<table>
<thead>
<tr>
<th>Title</th>
<th>A Block-Coupled Finite Volume Methodology for Linear Elasticity and Unstructured Meshes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authors(s)</td>
<td>þýCardiff, Philip; Tukovi, Ïeljko; Jasak, Hrvoje; Ivankovic, Alojz</td>
</tr>
<tr>
<td>Publication date</td>
<td>2016-10-15</td>
</tr>
<tr>
<td>Publication information</td>
<td>Computers and Structures, 175 : 100-122</td>
</tr>
<tr>
<td>Publisher</td>
<td>Elsevier</td>
</tr>
<tr>
<td>Item record/more information</td>
<td><a href="http://hdl.handle.net/10197/7940">http://hdl.handle.net/10197/7940</a></td>
</tr>
<tr>
<td>Publisher's statement</td>
<td>þýThis is the author s version of a work that was accepted for publication in Computers and Structures. Changes resulting from the publishing process, such as peer review, editing, corrections, structural formatting, and other quality control mechanisms may not be reflected in this document. Changes may have been made to this work since it was submitted for publication. A definitive version was subsequently published in Computers and Structures (VOL#, ISSUE#, (2016)) DOI: 10.1016/j.compstruc.2016.07.004.</td>
</tr>
</tbody>
</table>

Some rights reserved. For more information, please see the item record link above.
A Block-Coupled Finite Volume Methodology for Linear Elasticity and Unstructured Meshes

P. Cardiff¹,²*, Željko Tuković¹,³, Hrvoje Jasak³,⁴, and A.Ivanković¹

¹University College Dublin, School of Mechanical and Materials Engineering, Belfield, Ireland
²The University of Texas at Austin, Cockrell School of Engineering, Texas, USA
³University of Zagreb, Faculty of Mechanical Engineering and Naval Architecture, Croatia
⁴Wikki Ltd., London W8 7PU, United Kingdom

SUMMARY

The current article presents a new implicit cell-centred Finite Volume solution methodology for linear elasticity and unstructured meshes. Details are given of the implicit discretisation, including use of a Finite Area method for face tangential gradients and implicit non-orthogonal correction. A number of 2-D and 3-D linear-elastic benchmark test cases are examined using hexahedral, tetrahedral and general polyhedral meshes; solution accuracy and efficiency are compared with that of a segregated procedure and a commercial Finite Element software, where the new method is shown to be faster in all cases.

Received …

KEY WORDS: cell-centred Finite Volume method; Finite Area method; Bi-Conjugate Gradient Stabilised method; block-coupled; polyhedral mesh; OpenFOAM

1. INTRODUCTION

In the realm of Computational Solid Mechanics (CSM), the Finite Element (FE) method is ubiquitous; however, methods such as Finite Difference, Discrete Element, Finite Volume (FV), and so-called Meshfree methods, can provide viable alternatives in many applications. In particular, the FV method has been gradually broadening its applicability beyond Computational Fluid Dynamics and gaining momentum in the field of CSM; this may be primarily attributed to its attractively simple and strongly conservative nature, while also having a close relationship with FE methods. To date, the FV method has been applied to a large range of problems in CSM, appearing in a number of distinct forms characterised by different control volume discretisations: cell-centred [1–25], vertex-centred [26–29], and the so-called parametric formulation [30, 31]. To-date, the cell-centred FV approach has typically employed a segregated solution procedure, where the momentum equation is temporally decoupled into components which are solved sequentially, with outer Picard/Fixed-Point

*Correspondence to: University College Dublin, School of Mechanical and Materials Engineering, Belfield, Ireland. E-mail: philip.cardiff@ucd.ie
iterations providing the coupling. This method is extremely memory efficient but can suffer from poor convergence when the inter-displacement-component coupling is strong. To overcome such inadequacies, the current article presents the development of a block-coupled solution methodology, where inter-component coupling is implicitly included in a block matrix. Independent of the current approach, Das et al. [25] have developed a coupled cell-centred FV solution procedure; the current method shares a similarity with the procedure developed by Das et al. [25], but significantly differs in a number of regards, in particular with discretisation of tangential derivatives, and the treatment of boundary conditions. Details of the differences between the two methods are given later within the Numerical Method section.

The article is constructed as follows: Section 2 outlines the mathematical model, derived from the governing momentum equation and Hookean constitutive relation, where no distinction is made between the initial and deformed configurations. The newly developed implicit cell-centred FV discretisation is presented in Section 3; the control volume face normal derivative terms are discretised using standard central differencing, and the face tangential derivative terms are discretised using the Finite Area method. Details are given of the inclusion of a larger implicit computational molecule within 3-D unstructured polyhedral code, and implementation in open-source C++ library OpenFOAM (foam-extend-3.1). Section 4 presents the application of the method to three benchmark test cases, where the efficiency and accuracy of the method is compared with that of segregated FV approaches, and a commercial FE software.

2. MATHEMATICAL MODEL

Neglecting inertia and body forces for clarity, the conservation of linear momentum for an arbitrary body of volume $\Omega$ bounded by surface $\Gamma$ with unit normal $n$ is given in integral form as:

$$\int_\Omega \nabla \cdot \sigma \, d\Omega = \oint_\Gamma n \cdot \sigma \, d\Gamma = 0$$  \hspace{1cm} (1)

where the Cauchy stress tensor $\sigma$ for a linear elastic body is given by Hooke’s law:

$$\sigma = \mu \nabla \mathbf{u} + \mu \nabla \mathbf{u}^T + \lambda \text{tr}(\nabla \mathbf{u}) \mathbf{I}$$  \hspace{1cm} (2)

The Lamé coefficients $\mu$ and $\lambda$ relating to the Young’s modulus of elasticity, $E$, and the Poisson’s ratio, $\nu$, are given respectively as:

$$\mu = \frac{E}{2(1+\nu)}$$  \hspace{1cm} (3)

$$\lambda = \begin{cases} \frac{\nu E}{(1+\nu)(1-\nu)} & \text{for plane stress} \\ \frac{\nu E}{(1+\nu)(1-2\nu)} & \text{for plane strain and 3-D} \end{cases}$$  \hspace{1cm} (4)

Inclusion of transient effects and body force terms is trivial; the use of more complex constitutive relations is also possible, but may require modifications to the solution procedure for nonlinear materials.
The mathematical model is found by substituting the constitutive relation (Eq. 2) into the governing equation (Eq. 1):

$$\int_{\Gamma} \mathbf{n} \cdot (\mu \nabla \mathbf{u} + \mu \nabla \mathbf{u}^T + \lambda \text{tr}(\nabla \mathbf{u}) \mathbf{I}) \, d\Gamma = 0$$

(5)

where no distinction is made between the initial and deformed configurations i.e. small strains and small rotations are assumed.

3. NUMERICAL METHOD

The mathematical model presented in the preceding section is now discretised in a fully coupled implicit manner using the cell-centred FV method, providing a discrete approximation of the previously presented exact integral. The discretisation procedure is separated into two distinct parts: discretisation of the solution domain and discretisation of the governing equations. If temporal effects were considered, time would also be discretised into a finite number of time increments, where the mathematical model is solved in a time-marching manner.

3.1. Solution Domain Discretisation

The solution domain space is divided into a finite number of convex polyhedral cells bounded by polygonal faces. The cells do not overlap and fill the space completely. A typical control volume is shown in Figure 1, with the computational node $P$ located at the cell centroid, the cell volume is $V_P$, $N$ is the centroid of a neighbouring control volume, face $f$ has face area vector $f$, vector $d_f$ joins $P$ to $N$ and $r$ is the positional vector of $P$. Note that in contrast to standard FE methods, no distinction is made between different cell volume shapes, as all general polyhedra (e.g. tetrahedra, hexahedra, triangular prism, dodecahedra, etc.) are discretised in the same general fashion: this allows greater freedom during the often troublesome mesh generation phase.

Figure 1. General polyhedral control volume (adapted from [2, 32])
3.2. Equation Discretisation

The standard cell-centred FV discretisation approach for solid mechanics, as previously presented e.g. [3, 6, 9], partitions the surface force (diffusion term) into implicit and explicit components; the resulting linear system is then solved in a staggered/segregated manner where outer Picard/Fixed-Point iterations provide the necessary coupling. In contrast, the current method discretises the entire term in a fully implicit manner where only one solution of the final linear equation system is required. To facilitate this, the surface traction \( T = n \cdot \sigma \) is decomposed into normal \( (T_n) \) and tangential \( (T_t) \) components [12]:

\[
\int_{\Gamma} n \cdot \left[ \mu \nabla u + \mu \nabla u^T + \lambda \text{tr} (\nabla u) I \right] \, d\Gamma = \int_{\Gamma} T \, d\Gamma \\
= \int_{\Gamma} \left( mn \cdot T + (I - mn) \cdot T \right) \, d\Gamma \\
= \int_{\Gamma} \left(2\mu + \lambda\right) n \cdot \nabla u_n + \lambda n \text{tr} (\nabla_t u_t) + \mu n \cdot \nabla_t u_t + \mu \nabla_t u_n \, d\Gamma \\
(6)
\]

where \( \nabla_t = (I - nn) \cdot \nabla \) designates a tangential derivative; quantities in bold font are vectors or tensors; subscripts \( n \) and \( t \) represent the normal and tangential components of a vector, respectively.

The discretisation of the normal and tangential face-derivatives are now given separately.

Normal Derivative Terms The surface integrals may be replaced by a sum over the faces of a control volume, and assuming a linear variation of displacement \( u \) across the control volume, a second-order FV discretisation may be applied. The normal derivative terms \( (n \cdot \nabla) \) on face \( f \) are discretised as follows:

\[
\int_{\Gamma} (2\mu + \lambda) n \cdot \nabla u_n \, d\Gamma \approx \sum_f \left(2\mu_f + \lambda_f\right) \left| \Delta_f \right| (n_f n_f) \cdot \left( \frac{u_n^f - u_n^G}{|d_f|} \right) \\
+ \sum_f \left(2\mu_f + \lambda_f\right) (n_f n_f) \cdot \left[ k_f (\nabla_t u_n)_f \right] \\
(7)
\]

where subscript \( f \) refers to values at control volume faces, and the vector \( \Delta_f = \frac{d_f}{d_f + \Gamma_f} |\Gamma_f| \) [33]. The first term on the right-hand side assumes orthogonality of the mesh and discretises the normal derivative using central differencing. The second term on the right-hand side, which contains the tangential gradient \( \nabla_t u_n \), corrects for any mesh non-orthogonality, where the correction vector is \( k_f = \Gamma_f - \Delta_f \). This corresponds to the over-relaxed non-orthogonal correction method as described by Jasak [33]; however, in contrast to the previous employed iterative deferred/lagged correction method, the current implementation discretised the corrective tangential derivative term in a fully implicit manner, as will be described in the subsequent section.
The normal derivative of the tangential component $n \cdot \nabla u_t$ is discretised in a similar manner to that of the normal derivative of the normal component $n \cdot \nabla u_n$:

\[
\int_{\Gamma} \mu (n \cdot \nabla u_t) \cdot \frac{u^N - u^F}{|d_f|} \, d\Gamma_f \approx \sum_f \mu_f | \Delta_f | \left( \mathbf{I} - n_f n_f \right) \cdot \left( \frac{u^N - u^F}{|d_f|} \right) + \sum_f \mu_f \left( \mathbf{I} - n_f n_f \right) \cdot \left[ k_f \cdot (\nabla \cdot u_t)_f \right]
\]

where the tensor $(n_f n_f)$ has been replaced by $(I - n_f n_f)$, referring to the tangential component of displacement.

**Tangential Derivative Terms** As with the normal derivative terms above, the surface integral over tangential derivative terms is converted to a sum over control volume faces:

\[
\int_{\Gamma} [\mu \nabla \cdot u_n + \lambda n \cdot (\nabla \cdot u_t)] \, d\Gamma = \sum_f [\mu_f (\nabla \cdot u_n)_f + \lambda_f n_f \cdot (\nabla \cdot u_t)_f] |\Gamma_f|
\]

The tangential face derivates, $\nabla \cdot u_n$ and $\nabla \cdot u_t$, are then calculated directly on face $f$ using a face-Gauss/Finite Area (FA) method [34–37]:

\[
(\nabla \cdot u_n)_f = \frac{1}{|\Gamma_f|} \sum_e m_e (u_n)_e L_e
\]

where subscript $e$ refers to an edge of face $f$, $L_e$ is the length of edge $e$ (see Figure 2), and the summation is performed over all edges enclosing face $f$. This approach to calculation of tangential face gradients is a generalisation of the method used by Demirdžić et al. [1, 38] on structured meshes. The edge-centre unit bi-normal vector $m_e$ is calculated as follows:

\[
m_e = \hat{e} \times n_f
\]

where $\hat{e}$ is the unit vector parallel with the edge $e$, and operator $\times$ designates a cross product.

![Figure 2. Polygonal control area](image-url)
The edge-centre displacement components, \((u_n)_e\) and \((u_t)_e\), are approximated as the average of the edge end-points, assuming a linear variation:

\[
(u_e) = \frac{1}{2} [(u)_{e1} + (u)_{e2}]
\]

(12)

where the edge end-point displacements, \((u)_{e1}\) and \((u)_{e2}\), may be approximated in terms of the neighbouring cell-centre values using a weighted least squares interpolation with linear fit function [13, 34]:

\[
(u_{ep})_i = u_{i0} + C_i \cdot (r_i - r_{o0})
\]

(13)

where subscript \(j\) indicates the so-called point-cell neighbours (cells that contain the current point) and boundary face centres (boundary faces that contain the current point), and \(n\) is the total number of point-cell neighbours around the point; the reference position \(r_{o0}\) and reference field value \(u_{i0}\) are calculated directly as the weighted average of cell-centre and boundary face positions and cell-centre and boundary face field values. The point-cell weights \(\omega_j\) can be calculated in a number of ways: in the current study an arithmetic average is used, alternatively the weights could be based on the inverse of the distance or its square. \(C_i\) is the unknown least squares coefficient vector (three components):

\[
C_i = [(X^T W X)^{-1} X^T W] \cdot u_{\Delta i}
\]

(14)

where \(u_{\Delta i}\) is the vector consisting of the quantity \(u_i - u_{i0}\) for all cells in the interpolation stencil of point \(i\), \(W\) is the diagonal matrix consisting of weighting factors for all cells in the interpolation stencil of point \(i\) and \(X\) is the \(n \times 3\) matrix whose row \(j\) is position vector \(r_{ij} - r_{o0}\) of the cell \(j\) in the interpolation stencil of the point \(i\). This FA method with least squares cell-centre-to-point interpolation has been presented previously by Tuković et al. [34] for explicit calculation of face tangential derivatives; the current articles outlines an implicit implementation of this method.

The discretised form of the tangential derivative of normal component term is:

\[
\int_{\Gamma} \mu \nabla_e \cdot u_n \, d\Gamma \approx \sum_f \sum_j \sum_e m_e \sum_{pc} \frac{1}{2} \omega_{pc} u_{pc}^e \cdot L_e
\]

\[
\approx \sum_f \sum_e \sum_{pc} \frac{1}{2} \mu_j \omega_{pc} L_e \cdot m_e \cdot u_{pc}^e
\]

\[
\approx \sum_f \sum_e \sum_{pc} \frac{1}{2} \mu_j \omega_{pc} L_e \cdot (m_e \cdot n_f) \cdot u_{pc}^e
\]

(15)
In a similar fashion, the tangential derivative of the displacement tangential component can be discretised, where it is noted that $m_e \cdot n_f \equiv 0$ and hence $m_e \cdot (I - n_f n_f) \equiv m_e$:

$$
\int_{\Gamma} \lambda \ n \ tr(\nabla_t u_t) \ d\Gamma \approx \sum_j \lambda_j n_f \ tr \left( \sum_{e,pc} \frac{1}{2} \omega_{pc} u_{pc}^e L_e \right)
$$

$$
\approx \sum_j \sum_e \sum_{pc} \frac{1}{2} \lambda_j \omega_{pc} L_e n_f \ tr (m_e u_{pc}^e)
$$

$$
\approx \sum_j \sum_e \sum_{pc} \frac{1}{2} \lambda_j \omega_{pc} L_e n_f \ tr (m_e (I - n_f n_f) \cdot u_{pc})
$$

$$
\approx \sum_j \sum_e \sum_{pc} \frac{1}{2} \lambda_j \omega_{pc} L_e (m_e n_f)^T \cdot u_{pc}
$$

(16)

where it should be noted that the magnitude of the face area vector $\Gamma_f$ does not feature in either of the final discretised tangential terms but instead the edge lengths $L_e$.

The non-orthogonal correction terms that appear in Equations 7 and 8, which contain tangential derivatives, are discretised in the same manner as Equations 15 and 16. It should be noted that discretisation is second order in space and strongly conservative. Additionally, the resulting sparse solution matrix will be symmetric in structure but the coefficients will be asymmetric, potentially leading to higher memory and computation costs; however, as will be shown in the test cases section, the resulting method is shown to be efficient in terms of memory and computation time.

At this point it is worth mentioning differences with the method developed by Das et al. [25] for discretisation of face tangential gradients. As described, the current approach calculates tangential derivatives of displacement directly at control volume faces; in contrast, the approach of Das et al. [25] calculates tangential derivatives at control volume centres and linearly interpolates these gradients to the control volume faces. Through interpolation of cell-centre gradients, the computational molecule includes two levels of neighbour cells (so-called 2nd neighbours) around a given control volume, whereas in the current approach a smaller computational molecule is required where neighbours which share a mesh point with the current control volume are included.

In addition to an increase in accuracy, the smaller computational molecule results in a less dense matrix within the resulting linear system. It should be noted that Das et al. [25] incorrectly refers to the discretisation procedure as linearisation of the stress tensor; under the adopted assumption of a linear elastic material undergoing small strains and small rotations, the governing momentum equation is linear in displacement and as such no linearisation is necessary. The two terms containing tangential gradients, $\nabla \cdot (\nabla u_t)^T$ and $\nabla \cdot tr(\nabla u_t) I$, introduce inter-component coupling within the momentum equation but do not introduce any nonlinearity.

Boundary Conditions Traditionally cell-centred FV methods incorporate boundary condition equations directly into the governing discretised equation for control volumes adjacent to the boundary; this method was initially trialled for the current methodology; however, implicit calculation of tangential gradients along near-boundary faces (those with points on the boundary but not on the boundary themselves) becomes excessively complicated. Consequently, boundary
conditions are included as additional equations in the current methodology, where boundary face displacements are additional unknowns. This results in one discretised conservation equation for each control volume and one additional discretised boundary condition equation for each boundary face. Although not described in detail, the treatment of boundary conditions by Das et al. [25, 39] differs from the current approach in that boundary face values are not included as additional unknowns. As details of the implementation are not given, it is unclear if the method proposed by Das et al. [25] maintains the discretisation order of accuracy at boundary and near-boundary faces.

The discretisation procedure outlined in the previous section is applied to the faces (internal and boundary) of all mesh control volumes, where boundary-face values are implicitly included just as cell-centre values. This section outlines the implementation of the discrete displacement, traction and symmetry plane boundary condition equations.

**Displacement** The displacement boundary condition, a Dirichlet condition, may be constant in time or time-varying and fixes the boundary-face centre value of displacement, \( u_b \). The discrete displacement boundary condition vector equation is:

\[
\mathbf{I} \cdot \mathbf{u} = \mathbf{U}_b
\]

(17)

where \( \mathbf{u} \) is the boundary-face centre unknown displacement, \( \mathbf{I} \) is the identity tensor, and \( \mathbf{U}_b \) is the user specified boundary-face centre displacement.

**Traction** The traction boundary condition, a Neumann condition, is a constant in time or time-varying user specified traction; the traction on a boundary face is written as:

\[
T_b|\Gamma_b| = \int_{\Gamma_b} (2\mu + \lambda) \mathbf{n} \nabla u_n + \lambda \mathbf{n} \cdot \nabla \mathbf{u}_t + \mu \mathbf{r} \nabla u_n + \mu \mathbf{r} \nabla \mathbf{u}_t + \mu \mathbf{r} \nabla \mathbf{u}_n\ d\Gamma
\]

(18)

where the normal and tangential derivative terms are discretised in the same manner as for internal cells, where the main difference is that the central differencing approximation becomes one-sided; \( T_b \) is the user specified boundary traction. In practice, the negative of this equation is used to ensure that the diagonal coefficients are negative, as is the case for the control volume equations.

**Symmetry Plane** The symmetry plane condition, a mixed Dirichlet-Neumann condition, specifies by definition a zero normal displacement and a zero normal gradient of tangential displacement:

\[
\begin{align*}
    u_n &\equiv 0 \\
    \mathbf{n} \cdot \nabla \mathbf{u}_t &\equiv 0
\end{align*}
\]

(19)

where the condition is enforced at the centre of a boundary face. The discretised form of the normal displacement equation is:

\[
(n_n, n_t) \cdot \mathbf{u} = 0
\]

(20)
where \( n_b \) is the boundary face normal, and \( u \) is the unknown boundary face displacement. The tangential equation becomes:

\[
\left| \Delta_n \right| (I - n_b n_b) \cdot \left( \frac{u_b - u^p}{d_b} \right) + (I - n_b n_b) \cdot [k_b \cdot (\nabla_t u)_b] = 0
\] (21)

where the tangential derivative in the non-orthogonal correction term is calculated at the boundary face and discretised in a similar manner as for internal faces. Equation 20 only refers to the normal component of displacement whereas Equation 21 only refers to the tangential component of displacement; as such, these two vector equations can be summed to produce the final discretised boundary condition vector equation for each boundary face on a symmetry plane.

### 3.3. Solution Procedure

The final discretised form of the linear momentum equation for each control volume \( P \) can be arranged in the form of \( N_i \) linear algebraic equations:

\[
a^P_i \cdot u^P_i + \sum_{n_{Pj}} a^N_{ij} \cdot u^N_j = b^P_i
\] (22)

where \( n_{Pj} \) is the number of neighbour control volume and boundary faces in the current implicit molecule, \( a^P_i \) is the central tensor coefficient, \( a^N_{ij} \) are the neighbour tensor coefficients representing interactions with neighbour cell-centred unknowns or boundary-face-centred unknowns, and \( b^P_i \) is the source vector contribution which is zero in all cases here. If included, body forces and inertia terms would contribute to the source vector.

An additional \( N_b \) linear equations arise from the discrete boundary conditions, one for each boundary face centre:

\[
a^b_P \cdot u^P_P + \sum_{n_{bP}} a^N_{bP} \cdot u^N_P = b^b_P
\] (23)

where, similar to the control volume equations, \( a^b_P \) is the central tensor coefficient, \( a^N_{bP} \) are the neighbour tensor coefficients, and \( b^b_P \) is the source vector contribution; here the source will only contain contributions from traction boundary conditions and non-zero displacement boundary conditions.

The algebraic linear equations described above (Equations 22 and 23) are then assembled for all control volumes and boundary face centres forming a system of linear equations:

\[
[A] \begin{bmatrix} \phi \end{bmatrix} = [b]
\] (24)

where \([A]\) is a sparse \( N \times N \) matrix, with \( N = N_i + N_b \) being the total number of computational points. The tensorial coefficients \( a^P_i \) are on the diagonal of matrix \([A]\) and tensorial coefficients \( a^N_{ij} \) form the matrix off-diagonal. The solution vector \([\phi]\) contains the unknown cell-centre and boundary-face-centre displacement vectors, and \([b]\) is the source vector.

Given the nested summations involved in the specification of the matrix coefficients, the calculation of the final matrix coefficients are given here, for clarity, in algorithmic form, where the coefficients stemming from the normal and tangential derivatives are assembled separately. The normal coefficients are assembled as indicated in Algorithm 1.
Algorithm 1 Assembling normal derivate coefficient contributions within the block matrix

1: for all internal and boundary faces do:
2:   for both cells straddling the internal face OR boundary face and adjacent cell do
3:     coeff ← \((2\mu + \lambda)(n_j n_f)\frac{\Delta}{|\Gamma_f|} + \mu(I - n_j n_f)\frac{\Delta}{|\Gamma_f|}\)
4:     \(a_P \leftarrow a_P - \text{coeff.}\)  \(\triangleright\) Add contribution to matrix diagonal
5:     \(a_N \leftarrow a_N + \text{coeff.}\)  \(\triangleright\) Add contribution to matrix off-diagonal
6:   end for
7: end for

The coefficients arising from the tangential derivatives are assigned using a series of nested loops, shown in Algorithm 2.

Algorithm 2 Assembling tangential derivate coefficient contributions within the block matrix

1: for all cells do:
2:   for all internal and boundary faces of the current cell do
3:     for all edges of the current face do
4:       for all neighbour cells/boundary-faces of the end-points of current edge do
5:         coeff ← \(\frac{1}{2}\mu_f \omega_{pc} L_c(m_e n_f) + \frac{1}{2}\lambda_f \omega_{pc} L_c(m_e n_f)^T\)
6:         if non-orthogonal correction required then
7:           coeff ← coeff + \(\frac{1}{2}n_j n_f(2\mu_f + \lambda_f)\omega_{pc} L_c(k_f \cdot m_e)\)
8:           + \(\frac{1}{2}(I - n_j n_f)\mu_f \omega_{pc} L_c(k_f \cdot m_e)\)
9:         end if
10:        if the neighbour cell is the current cell then
11:           \(a_P \leftarrow a_P + \text{coeff.}\)  \(\triangleright\) Add contribution to matrix diagonal
12:        else
13:           \(a_N \leftarrow a_N + \text{coeff.}\)  \(\triangleright\) Add contribution to matrix off-diagonal
14:        end if
15:       end for
16:     end for
17:   end for
18: end for

Algorithms 1 and 2 complete the assembly of the \(N_i\) control volume discrete governing equations; the linear system is finalised once the additional \(N_b\) discrete boundary condition equations are added in a similar manner. In contrast to a segregated solution procedure, the source vector in the current formulation contains only contributions from the boundary conditions, and can be thought of as the driving force. If inertial and body force effects were included, they would also contribute to the source vector.

To illustrate the construction of the block matrix coefficients above, the coefficients for an internal cell in a simple 2-D uniform structured quadrilateral mesh are shown in Figure 3. It is interesting to note that the normal derivative terms contribute only to the central cell coefficient and the coefficients of the neighbour cells connected through a face \(i.e.\) top, left, right and bottom cells; whereas, the tangential derivative terms contribute exclusively to the far point-neighbour cell coefficients \(i.e.\) top-left, top-right, bottom-left and bottom-right cells. This is only the case for a fully uniform mesh; in non-uniform meshes the tangential derivative terms can contribute to all cells within the computational molecule, where the contributions do not fully cancel out.
employs zero fill-in and hence will be referred to as ILU(0). The combination of direct, for incomplete LU decomposition (ILU) preconditioning have been employed; the ILU preconditioner the current work, both a LU decomposition direct solver and an iterative BiCGStab method with procedures result in some level of matrix sparsity pattern during the solution, whereas direct solver sparse non-diagonally dominant linear systems [40]; however, the memory and calculation costs of direct methods quickly become impractical as the problem size increases. It should be noted that iterative solvers preserve the matrix sparsity pattern during the solution, whereas direct solver procedures result in some level of matrix fill-in producing much larger memory footprints. In the current work, both a LU decomposition direct solver and an iterative BiCGStab method with incomplete LU decomposition (ILU) preconditioning have been employed; the ILU preconditioner employs zero fill-in and hence will be referred to as ILU(0). The combination of direct, for small/medium problems, and iterative methods, for medium/large problems, should provide an

As a consequence of the tangential derivative terms contributing solely to the off-diagonal coefficients, the sparse $N \times N$ matrix will not be diagonally dominant, in contrast to the segregated methodology. This lack of matrix diagonal dominance has implications for the choice of linear solver for the linear system of equations: standard preconditioned Conjugate Gradient (CG) methods may not guarantee convergence, and other methods must be considered. In general, there are two choices for the solution of the linear system: direct methods or iterative methods. Candidate direct methods are Gaussian elimination, LU decomposition, or multi-frontal methods, whereas candidate iterative methods are Bi-Conjugate Gradient Stabilised (BiCGStab), or Generalised Minimal Residual (GMRes), where appropriate choice of preconditioner can also greatly affect the efficiency. In general, direct methods are currently the fastest technique to solve small/medium sparse non-diagonally dominant linear systems [40]; however, the memory and calculation costs of direct methods quickly become impractical as the problem size increases. It should be noted that iterative solvers preserve the matrix sparsity pattern during the solution, whereas direct solver procedures result in some level of matrix fill-in producing much larger memory footprints. In the current work, both a LU decomposition direct solver and an iterative BiCGStab method with incomplete LU decomposition (ILU) preconditioning have been employed; the ILU preconditioner employs zero fill-in and hence will be referred to as ILU(0). The combination of direct, for small/medium problems, and iterative methods, for medium/large problems, should provide an

**Figure 3.** Block-coupled matrix coefficients for cell $P$ in a simple 2-D structured mesh
efficient solution procedure. Although not implemented in the current study, an Algebraic Multi-
Grid (AMG) employing BiCGStab or GMRes should provide considerable speed-ups for larger
problems: this will be the focus of future work.

Regardless of linear solver type, the block-coupled solver uses considerably more memory than a
segregated approach: in a 3-D simulation, each entry in the block matrix is a $3 \times 3$ tensor; this is in
contrast to the scalar coefficient employed in a segregated approach. Additionally, the larger implicit
molecule required by the coupled approach decreases the sparsity of the matrix in comparison to
the segregated approach. As a result, it would be expected that the memory required by the coupled
approach would be an order of magnitude greater than that of the segregated approach. It should be
kept in mind that the segregated approach was developed when computers possessed prohibitively
small quantities of memory, whereas modern machines, in particular distributed memory super
clusters, have considerably larger memory-to-CPU ratios.

Although not the focus of the current study, the extension of the current methodology to
nonlinearity (geometric, material or loading) is worth briefly discussing: the segregated procedure
naturally extends to nonlinearity as inter-component coupling is included in a deferred correction
manner, consequently the inclusion of nonlinear terms requires no modification to the solution
methodology. In contrast, the current coupled methodology would require outer iterations over
the linear system to include nonlinearity. The use of Picard/Fixed-Point iterations would require
the least complexity in terms of implementation; however, the Newton-Raphson method (or
Newton type methods) would be the ideal candidate for an efficient nonlinear algorithm, based
on experiences in nonlinear FE methods.

3.4. Notes on Implementation: Sparse Matrix Addressing

Within an unstructured mesh code, the storage and addressing of the sparse matrix greatly affects
the efficiency of the implemented procedures. Recently developed block-coupled matrix tools in
OpenFOAM [41] provide many of the tools required for implementation of the current discretisation
and solution procedure, apart from one deficiency: as is the case in typical unstructured cell-centred
FV codes, a compact implicit molecule is employed where only near neighbours cells, connected
through a face, are implicitly coupled within the sparse matrix structure. The discretisation
developed here requires all point-neighbours (cells sharing a point with the current cell) to be
included implicitly. Figure 4 compares the standard cell-centred FV implicit molecule with the
required extended molecule; extended molecules for triangular and polygonal meshes are shown in
Appendix A. Consequently, an extended-molecule sparse matrix addressing has been developed to
allow implicit coupled inclusion of point-neighbour cell contributions.

There are a number of different methods proposed for storage of the sparse matrices
generated from FE/FV discretisations; here, the so-called Lower-Diagonal-Upper (LDU) format,
as implemented in OpenFOAM, is employed. The LDU format stores the sparse matrix in three
arrays: the diagonal stores the matrix diagonal ($A_{ij}$ for $i = j$); the lower stores the matrix lower
portion ($A_{ij}$ for $i > j$); and the upper stores the matrix upper portion ($A_{ij}$ for $i < j$). The LDU
format allows implementation of highly efficient iterative linear solvers, as multiplication of the
matrix with the solution vector is straight-forward. Addressing of the off-diagonal arrays (upper and
lower) is determined by the implicit computational molecule. In the case of the compact molecule,
the length of the off-diagonal arrays is equal to the total number of internal faces in the mesh
A BLOCK-COUPLED FINITE VOLUME METHODOLOGY FOR LINEAR ELASTICITY

i.e. a given cell with \( N_{\text{faces}} \) is implicitly coupled to \( N_{\text{faces}} \) other cells and hence has \( N_{\text{faces}} \) off-diagonal coefficients. The mesh face number then determines the addressing of the off-diagonal coefficients. By considering the internal mesh faces as *implicit bonds*, the method can be modified to use an extended molecule. For every cell, *implicit bonds* are defined connecting to each neighbour cell (or neighbour boundary face) within the implicit computational molecule of the current cell. This new *implicit bonds* connectivity can then be used in an analogous manner to the mesh faces in the standard approach, thus completely defining the addressing of the extended sparse matrix. Consequently, no modification is necessary to the implemented OpenFOAM block linear equation solvers, as they only require the sparse matrix, the sparse matrix addressing, and the source vector.

![Figure 4. Comparison of compact & extended implicit computational molecule](image)

To further illustrate the differences between the previous sparse matrix addressing format and the new extended structure, the sparse matrices for a simple 2-D square plate, meshed with \( 3 \times 3 \) square cells, are compared in Figure 5. The \( D \) entries signify a diagonal coefficient, the \( U \) entries signify an upper non-zero off-diagonal coefficient, and the \( L \) entries signify a lower non-zero off-diagonal coefficient. It is clear to see that there are three main differences between the previous segregated compact approach and the new extended approach:

- The compact approach has \( N_i \) unknowns resulting in a \( N_i \times N_i \) matrix; the extended approach has \( N = N_i + N_b \) unknowns with a \( N \times N \) matrix;
- Non-zero off-diagonal coefficients are only present for each *face-neighbour* in the compact approach, whereas for each *point-neighbour* in the extended approach;
- Each coefficient is a scalar in the segregated approach, whereas each coefficient is a \( 3 \times 3 \) tensor in the coupled approach.

As a result, the new extended approach clearly has considerably more computer memory requirements, and when moving to 3-D, the differences increase further; however, it should be kept in mind that the method is comparable to that of a standard FE block-coupled implementation. This particularly highlights the level of memory efficiency achieved by segregated approaches. For each of the test cases examined in the subsequent section, the memory requirements, in addition to the wall-clock time (entire time taken by the computer to complete the task), are presented.

4. TEST CASES

The accuracy and efficiency of the developed block-coupled linear elastic methodology is examined for three separate representative test cases. The first test case, that of a slender 2-D cantilever...
undergoing bending, has an analytical solution and highlights the efficiency of the developed method in cases where inter-component coupling is strong. The second and third cases, elastic benchmarks as proposed by Demirdžić et al. [3], are 3-D problems requiring relatively fine grids to obtain accurate solutions. For each case, the new coupled procedure is compared with a segregated FV procedure, the Demirdžić et al. [3] benchmark, and a commercial FE software; additionally, the effect of control volume shape is examined, comparing hexahedra, tetrahedra and general polyhedra meshes.

**Slender Cantilever in Bending**

The test case geometry, shown in Figure 6(a), consists of a rectangular beam $2 \times 0.1$ m with a Young’s modulus $E$ of 200 GPa and a Poisson’s ratio $\nu$ of 0.3. Five separate systematically refined uniform quadrilateral meshes are considered: 500, 4,500, 12,500 and 50,000 cells; the coarsest mesh is shown in Figure 6(b). The beam is fixed on the left boundary, $U_b = (0 \ 0)$, and a traction of $T_b = (0 \ 1)$ MPa is applied to the right boundary; the top and bottom boundaries are traction-free. Plane strain conditions are assumed.
The analytical deflection of the right-end of the beam is given as [43]:

\[ \Delta = \frac{PL^3}{3(E/1-\nu^2)I} = 0.01456 \text{ m} \]  \hspace{1cm} (25)

where \( P = 0.1 \text{ MN} \) is the applied load, \( L = 2 \text{ m} \) is the length of the beam, and \( I = \frac{bh^3}{12} = 0.13 \) is the second moment of area of the beam about its bending axis.

Using the difference between the predicted displacement and the analytical solution as a metric, the accuracy and order of accuracy of the current method may be assessed. Figure 7 compares the results from the new procedure with those from two segregated FV methods, and commercial FE software Abaqus (version 6.11-1) [44]. The first segregated FV method (Segregated 1 in Figure 7) uses internal face tangential gradients interpolated from neighbour cell-centre gradients, and boundary face tangential gradients extrapolated from the internal domain, as detailed by Jasak and Weller [6]; the second segregated FV method (Segregated 2 in Figure 7) employs the same discretisation as the new coupled procedure where the face tangential gradients are directly calculated using the FA method. The FE solution in Abaqus uses bi-linear shape functions with reduced-order integration as fully integrated linear elements are known to be erroneously stiff in bending. Quadratic elements, if used, would be expected to achieve greater accuracy, albeit at a considerably greater computational cost.

Examining Figure 7, it can be seen that the errors for all the methods reduce at an approximately second order rate, as is expected. Comparing the coupled method to the segregated method with an equivalent discretisation (Segregated 2), the solutions are almost identical, where the slight differences are due to the iterative solution tolerance in the segregated method being too loose. In this case, the new face tangential gradient discretisation (Coupled and Segregated 2) is more accurate solution than the method where the gradient is interpolated from adjacent cell centres (Segregated 1); furthermore, this face tangential gradient method is more accurate than the FE solution for each mesh examined. An important point to note is that the FV methods do not suffer erroneous locking, as experienced by standard fully integrated linear FE methods. The meshes examined here have cell
aspect ratios of unity; although not shown here, it has been found the use of large aspect ratio cells produces a stiffer response than for an equivalent mesh with aspect ratio close to unity, as is the case with the standard FE method. Even in the case of higher aspect ratio meshes, the reduction in error remains second order and the true solution is asymptotically approached.

The models have been solved in double precision using 1 CPU core (2.4 GHz Intel Ivy Bridge cores); the execution times for the newly developed coupled method are compared with those of the segregated FV and FE approaches in Table 1. All models have been solved in one quasi-static time-step.

For the coupled approach, the linear system has been solved using a direct sparse LU factorisation method as implemented in open-source C++ library Eigen [45], where an OpenFOAM wrapper class has been written to handle the communication. The segregated FV approaches employ a CG method with incomplete Cholesky preconditioning [46]; for this test case, it was found that the segregated solution required a relatively tight solution tolerance of \(10^{-8}\) to produce equivalent results to the coupled method. The finest meshes were not run for the segregated approaches due to the large expected wall-clock times; it should be noted that a geometric multigrid method (e.g. [3]), if used, would be expected to substantially reduce the required time of a segregated method.

<table>
<thead>
<tr>
<th>Mesh</th>
<th>Coupled Time</th>
<th>Coupled Memory</th>
<th>Segregated1 Time</th>
<th>Segregated1 Memory</th>
<th>Segregated2 Time</th>
<th>Segregated2 Memory</th>
<th>Abaqus Time</th>
<th>Abaqus Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>0.08</td>
<td>11</td>
<td>58</td>
<td>20</td>
<td>134</td>
<td>24</td>
<td>3</td>
<td>40</td>
</tr>
<tr>
<td>4500</td>
<td>0.5</td>
<td>50</td>
<td>384</td>
<td>27</td>
<td>1049</td>
<td>41</td>
<td>4</td>
<td>113</td>
</tr>
<tr>
<td>12500</td>
<td>1.4</td>
<td>140</td>
<td>1387</td>
<td>43</td>
<td>3171</td>
<td>74</td>
<td>5</td>
<td>197</td>
</tr>
<tr>
<td>50000</td>
<td>6</td>
<td>570</td>
<td>4737</td>
<td>112</td>
<td>11583</td>
<td>225</td>
<td>16</td>
<td>881</td>
</tr>
<tr>
<td>200000</td>
<td>36</td>
<td>2500</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>73</td>
<td>1800</td>
</tr>
</tbody>
</table>

Table 1. Slender cantilever: wall-clock time (in s) and maximum memory usage (in MB).
From Table I, it is clear to see that the developed coupled approach far outperforms the segregated approach showing speedups of over three orders of magnitude. It should be noted that this style of test case is a worst case scenario for a segregated approach, due to the dominant inter-component coupling. When compared with the FE procedure, the new approach is at least twice as fast for the examined meshes. Inspecting computer memory requirements, the coupled approach uses at least an order of magnitude more memory than the equivalent segregated approach; as previously discussed this is due to a combination of the extended implicit molecule, the inclusion of boundary face unknowns, and the additional memory requirements of a direct solver due to matrix fill-in.

Comparing the coupled and the FE methods, the memory requirements are approximately the same, with the coupled method using slightly less; this is to be expected as the FE solution procedure also uses a block coupled approach with a direct solver.

Out-of-Plane Bending of an Elliptic Plate

This 3-D test case consists of a thick elliptic plate with a centred elliptic hole (Figure 8); a constant pressure of $1 \text{ MPa}$ is applied to the upper surface, and the outer surface is fully clamped. The case has been described by the National Agency for Finite Element Methods and Standards (NAFEMS) [47], and in the context of the FV method has been examined and benchmarked in detail by Demirdžić et al. [3]. The assumed mechanical properties are Young’s modulus $E = 210 \text{ GPa}$ and Poisson’s ratio $\nu = 0.3$. Due to a double symmetry, only a quarter of the plate is analysed. The thickness of the plate is $0.6 \text{ m}$, and the inner and outer ellipses are given by:

\begin{align}
\left(\frac{x}{a}\right)^2 + \left(\frac{y}{b}\right)^2 &= 1 \quad \text{inner ellipse} \\
\left(\frac{x}{a}\right)^2 + \left(\frac{y}{b}\right)^2 &= 1 \quad \text{outer ellipse}
\end{align}

(26) (27)

Results are calculated on five systematically refined hexahedral meshes (96, 576, 4 608, 36 864 and 294 912 cells); this imitates closely the analysis of Demirdžić et al. [3], allowing fair comparison with their benchmark results. Subsequently, calculations are performed on tetrahedral and polyhedral meshes to examine the effect of employing fully unstructured grids.

The predicted stress distributions of the coupled method on the plane $z = 0.3 \text{ m}$, $\sigma_{xx}$, $\sigma_{yy}$, $\sigma_{xz}$ and $\sigma_{yz}$ components, are shown in Figures 9(a) and 9(b) side-by-side with those from Demirdžić et al. [3]; qualitatively, a close agreement can be seen with slight mismatches in the alignment of stress contours at the symmetry plane.

The distribution of the local stress component $\sigma_{ss}$ along the local co-ordinate $s$ is presented for the 5 hexahedral meshes in Figure 10(a), where the local co-ordinate $s$ runs along the upper inner elliptic edge of the plate (see Figure 8). As the mesh is refined, the results can be seen to approach a grid independent solution and the reference solution from Demirdžić et al. [3]. The reference solutions have been generated by fitting cubic splines through the key point values given in Demirdžić et al. [3]. Examining the $\sigma_{ss}$ stress at $s = 0.0$, as suggested in the NAFEMS benchmark, the percentage discretisation error is shown in Figure 10(b) to reduce at a second order rate with decreasing mesh spacing. The reference solution has been approximated using Richardson extrapolation [48] and the solution from the two finest meshes. As there are no FV computational nodes at the corner point, the stress values are extrapolated from adjacent boundary face-centres and cell-centres. The target

*Computers and Structures (0000)*
stress value given in the NAFEMS benchmark cannot be directly used for comparison as the fixed
displacement boundary conditions used in the current study mimic those used in Demirdžić et al.
[3] and do not correspond exactly to those in the NAFEMS benchmark.
The distribution of the von Mises equivalent stress $\sigma_{eq} = \sqrt{\frac{1}{2} |\sigma|^2}$, where $\sigma'$ is the deviatoric
stress, and displacement components $u$ and $v$, along $r = \sqrt{x^2 + y^2} = 2.1$ m, $z = 0.3$ m, are
shown in Figures 11(a) and 11(b). In each case, the predicted results converge towards the
reference solutions. Figures 12(a), 12(b), and 12(c), show the distribution of equivalent stress and
displacement components $u$ and $v$ along the line $x = y$, $z = 0.3$ m; the predicted results approach
the reference solution in a similar manner to the previous plots.
Table II shows the wall-clock times and memory requirements for each of the runs; for
comparison, results are given for the segregated FV methods (Segregated 1 and Segregated 2),
and commercial FE software Abaqus. Unlike in the previous 2-D test case where a direct LU
decomposition solver was employed, here a BiCGStab linear solver with ILU(0) preconditioner has been used for the coupled approach. The BiCGStab solver was found to be more time and memory efficient for 3-D cases than the Eigen library LU decomposition direct solver. A GMRes iterative solver with ILU(0) preconditioner was also trialled, but found to be approximately twice as slow as BiCGStab in all 3-D cases examined. A solution tolerance of $1 \times 10^{-6}$ has been used in the FV models. From Table II, it can be seen that the new coupled method is faster than the segregated FV methods by a factor of 2.5 to 6 times; as expected, the memory requirements are greater, by approximately 4.5 times in the largest mesh case. Interestingly, when compared to the FE solution, the new coupled method is faster and considerably more memory efficient in all cases; in the finest mesh case the coupled method is almost 6 times faster and uses 8 times less memory. The larger memory requirements of the FE solution may be attributed to the employed direct solver; however, the larger FE wall-clock times are somewhat more surprising, and highlight the efficiency of the new method.

To examine the applicability of the method on truly unstructured meshes, the analysis has been performed on a general polyhedral grid and a tetrahedral grid, each with approximately the same number of cells as the finest hexahedral grid (mesh 5). The polyhedral mesh contains 275,533 cells and has been generated using cfMesh [49], a new open-source cartesian-based meshing software. The tetrahedral mesh contains 295,536 cells and has also been generated using cfMesh; images

Figure 10. Elliptic plate: $\sigma_{ss}$

Figure 11. Elliptic plate: distribution of equivalent stress $\sigma_{eq}$ and displacement components $u$ and $v$, along $r = \sqrt{x^2 + y^2} = 2.1$ m; $z = 0.3$ m
Figure 12. Elliptic plate: distribution of equivalent stress and displacement components $u$ and $v$ along $x = y$, $z = 0.3$ m

<table>
<thead>
<tr>
<th>Mesh</th>
<th>Coupled Time</th>
<th>Coupled Memory</th>
<th>Segregated$^1$ Time</th>
<th>Segregated$^1$ Memory</th>
<th>Segregated$^2$ Time</th>
<th>Segregated$^2$ Memory</th>
<th>Abaqus Time</th>
<th>Abaqus Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>72</td>
<td>0.03</td>
<td>7</td>
<td>0.5</td>
<td>6</td>
<td>0.5</td>
<td>6</td>
<td>4</td>
<td>24</td>
</tr>
<tr>
<td>576</td>
<td>0.15</td>
<td>13</td>
<td>1.0</td>
<td>8</td>
<td>2</td>
<td>9</td>
<td>5</td>
<td>31</td>
</tr>
<tr>
<td>4 608</td>
<td>1.6</td>
<td>51</td>
<td>6.5</td>
<td>20</td>
<td>14</td>
<td>30</td>
<td>6</td>
<td>107</td>
</tr>
<tr>
<td>36 864</td>
<td>11</td>
<td>300</td>
<td>102</td>
<td>80</td>
<td>145</td>
<td>120</td>
<td>34</td>
<td>1 197</td>
</tr>
<tr>
<td>294 912</td>
<td>242</td>
<td>2 200</td>
<td>1 474</td>
<td>500</td>
<td>1 990</td>
<td>800</td>
<td>1 375</td>
<td>17 900</td>
</tr>
</tbody>
</table>

Table II. Elliptic plate: wall-clock time (in s) and maximum memory usage (in MB)

of the meshes are shown in Appendix B. The stresses and displacements for the polyhedral and tetrahedral meshes are compared with results from the finest hexahedral solution and Demirdžić et al. [3] benchmark solution in Figure 13. Additionally, solutions are given for comparison using commercial FE software Abaqus where the finest hexahedral mesh has been used. In each case, both the polyhedral and tetrahedral results are seen to be very close to that of the hexahedral solution, the reference solution, and the FE solution. The stress contours in the $z = 0.3$ m $x$-$y$ plane are shown in Figure 14, comparing with solutions from the finest hexahedral mesh, the Demirdžić et al. [3] benchmark, and the Abaqus solution: all results are seen to agree closely.
Figure 13. Elliptic plate: comparison of predicted results from the new coupled method for hexahedral, polyhedral and tetrahedral meshes, the Demirdžić et al. [3] benchmark, and the Abaqus FE hexahedral mesh solution

Narrow T-Section Component Under Tension

The final benchmark case, shown in Figure 15(a), consists of a narrow engineering component with a T cross-section, and has been as proposed by Demirdžić et al. [3] as a benchmark. Due to symmetry, the solution domain consists of one quarter of the component. A constant negative pressure of 1 MPa is applied to the lower surface and the upper left surface is fully clamped, as illustrated graphically in Figure 15(a). The Young’s modulus is 210 GPa and Poisson’s ratio is 0.3. A hole of radius $R = 5 \text{ mm}$ is located at the expected stress concentration. Four separate systematically refined hexahedral meshes (624, 4 992, 39 936, and 319 488 cells) are examined, mimicking the meshes employed by Demirdžić et al. [3]; the coarsest mesh is shown in Figure 15(b).
Figure 14. Elliptic plate: stress component distributions on the plane \( z = 0.3 \) m (right) for the hexahedral, polyhedral and tetrahedral meshes compared with the Demirdžić et al. [3] benchmark and the Abaqus FE hexahedral mesh solution.

Figure 16 shows the \( \sigma_{xx} \) and \( \sigma_{yy} \) stress distributions on the plane \( z = 0.0 \) m for the finest mesh. When compared with the results of Demirdžić et al. [3], the stress contours are almost indistinguishable.

Follow the proposed benchmark of Demirdžić et al. [3], the equivalent stress and displacement components are plotted along three separate profiles on the \( z = 0 \) plane: \( r = 1.5R \), \( x = 0 \) and \( y = 0 \), where \( r \) and \( \theta \) are the polar coordinates centred on the hole with \( \theta = 0 \) at the bottom vertical section.
The predictions for the four hexahedral meshes are shown in Figures 17, 18 and 19. The results...
converge quickly to the the reference solution for each case, except for the displacement component in Figure 19(b); however, the scale here magnifies the discrepancy.

Figure 17. Narrow T-section component: equivalent stress plotted along $z = 0$ plane, $r = 1.5R$

Figure 18. Narrow T-section component: equivalent stress and $x$ displacement component plotted along $z = 0$ plane, $y = 0$

Figure 19. Narrow T-section component: equivalent stress and $y$ displacement component plotted along $z = 0$ plane, $x = 0$

The wall-clock times and memory requirements for each run are given in Table III; once again, for comparison the results are given for the segregated FV methods, and commercial FE software Abaqus. The same linear solvers and tolerances are employed as in the previous 3-D test case.
The results are very similar to those in the previous 3-D test case, where the new coupled method requires approximately a factor of 4 less wall-clock time and a factor of 4 more memory than the segregated FV methods. When compared with the Abaqus FE solution, the new coupled method is once again found to require less time and less memory; for the finest grid, 3 times less wall-clock time, and 5 times less memory, is required.

<table>
<thead>
<tr>
<th>Mesh</th>
<th>Coupled Time</th>
<th>Coupled Memory</th>
<th>Segregated¹ Time</th>
<th>Segregated¹ Memory</th>
<th>Segregated² Time</th>
<th>Segregated² Memory</th>
<th>Abaqus Time</th>
<th>Abaqus Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>624</td>
<td>0.2</td>
<td>15</td>
<td>0.7</td>
<td>8</td>
<td>1.5</td>
<td>10</td>
<td>2</td>
<td>98</td>
</tr>
<tr>
<td>4,992</td>
<td>2</td>
<td>58</td>
<td>6</td>
<td>24</td>
<td>10</td>
<td>30</td>
<td>3</td>
<td>102</td>
</tr>
<tr>
<td>39,936</td>
<td>29</td>
<td>340</td>
<td>98</td>
<td>88</td>
<td>416</td>
<td>140</td>
<td>33</td>
<td>1,500</td>
</tr>
<tr>
<td>319,488</td>
<td>421</td>
<td>2,400</td>
<td>2,220</td>
<td>560</td>
<td>2,647</td>
<td>900</td>
<td>1,236</td>
<td>13,000</td>
</tr>
</tbody>
</table>

Table III. Narrow T-section component: wall-clock time (in s) and maximum memory usage (in MB)

As in the previous test case, results from a polyhedral and tetrahedral mesh are compared with those of the finest hexahedral mesh, the Demirdžić et al. [3] benchmark solution, and the Abaqus FE solution, shown in Figure 20. The polyhedral and tetrahedral meshes have been generated using cfMesh, where the polyhedral mesh contains 337,085 cells and the tetrahedral contains 282,415 cells; images of meshes are shown in Appendix B. Figure 21 shows the stress and displacement predictions along the benchmark profiles for the polyhedral and tetrahedral meshes; for comparison, results are given for the finest hexahedral mesh, the Demirdžić et al. [3] benchmark, and FE software Abaqus where the finest hexahedral mesh was used. All predictions are seen to agree closely; the largest discrepancy can be seen in the displacement component v in Figure 20(e); however, this is magnified by the vertical axis scale. Small oscillations are present in the tetrahedral mesh solution, which are visible in the stress distribution images and the stress/displacement plots. Although the origins not clear, these oscillations may be related to the so-called checker-boarding issues that can occur with cell-centred FV methods; this will be examined further in future work, and a number of the previously proposed preventative methods will be considered.

5. DISCUSSION & CONCLUSIONS

A new fully implicit block-coupled FV methodology has been presented, including a face tangential derivative approximation based on least squares Finite Area method. The accuracy of the method has been shown through detailed comparison with analytical solutions and numerical benchmarks. For the cases examined, the new method has been found to be faster than the segregated FV methods in every instance, with approximate speed-ups of 5 times. In particular, the coupled method shows speed-ups of three orders of magnitude for the simple 2-D bending cantilever test case. As expected, the new method required approximately 4 times as much memory as the memory efficient segregated methods. When compared with a commercial FE software, the new method has been found to be faster in every case, where for the elliptic plate test case the new coupled method is almost 6 times faster. Examining memory requirements, it was clear that the iterative BiCGStab employed by the new method scaled more favourably than the FE solution direct solver, with a factor of 8 difference.
Figure 20. Narrow-T section component: comparison of predicted results from the new coupled method for hexahedral, polyhedral and tetrahedral meshes, and the Abaqus FE hexahedral mesh solution for the largest mesh in elliptic plate test case. It can be concluded that the new method provides an efficient and accurate alternative to segregated FV methods and standard FE methods for the analysis of complex 3-D problems in linear elasticity.
Figure 21. Narrow T-section component: stress component distributions on the plane \( z = 0.0 \) m, \( \sigma_{zz} \) (left) and \( \sigma_{yy} \) (right), comparing the hexahedral, polyhedral, and tetrahedral results with the Demirdzić et al. [3] benchmark and the Abaqus FE hexahedral results.

6. ACKNOWLEDGEMENTS

Financial support is gratefully acknowledged from the Irish Centre for Composites Research (IComp). The authors wish to acknowledge the DJEI/DES/SFI/HEA Irish Centre for High-End Computing (ICHEC) for the provision of computational facilities and support.

A. APPENDIX: IMPLICIT COMPUTATIONAL MOLECULE

The extended implicit computational molecule for triangular and general polygonal meshes are shown in Figure 22.
B. APPENDIX: POLYHEDRAL AND TETRAHEDRAL MESHES

The tetrahedral and general polyhedral meshes employed in the elliptic plate test case are shown in Figure 23. The tetrahedral and general polyhedral meshes employed in the narrow T-section component test case are shown in Figure 24.

C. REFERENCES


Computers and Structures (0000)
Figure 24. Narrow T-section component: front view of the tetrahedral and general polyhedral meshes


A BLOCK-COUPLED FINITE VOLUME METHODOLOGY FOR LINEAR ELASTICITY


