a combination of radial and polynomial basis functions by Wang and Liu [14]. Radial basis function interpolants have also been successfully applied to the development of meshless methods for strong-form PDEs, by Liu et al. [15] and Liu et al. [16]. In application, interpolation methods showed higher convergence rates than traditional meshed methods and integration techniques. However they were seen to be more computationally expensive, particularly in higher dimensional applications.

While a variety of different meshless techniques have been developed, the majority of these have been built for applications to weak-form PDE systems. Very few meshless techniques exist within the literature which focus on application to conservative, strong-form PDEs. The field of numerical mathematics is often centred around providing a diverse range of tools to solve models for applications in an array of other fields. Thus we believe that the development of a new strong-form meshless method, and its application to a challenging physical problem, represents a novel contribution to the field.

Within this paper, we outline the basic framework of a new meshless method, which we call the Meshfree Finite Volume Method (MFVM). The MFVM is an interpolation based technique suitable for application to strong-form PDEs. In the next section we outline the preliminary concepts for the method such as the interpolant choices. In Sections 3 and 4 we present a full derivation of the method, including spatial and temporal discretisations and error bounds. In Section 5 we demonstrate basic error convergence under a variety of different numerical problems, and present a unique application of the method to a multi-phase micro-scale porous media model. We then close with a discussion of the method’s advantages and disadvantages in Section 6.

2. Preliminaries

2.1. Interpolant

To construct an interpolation based method, a suitable interpolant must first be chosen. The chosen interpolant should exhibit high versatility in representing function behaviours, while still being computationally inexpensive to construct. Equally, a common problem with many interpolant choices is the potential for singularities to form; where the numerical scheme breaks down if either not enough data is present in an area of the solution domain, or certain nodal arrangements are used. Considering this, an ideal interpolant should also be non-singular, regardless of nodal arrangements or the number of data points used in its construction. An interesting interpolant choice originally presented by Meinguet [17] is a curvature-constrained, least-squares interpolant. The interpolant $p(x, y)$ is chosen such that it simultaneously minimised squared error against a set of $N$ data points $(x_i, y_i, z_i), i = 1 \ldots N$, and the curvature of the function over the domain $\Omega$. Choosing the interpolant from a function space $P$ we define it as:

---

1 Strictly speaking this function is a least-squares approximation, not an interpolant. However for simplicity we will use the term interpolant to mean the approximation of a series of data points by a chosen function. This encompasses true interpolation as well as regression fittings.
\[
p(x, y) = \arg \min_{f \in P} \left( \sum_{i=1}^{n} |z_i - f(x_i, y_i)|^2 + \lambda \rho(\Omega, f) \right),
\]

where \(\lambda\) is a smoothing parameter which balances the least-squares and curvature constraint components, and

\[
\rho(\Omega, p) = \int_{\Omega} \left( \frac{\partial^2 p}{\partial x^2} \right)^2 + \left( \frac{\partial^2 p}{\partial y^2} \right)^2 + \left( \frac{\partial^2 p}{\partial x \partial y} \right)^2 \, dx \, dy.
\]

is the total curvature of the function over the domain.

In the formation of the interpolant the curvature-constraint serves a two-fold purpose. Firstly, it allows the interpolant to be completely non-singular, as in the presence of limited data it generates a planar surface. Secondly, the curvature-constraint can help to limit non-physical behaviour, which can often result in high-curvature (e.g. rippling).

### 2.2. Domain triangulation

To increase the versatility of the global interpolant defined in (1), the domain \(\Omega\) can be segmented into a set of disjoint elements \(\Omega_k, k = 1 \ldots K\) which when unioned approximate the overall domain. A separate interpolant is then constructed for each element, with the enforcement of continuity and differentiability along shared element edges. The global interpolant can be defined as the union of all local interpolants. Using a segment-based interpolant within a meshless method has the disadvantage of introducing a background element structure to the problem, which is used to define interpolating regions. However, unlike within meshed-schemes this element structure is not directly dependent on the spatial locations of data nodes, meaning complex domain shapes can be easily segmented, leading to more versatile implementations.

One of the most commonly used techniques for two-dimensional domain segmentation is triangulation (see [18] for further details). If the domain is segmented into a series of disjoint triangles, a separate interpolant must be defined for each element. A suitable choice of interpolant space is the Clough–Tocher (CT) space [19]. The CT space is a function space of cubic-polynomials defined over an equilateral triangle with vertices \(P_1 = (0, \frac{\sqrt{3}}{2}), P_2 = (-1, -\frac{\sqrt{3}}{2}), P_3 = (1, -\frac{\sqrt{3}}{2})\), referred to as the standard element \(\Omega_k\) (shown in Fig. 1). The polynomial is defined by a weighted sum of the function values and first derivatives at each vertex, as well as the outward normal derivatives at each mid-point edge. To aid computational efficiency a simple averaging approximation can be made to obtain the derivatives at mid-points, which reduces the degrees of freedom from 12 to 9, leading to a Hsieh–Clough–Tocher (HCT) function space. Using the HCT functions, for a point \((\xi, \eta)\) within the element, the interpolant \(\hat{p}\) can be calculated as

\[
\hat{p}(\xi, \eta) = \sum_{i=1}^{9} \hat{h}_i(\xi, \eta) d_i,
\]

where \([d_i]_{i=1}^{9}\) denotes the function and derivative values at the vertices, and \([\hat{h}_i]_{i=1}^{9}\) denotes the 9 HCT basis functions (for exact forms, the reader is directed to [20]). To transform this interpolant to a domain element \(\Omega_k\) with coordinate system \((x, y)\) a simple affine transformation [21] is used, of the form

\[
(\xi, \eta)^T = B_k(x, y)^T + (b_1, b_2)^T = F_k(x, y),
\]

with the exact form of \(F_k\) given in Appendix A.
The set of basis functions \( \{ h_k^r \}_r \) for element \( \Omega_k \) can be defined as

\[
h_k^r(x, y) = \hat{h}_i \circ F_k(x, y).
\]

To compute gradients of the basis functions we first introduce the notation

\[
\nabla = \left( \frac{\partial}{\partial x}, \frac{\partial}{\partial y} \right)^T, \quad \hat{\nabla} = \left( \frac{\partial}{\partial \xi}, \frac{\partial}{\partial \eta} \right)^T.
\]

Through the chain rule, it can be shown that for a differentiable function \( \phi(x, y) \)

\[
B_k^T \left( \nabla \phi \circ F_k^{-1}(\xi, \eta) \right) = \left( \hat{\nabla} \hat{\phi} \right) \circ F_k(x, y),
\]

where \( \hat{\phi}(\xi, \eta) = \phi \circ F_k^{-1}(\xi, \eta) \), and \( F_k^{-1} \) is the inverse affine transformation of \( F_k \). If we set \( \phi = h_k^r \), (3) can be rearranged to give

\[
\nabla h_k^r(x, y) = B_k^{-T} \left( (\hat{\nabla} \hat{h}_i) \circ F_k(x, y) \right).
\]

Similar expressions can be derived for higher-ordered derivatives. Finally, we note that the Jacobian matrix of the transformation \( F_k \) is \( B_k \) meaning that

\[
\int_{\Omega_k} h_k^r \, dxdy = |\det B_k| \int_{\hat{\Omega}_k} \hat{h}_i \, d\xi d\eta.
\]

(5)

Using (4) and (5), we can perform all function evaluations and integrations on the standard element, and then transform the results back to each local element in the problem domain, greatly reducing computational expense.

2.3. Formation as a matrix system

Using the HCT function space, we formulate the interpolant in (1) as a matrix system defining the unknown function and derivative values at each node in the triangulation of the problem domain. If the domain is segmented into \( K \) distinct elements, using \( Q \) element vertices, we have \( 3Q \) unknowns. We group these unknowns together as a vector \( \mathbf{c} \), and denote the unknowns for element \( \Omega_k \) as \( \mathbf{c}_k \). Considering (2) the interpolant for element \( \Omega_k \) is

\[
p(x, y) = \mathbf{c}_k^T \mathbf{h}^k(x, y),
\]

where \( \mathbf{h}^k(x, y) \) denotes \( \hat{h} \circ F_k(x, y) \). Considering the least-squares component of (1) we have

\[
\mathbf{c} = \arg \min_{\mathbf{d} \in \mathbb{R}^Q} \sum_{i=1}^{N} (z_i - d_i^T \mathbf{h}^k(x_i, y_i))^2 = \arg \min_{\mathbf{d} \in \mathbb{R}^Q} \| \mathbf{z} - A\mathbf{d} \|^2_2,
\]

where \( A \in \mathbb{R}^{N \times 3Q} \) is a matrix of the contributions from each \( \mathbf{h}^k \). By minimising this expression, we obtain the set of normal equations

\[
A^T A \mathbf{c} = A^T \mathbf{z}.
\]

When solved, this system defines the coefficients of the surface interpolant \( p(x, y) \) that minimises the total squared error from the data vector \( \mathbf{z} \).

Alongside this, we consider the minimisation of the curvature-constraint component of (1). We first note that the total domain integral is comprised of a sum of integrals over each element \( \Omega_k \). Since the HCT functions are semi-cubic, we can exactly compute each integral using 4-point Gaussian quadrature [22]. Denoting the four quadrature points for the standard element as \( (\xi_i, \eta_i) \), \( i = 1 \ldots 4 \), with corresponding weights \( w_i \), we have

\[
\rho(\Omega, p) \approx \sum_{k=1}^{K} \sum_{i=1}^{4} w_i \left[ \left( \frac{\partial^2 \mathbf{h}^k}{\partial x^2} | x^k_i \right)^2 + \left( \frac{\partial^2 \mathbf{h}^k}{\partial y^2} | y^k_i \right)^2 + \left( \frac{\partial^2 \mathbf{h}^k}{\partial x \partial y} | x^k_i \right)^2 \right],
\]

where \( x^k_i = F_k^{-1}(\xi_i, \eta_i) \). Minimising \( \rho(\Omega, p) \) with respect to each variable \( c_j \) gives a system of \( 3Q \) linear equations, formulated as the matrix system

\[
M \mathbf{c} = 0,
\]

where \( M \in \mathbb{R}^{3Q \times 3Q} \). Combining this with (6) gives a matrix system that defines the curvature-constrained, least-squares interpolant coefficients as
(A^T A + \lambda M) c = A^T z. \tag{7}

We note first that \( A^T A \) is symmetric–positive–definite by definition. Furthermore, by construction \( M \) is also symmetric–positive–definite, as it is generated using a sum of squares. Thus, since \( \lambda \) is a positive constant, the matrix \( A^T A + \lambda M \) is SPD. This is a very desirable property, as it allows for efficient solution methods through either Cholesky factorisation, or a preconditioned conjugate gradient method [1].

2.4. Estimation of the smoothing parameter \( \lambda \)

An effective technique for the estimation of parameter \( \lambda \) within (7) is the generalised cross validation algorithm, originally presented by Craven and Wahba [23]. This involves the minimisation of the cross validation function

\[
\hat{V}(\lambda) = \frac{N\|\bar{z}(I_N - Q_\lambda)\|^2}{\text{trace}(I_N - Q_\lambda)},
\]

where \( Q_\lambda = A(A^T A + \lambda M)^{-1} A^T \) is the influence matrix. Hutchinson and De Hoog [24] have developed an efficient numerical algorithm for approximating the minimisation of this function through the approximation

\[
\text{trace}(I_n - Q_\lambda) \approx k^T (I_n - Q_\lambda) k,
\]

where the elements of \( k \) take values \(-1\) and \(1\) with equal probability. Using this approximation the cross validation function becomes

\[
\hat{V}(\lambda) = N \frac{\|y\|^2 - 2\alpha_1 A^T y + \alpha_1 A^T A \alpha_1}{(N - \alpha_2^T A^T k)^2},
\]

where \( \alpha_1 = (A^T A + \lambda M)^{-1} A^T y \), and \( \alpha_2 = (A^T A + \lambda M)^{-1} A^T k \).

We note that in application we have found that \( \lambda \) need only be estimated to within an order of magnitude, meaning the minimisation can be performed inexpensively.

3. Meshfree Finite Volume Method framework

To formulate our meshless method, we start with a strong-form PDE for a variable \( \phi \), written in conservation form as

\[
\frac{\partial \phi}{\partial t} + \nabla \cdot \mathbf{J}(\phi) + S(\phi) = 0,
\]

where \( \mathbf{J} = (J_x, J_y)^T \) and \( S \) are the flux and source terms respectively. We formulate our method using techniques derived from a classical finite volume setting. We first discretise the domain into a series of \( N \) nodes, with unknown function values \( \phi_i \) (\( i = 1 \ldots N \)). We then construct a control volume \( V_i \) with boundary \( \partial V_i \) around each node. In a classical finite volume setting neighbouring control volumes would have shared edges so as to uniquely and fully cover the domain. However, in a meshless environment this is not required. We instead construct disjoint radial control volumes of radial length \( \delta_i \), with arc control volumes constructed along boundary domains. An example is given in Fig. 2.
Integrating (8) around each control volume, and applying Gauss’ Divergence Theorem gives

\[
\int_{V_i} \frac{\partial \phi}{\partial t} dV_i + \oint_{\partial V_i} J(\phi) \cdot \hat{n} dS_i + \int_{V_i} S(\phi) dV_i = 0,
\]

(9)

where \(dS_i\) is an infinitesimal line segment on the boundary \(\partial V_i\).

We compute the two volume integrals in (9) using a mid-point quadrature rule, and the boundary integral using \(q\) evenly spaced quadrature points along the arc edge, to give

\[
\frac{d\phi_i}{dt} + \frac{2}{\bar{q} \delta} \sum_{l=1}^{q} (J_x(\phi_{l}) \cos(\theta_l) + J_y(\phi_l) \sin(\theta_l)) + S(\phi_i) = 0,
\]

(10)

where \(J = (J_x, J_y)^T\), and \(\theta_l = \frac{2\pi l}{q}\). This approximation has error of order \(O(\max(\delta^2, q^{-2} \delta^3))\) [25]. Boundary conditions are treated in the standard way; through replacement of flux values within the boundary integrals with the specified Robin or Neumann flux condition, or through direct calculation of Dirichlet conditions.

Considering (10) we arrive at a differential equation for each \(\phi_i\), of the form

\[
\frac{d\phi_i}{dt} = B_i(\phi_i),
\]

where the form of \(B_i(\phi)\) follows logically from (10). We note however that each \(B_i(\phi)\) may still contain spatial derivatives, depending on the form of \(J\). To solve this equation system numerically, approximations of the spatial derivatives need to be constructed. In a standard finite volume method spatial derivatives are approximated using finite differences constructed between neighbouring nodes, and in control volume finite element methods the gradients are approximated through the use of shape functions [1]. Within a meshless environment we can instead use data interpolants to approximate these terms. We simultaneously interpolate the domain, and use this interpolant to approximate each \(B_i(\phi)\). This leads to a differential algebraic equation (DAE) system

\[
\frac{d\phi}{dt} = \hat{B}(\phi, c),
\]

(11)

\[
(A^T A + \lambda M)c - A^T \phi = 0,
\]

(12)

where \(\hat{B}\) is a vector function comprising of the approximations of each \(B_i(\phi)\) using \(c\).

The error in the approximation of a function \(f\) (and its derivatives) over an element \(\Omega_k\) using a HCT interpolant \(p(x)\) is bounded by

\[
\|f - p\|_{m,2} \leq C_l^3 \|f\|_{m,2}, \quad m = 0, 1, 2,
\]

(13)

for a positive constant \(C\), where \(l_k\) is the maximum side length of \(\Omega_k\) and \(\|\cdot\|_{m,2}\) is the standard Sobolev norm [26]. This means that the spatial error of the formulation given in (11) and (12) has an error order dependent on the order of spatial derivatives within \(J\). As an example, for \(J\) containing at most first-order spatial derivatives the method has error of order \(O(l^2)\), where \(l = \max(l_k)\).

4. The exponential Euler method

To provide an efficient temporal iteration scheme, we use the exponential Euler method. We first take (11), multiply by \(A^T\) and then apply (12) to give

\[
(A^T A + \lambda M) \frac{dc}{dt} = A^T \hat{B}(\phi, c).
\]

Within the function \(\hat{B}\) we estimate \(\phi\) using the interpolant surface \((\phi \approx Ac)\), leading to a differential equation system for \(c\)

\[
\frac{dc}{dt} = (A^T A + \lambda M)^{-1} A^T \hat{B}(Ac, c) = g(c).
\]

(14)

We linearise this system about \(t = t_n\) such that

\[
\frac{dc}{dt} = g(c_n) + J_n(c - c_n), \quad c(t_n) = c_n,
\]

where

\[
J = (A^T A + \lambda M)^{-1} A^T \frac{d}{dc} \left( \hat{B}(Ac, c) \right).
\]
Calculating the exact solution of the linearised system at \( t = t_{n+1} \) gives the scheme

\[
c_{n+1} = c_{n} + \tau_{n} \psi(\tau_{n}, J_{n}) g(c_{n}),
\]

where

\[
\psi(A) = A^{-1}(e^A - I), \quad \tau_{n} = t_{n+1} - t_{n}.
\]

It is well known in the literature that the EEM discretisation is of second-order, and similar to backward differentiation formulae (BDFs) is A-stable [27]. In application, for the EEM to perform competitively against iterative BDF schemes, the function \( \psi(\tau_{n}, J_{n}) \) is approximated using a Jacobian free strategy. The approximation used [27] involves projection onto the Krylov subspace \( K_{m}(J_{n}, g_{n}) \) such that

\[
\psi(\tau_{n}, J_{n}) g_{n} \approx \beta V_{m} \psi(\tau_{n} H_{m}) e_{1},
\]

where \( \beta = \|g_{n}\|_{2} \), and \( V_{m} \) is a matrix whose columns form an orthonormal basis for \( K_{m}(J_{n}, g_{n}) \). Arnoldi’s method [28] is used to generate the decomposition

\[
J_{n} V_{m} = V_{m} H_{m} + \beta_{m} v_{m+1} e_{m}^{T}.
\]

This allows the computation of \( \psi(\tau_{n}, J_{n}) \) to be efficiently approximated by \( \psi(\tau_{n} H_{m}) \), where the Hessenberg matrix \( H_{m} \) is of much smaller dimension. To avoid the explicit formation of \( J \), we approximate its action on a vector \( w \) using difference quotients

\[
J(c)w \approx \frac{g(c + \varepsilon w) - g(c)}{\varepsilon},
\]

for a suitably small \( \varepsilon \) [29]. This approximation is standard within the literature, and based on numerical results does not appear to introduce significant error. Carr et al. [30] showed that the approximation given in (16) retains second-order accuracy.

By using a Jacobian–free Krylov subspace method, each step in (15) can be efficiently approximated, requiring only the calculation of \( g(c_{n}) \). However, we note that the form of \( g \) requires the computation of \( (A^{T} A + \lambda M)^{-1} b \) for a given vector \( b \). Since \( (A^{T} A + \lambda M) \) is SPD, we efficiently approximate the action of its inverse using a sparse conjugate gradient method [1] with incomplete Cholesky preconditioning.

4.1. Adaptive timestepping

To further improve the efficiency of the EEM, we implement an adaptive time-stepping algorithm, through the comparison of the solution \( c_{n+1} \) to a two-step approximate solution \( c_{n+1}^{(2)} \), which is computed as

\[
c_{n+1/2} = c_{n} + \frac{\tau_{n}}{2} V_{m} \psi\left(\frac{\tau_{n}}{2} H_{m}\right) e_{1},
\]

\[
c_{n+1}^{(2)} = c_{n+1/2} + \frac{\tau_{n}}{2} V_{m} \psi\left(\frac{\tau_{n}}{2} H_{m}\right) V_{m}^{T} g(c_{n+1/2}),
\]

requiring only one extra evaluation of \( g \), and retaining second-order accuracy [30]. Carr et al. [30] showed that an effective multiplicative factor for the timestep can be defined as

\[
\alpha = 0.9 \left(\frac{1}{\|c_{n+1} - c_{n+1}^{(2)}\|_{\text{WRMS}}}ight)^{1/3},
\]

where

\[
\|c\|_{\text{WRMS}} = \left(\frac{1}{N} \sum_{i=1}^{N} (c_{i}/w_{i})^{2}\right)^{1/2},
\]

with weights \( w_{i} = RT OL_{i}|c|_{i} + AT OL_{i} \), for given absolute and relative error tolerances \( AT OL_{i} \) and \( RT OL \). The success of the current timestep is determined by the condition

\[
0.25 \|c_{n+1} - c_{n+1}^{(2)}\|_{\text{WRMS}} \leq 1.
\]
4.2. Choice of Krylov subspace dimension

The criterion for adaptively selecting the Krylov subspace dimension \( m \) in all approximations of matrix exponentials within the EEM is based on the error estimate given in [27]

\[
\psi(\tau_n J_n) g_m - \beta_m \psi(\tau_n H_m) e_1 \approx \tau_n \beta_m \left[ e_m^T \psi(\tau_n H_m) e_1 \right] v_{m+1} = \rho_m.
\]

To ensure that the error of this approximation does not exceed the error of the temporal linearisation, we use the selection criterion suggested by Hochbruck et al. [27]

\[
\tau_n \| \rho_m \| \text{WRMS} < 1.
\]

Due to \( m \) being quite small, \( \psi(\tau_n H_m) \) can be inexpensively computed. In the literature this is commonly achieved using Padé type approximations [27,30].

5. Results

To illustrate the MFVM we provide applications of the method to a variety of different problems.

5.1. A 2D advection–diffusion problem

The first application we present is a linear advection–diffusion PDE, of the form

\[
\frac{\partial \phi}{\partial t} = \nabla \cdot (\phi \mathbf{v} - D \nabla \phi) = 0,
\]

(17)

where

\[
D = \text{diag}(D_{xx}, D_{yy}), \mathbf{v} = (u, v)^T.
\]

On an infinite domain, with decay boundary conditions, and an initial condition

\[
\phi(x, y, 0) = \exp\left(-\frac{(x-x_c)^2}{D_{xx}} - \frac{(y-y_c)^2}{D_{yy}}\right).
\]

this PDE has the closed-form analytical solution

\[
\phi(x, y, t) = \frac{1}{4t + 1} \exp\left(-\frac{(x-ut-x_c)^2}{D_{xx}(4t + 1)} - \frac{(y-vt-y_c)^2}{D_{yy}(4t + 1)}\right),
\]

where \((x_c, y_c)^T\) is the centre of the pulse initial condition. We restrict this problem to the finite domain \([0, X] \times [0, Y]\) and apply Dirichlet boundary conditions given by evaluating the analytical solution on the domain edges. For parameter values, we set \( D_{xx} = D_{yy} = 0.01 \) and \( u = v = 1 \), creating a model with a high Péclet number, that numerically behaves quite stiffly.

In this study the advective components of the flux were approximated at the quadrature points on the control volume faces using the surface interpolant. Although this averaging approach did not appear to cause instability or inaccuracy of the solution for the range of parameters considered, despite the high Péclet number, we believe that it may prove beneficial to consider flux limiting schemes in future work. The investigation of the most appropriate flux limiter to use in this situation is deferred as a clear direction for future work.

To analyse the performance of the scheme, we solve (17) repeatedly under an increasing number of interpolation regions, characterised by the maximum element side length of any region. Fig. 3 gives a log–log plot of the error reduction under decreasing element size. As the figure shows, the log error of the system decreases with a slope of approximately 1.9, suggesting quadratic error convergence. This follows logically from (13), considering that the flux term of (17) contains at most first-order spatial derivatives. Error convergence conforming to (13) has been seen in many other PDEs that the method has been tested against. However, these results have been omitted for brevity.

5.2. Comparison to an unstructured finite volume method

To contrast the MFVM to a standard unstructured finite volume method, we present the same advection–diffusion problem as outlined in section 5.1. For the meshed FVM technique, we place down a series of nodes across the domain, and discretise the domain into a series of triangular elements. We then centre control volumes at each element vertex, with edges of the volumes being created by joining the centroids of each element to the mid points of each element edge. Once control volumes were created, the fluxes through each face were approximated using finite differences, with averaging for the diffusive terms and upwinding for the advection terms.
Fig. 3. Log–log plot of error against the maximum spline region side length, solved with $X = Y = 2$. The slope of the line of best fit is approximately 1.9, suggesting quadratic error convergence.

Fig. 4. The four unstructured meshes that the FVM problem and MFVM were tested on, with triangular spline elements outlined in blue, and FVM control volumes outlined in red. The four meshes consist of 28, 126, 472 and 1863 control volume nodes respectively. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

To compare this to the MFVM, we use the element structure from the FVM formulation as the interpolating region structure for the MFVM, with control volumes dispersed evenly over the entire domain (to an average of 4 per spline region). We compare the maximum absolute error of the two methods from the exact solution (given in section 5.1) under four different unstructured meshes, shown in Fig. 4. All solutions were computed at an end time of $t = 1$, with $D_{xx} = D_{yy} = 0.1$ and $u = v = 1$, using a timestep chosen to be small enough to ensure error is due primarily to the spatial approximations.
achieved quadratic, under section 3, regions. We assume that solutions of the problem on domains do not lie fully within the circle, and the high density of control volumes on the circular boundary.

Table 1
Comparison of error between an unstructured FVM and the MFVM for a variety of unstructured meshes.

<table>
<thead>
<tr>
<th>Element nodes</th>
<th>Method</th>
<th>Coefficient matrix size</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>28</td>
<td>FVM</td>
<td>28 x 28</td>
<td>0.2352</td>
</tr>
<tr>
<td>126</td>
<td>MFVM</td>
<td>64 x 64</td>
<td>0.0087</td>
</tr>
<tr>
<td>472</td>
<td>FVM</td>
<td>126 x 126</td>
<td>0.0155</td>
</tr>
<tr>
<td>1863</td>
<td>MFVM</td>
<td>378 x 378</td>
<td>0.0015</td>
</tr>
<tr>
<td>28</td>
<td>FVM</td>
<td>472 x 472</td>
<td>0.0043</td>
</tr>
<tr>
<td>126</td>
<td>MFVM</td>
<td>1416 x 1416</td>
<td>0.00045</td>
</tr>
<tr>
<td>472</td>
<td>FVM</td>
<td>1863 x 1863</td>
<td>0.0012</td>
</tr>
<tr>
<td>1863</td>
<td>MFVM</td>
<td>5589 x 5589</td>
<td>0.0001</td>
</tr>
</tbody>
</table>

Fig. 5. An example discretisation of the circular domain, with splining nodes and control volumes shown (left), and corresponding problem solution (right). Note that splining regions do not lie fully within the circle, and the high density of control volumes on the circular boundary.

The error convergence can be seen in Table 1, with the MFVM vastly outperforming the FVM, producing significantly lower spatial errors, even when the matrix system size was smaller, and with error converging at a faster rate.

5.3. Circular domain problem

One of the key advantages of the MFVM is that while convergence is related to the number of splining regions, boundary conditions are enforced on the control volumes, not on the interpolants themselves. This means that the splining regions do not need to perfectly adhere to the problem domain. To illustrate this, we present the solution of a diffusion problem on a circular domain. To provide a simple test case, we take the diffusion PDE

\[ \frac{\partial \phi}{\partial t} = \nabla \cdot (D \nabla \phi) + S(x, y, t), \]

where \( D = \text{diag}(D_{xx}, D_{yy}) \) and \( S(x, y, t) \) is a source term.

We define the problem on the circular domain \( \{(x, y) \in \mathbb{R}^2 | x^2 + y^2 \leq 1\} \), with a zero Dirichlet condition on the boundary. We assume a solution \( \phi_{\text{exact}} \) of the form

\[ \phi_{\text{exact}}(x, y, t) = \frac{1}{10t + 1} (1 - x^2 - y^2), \]

with \( S(x, y, t) \) chosen such that \( \phi_{\text{exact}} \) satisfies the original PDE.

To solve this problem numerically, we discretise the square domain \([-1.1, 1.1] \times [-1.1, 1.1]\) into a series of splining regions. We then place radial control volumes over the entire domain, with particular care to place many on the circular boundary defined by \( x^2 + y^2 = 1 \), as shown in Fig. 5. The PDE is then solved using the discretisation techniques outlined in section 3, with nodes outside the circular domain set to value 0. Using \( D = 1 \), we solved the problem to an end time of \( t = 1 \), under an increasing number of splining regions and control volumes. Convergence of the maximum error is approximately quadratic, and is shown in Fig. 6. This illustrates an advantage of the method, that quadratic convergence can still be achieved without the splining regions precisely matching to the problem domain.
Fig. 6. Log–log plot of error against the maximum spline region side length for the circular domain problem. The slope of the line of best fit is approximately 2.25, suggesting quadratic error convergence.

Fig. 7. A schematic of the problem domain (left) with splining regions and control volumes marked. The corresponding problem solution at $t = 1$ with $C = 2$, is also given (right).

5.4. Non-linear advection–diffusion problem

For another non-square domain problem we take a non-linear transport PDE

$$\frac{\partial \phi}{\partial t} = \nabla \cdot (D \nabla \phi - c \phi),$$

where $D = \text{diag}(\phi^2, 5\phi^2)$ and $c = C \phi$. We solve this problem on the L-shaped domain $([-1, 1] \times [1, 1]) \backslash ([0, 1] \times [0, 1])$, with a zero Dirichlet boundary, and the initial condition

$$\phi(x, y, 0) = \exp(-20(x - 0.25)^2 - 20(y - 0.25^2)).$$

Splining regions and control volumes were placed evenly and regularly across the domain, as indicated in Fig. 7. To analyse error convergence we first solve the problem with a regular finite difference technique to an end time of $t = 1$, under a highly refined spatial and temporal mesh. We take this as an approximation to the exact solution, and calculate the MFVM solution error as the absolute difference between the MFVM and finite difference solution over the domain. Error convergence for $C = 1$ and $C = 2$ is approximately quadratic, and can be seen in Fig. 8.

5.5. Modelling multi-phase porous media

As a final illustration of the advantages of the method we present an application in which a meshless technique is desirable. An interesting problem choice can be found in multi-phase porous media modelling, applied to heterogeneous wood drying. To model the transfer of water within wood, a simple diffusion model can be used, of the form

$$\frac{\partial \phi}{\partial t} = \nabla \cdot (D \nabla \phi),$$
Fig. 8. Error convergence for the PDE from Section 5.4, with \( C = 1 \) (left), and \( C = 2 \) (right). The slopes of the lines of best fit are approximately 2.31 and 1.79, suggesting quadratic error convergence.

Fig. 9. The processing stages of the pore morphology image. The initial image (left), identification of contour boundaries (middle) and conversion to a two-phase grid (right).

where \( \mathbf{D} = \text{diag}(D_{xx}, D_{yy}) \). To effectively apply this model to a physical scenario, accurate estimates of the diffusion rates \( D_{xx} \) and \( D_{yy} \) are needed. However, due to the porous structure of wood at the micro-scale, these rates can be quite difficult to infer on the macro-scale.

Recently, Perré et al. showed that diffusion rates could be estimated by simulating multi-phase flow on the micro-scale pore structure [31]. By constructing a virtual representation of the pore structure from a pore morphology image, a realistic two-phase (gas and solid) domain can be created (as seen in Fig. 9). Models on the micro-scale structure can then incorporate the separate behaviours of water and vapour flow through each phase, and thus allow for the estimation of the averaged macro-scale behaviour.

To model the transfer of bound water \((\rho_0 X)\) and water vapour \((\varepsilon_g \rho_v)\) within the pore morphology we use a model presented by Carr et al. [32], based on previous work by Perré and Turner [33]. For full detail of the model, the reader is referred to [32], but in brief

\[
\frac{\partial \phi}{\partial t} + \nabla \cdot \mathbf{Q}(l) = 0, \tag{18}
\]

where

\[
\phi = \rho_0 X + \varepsilon_g \rho_v,
\]

with \( \rho_0 \) being the average bound water phase density \((\text{kg} \cdot \text{m}^{-3})\), \( X(x, y, t) \) the moisture content, \( \varepsilon_g \) the gas-phase volume-fraction, and \( \rho_v(x, y, t) \) the water-vapour density. The flux term \( \mathbf{Q}(l) \) is driven by diffusion, which is defined differently in the bound water phase \((l = b)\) and the gas phase \((l = g)\):

\[
\mathbf{Q}^{(b)} = -\rho_0 D_b \nabla X,
\]

\[
\mathbf{Q}^{(g)} = -\frac{D_v}{1 - \rho_v/\rho_g} \nabla \rho_v.
\]
where $D_b$ and $D_v$ denote the diffusion rates (m$^2$s$^{-1}$) for bound water and water vapour respectively. Under a constant atmospheric pressure, and isothermal temperature of 20°C, we have the bound liquid and vapour diffusivities

$$D_b = \exp(-27.4867 + 10.8951X),$$
$$D_v = 2.568 \times 10^{-5}.$$

The water vapour density ($\rho_v$) is driven by the ideal gas law, such that

$$\rho_v = \frac{P_v(X)M_v}{297.15R},$$

with vapour pressure ($P_v$) driven by the bound water such that

$$P_v(X) = \exp(7.507)\left(1 - \exp\left(-0.764269 \left(\frac{X}{XFSP}\right) - 3.67872 \left(\frac{X}{XFSP}\right)^2\right)\right),$$

with fiber saturation point $XFSP = 0.305$ and molar mass $M_v = 0.01806$ (kg mol$^{-1}$). To close the system, we apply periodic boundary conditions, with a gradient $\Delta X_x$ and $\Delta X_y$ in moisture content imposed over the domain, such that

$$X(x_2, y, t) = X(x_1, y, t) + \frac{\Delta X_x}{x_2 - x_1},$$
$$X(x, y_2, t) = X(x, y_1, t) + \frac{\Delta X_y}{y_2 - y_1},$$

where the domain is $[x_1, x_2] \times [y_1, y_2]$. To formulate this problem using the MFVM, we assume that the radial CVs are small enough such that each CV lies entirely inside either the solid of water vapour phase, and that there are enough nodes to accurately represent each phase. We then solve the equations using the numerical methods outlined in the previous sections. The number of nodes was gradually increased until spatial convergence was achieved.

To use this model to estimate the macro-scale parameters $D_{xx}$ and $D_{yy}$, we iterate the model through time until a steady-state has been reached. For each node $i$, we then calculate

$$(\Delta XD_{xx})^i = D_{ij} \frac{\partial u}{\partial X},$$

with $l = b$ and $l = v$ for the bound water and water vapour phases respectively. The global parameter $D_{xx}$ can then be approximated as

$$D_{xx} = \frac{1}{N\Delta X} \sum_{i=1}^{N} (\Delta XD_{xx})^i,$$

and similarly for $D_{yy}$, where $N$ is the total number of nodes used in the discretisation.

Fig. 10 gives the estimates of $D_{xx}$ and $D_{yy}$ for micro-scale images of early and late wood, taken from the work of Perré et al. [31]. As can be seen in the figure, the results match extremely well to previous estimates given in [31] using a Lattice Boltzmann method, despite the estimates being produced through completely different modelling procedures.

In Fig. 11 we show how the solution flux in the bound water phase changes under mesh refinement. As the mesh refinement increases, we see the fluxes become more stable and ordered, and the distinction between bound water and water vapour phases becomes easily distinguishable. It can be noted that there is a large increase in flux at the phase boundaries, on the most conductive side. This is due to the interpolative nature of the MFVM. The true solution of the problem has discontinuous fluxes at the phase boundaries, due to differences in diffusion rates. However, since the MFVM imposes continuity in the function value and its derivatives, a sharp gradient is exhibited, leading to large fluxes very close to the boundary edge, as the problem transitions from high flux in the water vapour phase, to low flux in the bound water phase.

To compare results further, we used a regular structured finite-difference code to solve the steady-state equivalent of (18). As Table 2 shows, the error between parameter estimation using the two solution schemes is quite low. This gives further strength to the utility of the MFVM.

Implementation of the MFVM requires a choice of the number of control volumes ($N$), number of spline elements (defined by $Q$ nodes), and the number of quadrature points ($q$) used in approximation of the control volume integrals. To further analyse error, we consider how the solution changes as these parameters change. We first construct a fine mesh solution for the earlywood problem, with $N = 131790$, $Q = 6400$, $q = 32$. We can then measure the convergence of solutions as $N$, $Q$, and $q$ approach their fine mesh values. We use a regular, right angle triangle splining structure, with the number of elements equal to $2(Q - 1)^2$. For a coarse solution $\phi_{\text{coarse}}$, we evaluate the fine mesh solution at the centre of all the control volumes used in the calculation of $\phi_{\text{coarse}}$, and define two error metrics:
Fig. 10. Plot of the estimated $D_{xx}$ and $D_{yy}$ values for earlywood ($\varepsilon = 0.404$) and latewood ($\varepsilon = 0.7882$). The estimates are bounded by the maximum and minimum diffusion rates that are possible, calculated from taking heterogeneous phases completely in series or in parallel. An enlarged plot of the parameter estimates for latewood is also given (bottom left). Structural contours for earlywood (bottom middle) and latewood (bottom right) are also given, with the lumen given in blue, and walls and lumella given in red and yellow. Note that the diffusion values have been scaled through division by $D_v$. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

$$\text{err}_{\text{max}} = \max_{x,y \in \Omega} |\phi_{\text{fine}} - \phi_{\text{coarse}}|,$$

$$\text{err}_{\text{norm}} = \frac{\|\phi_{\text{fine}} - \phi_{\text{coarse}}\|_2}{\|\phi_{\text{fine}}\|_2},$$

Fig. 11. Plots of the flux in the bound water of wood surrounding a water vapour phase, under coarse (left), medium (middle) and fine (right) meshes. The value of $\phi$ is given by the colorbar. As the mesh is refined, the contours gain more distinct shape, and the boundaries between bound water and water vapour phases become clearer. Coarse, medium and fine meshes correspond to values $N = 8256$, $32852$, $131790$ and $Q = 900$, $2500$, $6400$ respectively. (For interpretation of the references to colour in this figure, the reader is referred to the web version of this article.)
where \( \| \cdot \|_2 \) denotes the vector 2-norm. In Tables 3, 4 and 5 we show the error convergence as all three discretisation parameters approach the fine mesh values. As can be seen in the tables, appropriate choice of all three parameters is necessary to ensure high solution accuracy. However, \( Q \) appears to have the highest importance, most dominantly driving error convergence. This appears logical considering the error bounds shown in equation (13).

6. Discussion

Within this paper we have presented a framework for a new meshless numerical scheme, the Meshfree Finite Volume Method. Through the use of spline interpolation, fluxes can be approximated on the edges of nodal control volumes, even when these control volumes are disjoint. By coupling this scheme with an efficient exponential Euler temporal discretisation, and Krylov subspace based methods the MFVM can be applied to the solution of complex non-linear PDEs, particularly in scenarios where classical finite difference and finite volume methods would become computationally intractable.

Within the literature, the majority of meshless methods are focused on application to weak-form PDEs. This may be in part due to the favourability of weak-form models in solid mechanics problems, where meshless methods are often desirable. We believe that the MFVM helps address this gap in the literature by focusing on adaptation of the finite volume method, and application to conservative, strong-form PDEs. This is quite unique and allows for versatile and straightforward implementation of the method to PDEs that can be written in conservative form. As shown in Section 5, the method can be readily implemented on non-rectangular domains, without leading to an expensive system size, or to highly inaccurate solutions. It also outperformed a classical unstructured FVM when applied to a high Péclet problem. We believe that these highlight significant advantages of the method, and its potential value to the field.

The implementation of the MFVM involves the choice of a series of different discretisation parameters, that are not dependent on the PDE model being solved. In Tables 6 and 7 we discuss how these key parameters affect the runtime of the four solver phases: initialisation of matrices, the estimation of \( \lambda \) through the GCV algorithm, surface interpolation, and solution of the resulting system, as well as total solution accuracy.
Table 6
An outline of the effects of the MPVM parameters throughout difference phases of the scheme.

<table>
<thead>
<tr>
<th>Phase</th>
<th>CV radius ((d))</th>
<th>Quadrature ((q))</th>
<th>Number of element nodes ((Q))</th>
<th>Number of data nodes ((N))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interpolation</td>
<td>Not dependent.</td>
<td>Not dependent.</td>
<td>Heavily dependent, as this affects the size of (A).</td>
<td>Heavily dependent, as this affects the size of (A).</td>
</tr>
<tr>
<td>Solver</td>
<td>Mildly dependent (can affect system condition number).</td>
<td>Mildly dependent (can affect system condition number).</td>
<td>Highly dependent (affects system size).</td>
<td>Highly dependent (affects system sparsity).</td>
</tr>
<tr>
<td>Solution accuracy</td>
<td>Moderately dependent (too large causes inaccuracy).</td>
<td>Moderately dependent.</td>
<td>Highly dependent, spatial error converges relative to (Q).</td>
<td>Highly dependent, particularly for complex problems.</td>
</tr>
</tbody>
</table>

Table 7
An outline of key points to consider, when choosing parameters for the MPVM.

<table>
<thead>
<tr>
<th>Phase</th>
<th>Points of consideration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initialisation</td>
<td>When high quadrature is used alongside large (N) the initialisation time can grow quite large. However, for complex problems, this phase could be easily parallelised. Equally, in application, it was found that high quadrature was only necessary for boundary nodes.</td>
</tr>
<tr>
<td>Generalised cross validation</td>
<td>While the GCV minimisation could be slow, (\lambda) does not need to be updated frequently, or found to high accuracy. This meant it could be implemented on a logarithmic searching scale, making the phase runtime insignificant compared to other phases.</td>
</tr>
<tr>
<td>Interpolation</td>
<td>This process was quite efficient due to the interpolant matrix being SPD (allowing for efficient solution methods).</td>
</tr>
<tr>
<td>Solver</td>
<td>The use of the Exponential Euler Method with adaptive timestepping allowed for great runtime improvements as system size increased. For smaller size problems, preconditioned iterative methods may perform better.</td>
</tr>
<tr>
<td>Solution accuracy</td>
<td>A balance needs to be met with parameter choices. If adaptive timestepping is used alongside a sufficiently high number of quadrature points, then (Q) becomes the most significant parameter. However, for this to be true, (N) must also be sufficiently high so as to completely capture the PDE behaviour.</td>
</tr>
</tbody>
</table>

While we believe that this technique represents a novel contribution to the field, there are clear directions for future work. The error bounds presented in (13) appear to hold in application (as shown in Fig. 3). However, a more detailed error analysis of the method is clearly desirable.

Following on from this, all of the results presented in this paper were obtained using \(D^2\) smoothing splines, based off of the HCT basis functions. Clearly this means the method in its current form is only suitable for solutions that can accurately be represented by these splines. However there are many other potential choices for interpolants, which would not affect the general methodology of the MPVM. The analysis of the scheme under a variety of different interpolant choices could potentially provide for improvements in both accuracy and efficiency and is intended for future work.

Another significant choice in the method design was the choice of penalty term. One role of the penalty term is to allow for interpolation in regions with few nodes. However, in theory the term can also be chosen to promote desired behaviours within the solution scheme. For results in this study, a curvature-constraint penalty was applied, as this penalises rapid solution change, and thus promotes stability. Other potential penalty terms such as monotonicity or positivity constraints could also prove effective.

For the advection–diffusion problem presented in section 5.1, all fluxes were calculated at the quadrature points on control volume faces. In more complex problems with time-dependent advection rates, this may cause instability or inaccuracy in the solution. A way to overcome this would be the introduction of flux limiting schemes for adaptive integral calculation. The investigation of the most appropriate flux limiter is a clear direction for future work.

The diversity of applications of mathematical models to real-life scenarios means that a diverse range of numerical methods are needed. We believe that the meshfree finite volume method presents a novel contribution to the field, and is suitable for application to the solution of a variety of strong-form PDEs.

Acknowledgements

This research was funded by Australian Research Council (ARC) grant DP150103675.

We wish to thank Dr Daryl Kempthorne (Queensland University of Technology) for his valued advice on the implementation of HCT element interpolation in MATLAB.
The authors would like to thank the anonymous reviewers for their comments and helpful suggestions that improved the overall quality of the paper.

Appendix A. Affine transformation

Part of the efficiency of the MFVM relies on the use of affine transformations to map points from a general triangular element with vertices \((x_1, y_1), (x_2, y_2), (x_3, y_3)\) and coordinate system \((x, y)\) to the Clough–Tocher element, with coordinate system \((\xi, \eta)\). This transformation is of the form

\[
\begin{align*}
\xi &= a_{11}x + a_{12}y + b_1, \\
\eta &= a_{21}x + a_{22}y + b_2,
\end{align*}
\]

where

\[
\begin{align*}
a_{11} &= \frac{2y_2 - y_1 - y_0}{\Delta}, & a_{12} &= \frac{-2x_2 + x_0 + x_1}{\Delta}, \\
a_{21} &= \frac{-\sqrt{3}(y_1 - y_0)}{\Delta}, & a_{22} &= \frac{\sqrt{3}(x_1 - x_0)}{\Delta}, \\
b_1 &= \frac{x_2(y_1 + y_0) - y_2(x_1 + x_0)}{\Delta}, & b_2 &= \frac{2}{\sqrt{3}} + \frac{\sqrt{3}(x_2(y_1 - y_0) - y_2(x_1 - x_0))}{\Delta},
\end{align*}
\]

and

\[
\Delta = \det \begin{pmatrix} 1 & x_0 & y_0 \\ 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \end{pmatrix}.
\]