A Ball-Vertex Approach to r-Refinement for Accuracy Enhancements in CFD Calculations.

Bevan W.S. Jones, Arnaud G. Malan*, Jacobus A. van Rooyen

Department of Mechanical Engineering, University of Cape Town, Rondebosch, South Africa

*South African Research Chair (SARChI) in Industrial Computational Fluid Dynamics

Abstract
Presented in this paper is an accuracy enhancing r-refinement scheme based on the ball-vertex method [1, 2]. Due to the success of the method in mesh movement, this work seeks to extend its use to error minimization for computational fluid dynamics calculations. To this end, element edges are loaded via error driven monitor functions. The latter are estimated from both field gradients and curvature. Boundary nodal movement is facilitated via the use of automated Bézier surface reconstruction. The application study involves both analytical as well as industrial test-cases. The latter involves 2D and 3D transonic flow calculations. When compared with the mesh independence solution, an error reduction in the computed coefficient of lift and moment of 60% was achieved even on relatively coarse meshes and close to an order of magnitude on finer meshes. Finally, a mesh deformation, moving boundary, problem was completed to demonstrate duality as both a mesh optimisation and deformation tool.

Keywords: r-refinement, Mesh Movement, Truss Networks, Ball-Vertex Method, moving boundary problems

1. Introduction
The use of computational fluid dynamics (CFD) to solve multi-physics problems has seen rapid growth over the last decade. As computational power has increased, so the number of elements has increased resulting in improvements in accuracy. Despite this, computational cost remains a major challenge and is expressed through the limitation of available hardware for simulations. Foremost is the accuracy of the simulation, and here the higher the desired fidelity, the higher the hardware requirements which, in turn, increase cost.

Clearly the improvement of computational efficiency, i.e. accuracy vs. computational cost, should remain an active area of research. Mesh adaptation is a prime example which involves placing vertexes so as to reduce errors arising from the underlying discretisation process. Areas of the flow with high gradients and curvatures or where mesh spacing is inappropriately large are typically associated with high errors. This significantly worsens efficiency of a field solver. Therefore, by improving the error topology over a mesh, a mesh adaptation process can be employed to achieve greater computational efficiency.
The effect then is to increase computational efficiency via reducing error through relo-
cating nodes to areas of high field gradient and/or curvature (which implies discretisation
error). However, the degree to which local element spacing can be manipulated via manual
re-meshing is limited to what is known about the flow field a priori. Additionally, even if
the flow field is well understood, the concentration of nodes into areas of high gradient is
time intensive if done manually. This is ascribed to flow phenomena, such as shocks and
vortices, often taking convoluted and complex patterns.

Mesh adaptation may also be effected in a more automated fashion during the solution
process which offers a posteriori knowledge of the field. Here, robustness is paramount,
while applicability to complex geometries (in 2D and 3D) is important to ensure industrial
relevance. To date, a number of approaches, namely p-, h-, and r-refinement, have been
developed. Here r- involves the relocation of mesh vertices. In contrast, p- increases the
order accuracy of the discretised equations, while h-refinement seeks to reduce error by the
addition and subtraction of mesh elements.

h-schemes have enjoyed wide spread commercial use in part due to the ability to take
a coarse initial mesh and obtain a solution of almost any desired accuracy (given adequate
computational resources). The method does however increase mesh complexity as well as
require a domain decomposition operation, post refinement, for parallel problems. However,
by coupling h-schemes with p- or r- types [3, 4], certain synergies can be realised. As an
example, r-refinement can be used to optimise a certain mesh without impact on connectivity.
If solution accuracy is still insufficient, a limited amount of h-adaptivity can be introduced.
This can be done without disturbing parallel computing load balancing to a large degree.

The underlying principle of r-refinement is to achieve an equi-distribution of the error
over the mesh, ideally resulting in the optimal mesh for a given number of nodes. One of
the first to exploit this concept was de Boor [5]. r-Refinement is still in its infancy when
compared with h- and p- types, which are now of industrial strength [6]. Its behaviour is
less well documented and further work is required. Reviews have been conducted by Tang
[7], Budd, Huang and Russell [6], and Baines, Hubbard and Jimach [8], which refer to the
r-method as the ’Moving Mesh Method’. Much of the difficulty with r-refinement is the
robustness of the schemes developed to date. Mesh entanglement or unacceptable element
quality is still of concern.

A common thread through existing r-refinement literature, (see [7, 6, 8]) is the duality
in ability to solve for moving boundary problems (mesh deformation) as well as mesh error
minimisation. The mechanics of both methods utilise some monitor function to drive the
adaptive process, whether it be for boundary related changes or the desire to change the
error topology over the mesh. As noted previously, industrial relevance requires avoidance
of mesh entanglement. Here, a truss network approach, wherein spring stiffness is inversely
proportional to edge length, holds potential.

This leads to the consideration of the truss network or spring analogy as a mesh movement
method for CFD. The Ball-Vertex method was first proposed by Bottasso [1, 2] and proved
effective for mesh movement. This method achieves improved performance via the addition
of a spring between each vertex of the element and the opposite element face (the face
through which the vertex would pass if it were to become concave). The added spring
is placed orthogonal to this face. Compared to the torsional spring method [9, 10], the ball-vertex variant is reported to offer improved robustness under severe deformations [1].

Extending such to error reduction, therefore, appears a valid proposition as alluded to by Acikgoz [11]. Though the focus of the later work was more on h-adaptivity and mesh deformation for FSI problems, the closing chapter briefly explores application to error field reduction. Consequently, this paper considered the use of the Ball-Vertex mesh movement algorithm for the purpose of error reduction of industrial CFD calculations. This is of value as it would improve accuracy without altering mesh connectivity. This required enhancements to account for truss rotations as well as the formulation of a cost effective monitor function for challenging CFD applications. In order to demonstrate industrial relevance, the developed technology is applied to transonic flow on aerofoils in both 2D and 3D.

2. Governing Equations

2.1. Classic Truss Networks

The mechanics of linear truss networks has long been understood and involves finding a solution to vertex positions for connected members of a structural system. For a generalised truss network, the system can be expressed in matrix form as

\[ K \delta x = f \]  

where \( K \) represents the system stiffness matrix, \( \delta x \) is the vertex displacement vector, and \( f \) the load vector. The displacement of a node can be solved by an Algebraic Multi-Grid Solver (AMG) or some other sparse solver. In this work, it is typical to set the stiffness of a truss element (edge or element) to be inversely proportional to length. This offers the desired stiffening effect.

In the derivation of the above system, an assumption is typically made to negate large strains and rotations. However, for the purposes of this work this is problematic due to large nodal movement, resulting in significant changes in stiffness and truss member orientation. A novel approach was adopted, mixing the original system matrix with the current, thus helping to alleviate errors made over multiple iterations. The resulting system is then of the form

\[ \frac{1}{2} (K^{n+1} + K^0) \delta x^{n+1} = f^{n+1} \]  

where

\[ \delta x^{n+1} = x^{n+1} - x^0 \]

with \( K^0 \) representing the system stiffness matrix sampled on the first iteration and/or first refinement pass. \( x^0 \) represents the original nodal co-ordinates which are sampled similarly. The above system is then solved iteratively until \( \delta x^{n+1} \) is converged.
2.2. Ball-Vertex Augmentations

The ball-vertex method [1, 2] involves adding an additional spring to every node in every attached element. With reference to Figure 1, the additional spring is placed between each node and an imaginary diagonal face. The face has an orientation such that should the node pass through it, the containing element would become concave. This represents the shortest distance, referred later to as $L_{ip}$, to element invalidity due to node-edge or node-face crossings (mesh entanglement).

The position of the resulting pseudo node $p$ is given by

$$x_p = x_i + ((x_k - x_i) \cdot t_k) t_k$$  \hspace{1cm} (3)

where $k$ denotes any of the nodes, $j_1-3$, attached to $i$ as depicted, and $t_k$ is the unit vector pointing from node $i$ to the selected $j$ node. The displacement of $p$ is interpolated using the connected nodes by

$$\delta x_p = \xi \delta x_{j1} + \eta \delta x_{j2} + (1 - \xi - \eta) \delta x_{j3}$$  \hspace{1cm} (4)

where

$$\xi = \frac{(x_{j2} - x_{j1}) \cdot (x_p - x_{j1})}{|x_{j2} - x_{j1}|}$$

$$\eta = \frac{(x_{j3} - x_{j1}) \cdot (x_p - x_{j1})}{|x_{j3} - x_{j1}|}$$

From here a sub-matrix for the new spring stiffness can be built and added to the global $K$ matrix. The sub-matrix is of the form

$$K_{sub} = k_{ip} \begin{pmatrix} t_{ip}t_{ip} & -\xi t_{ip}t_{ip} & -\eta t_{ip}t_{ip} & -(1 - \xi - \eta) t_{ip}t_{ip} \\ \xi^2 t_{ip}t_{ip} & \xi^2 t_{ip}t_{ip} & \xi(1 - \xi - \eta) t_{ip}t_{ip} \\ \eta^2 t_{ip}t_{ip} & \eta^2 t_{ip}t_{ip} & \eta(1 - \xi - \eta) t_{ip}t_{ip} \\ symm. & symm. & symm. \end{pmatrix}$$  \hspace{1cm} (5)
The individual stiffness of the spring \( i_p \) is given as \( k_{i_p} \) and is discussed shortly. Importantly, this formulation requires no additional system complexity as pseudo nodes do not result in additional degrees of freedom.

2.3. Stiffness and Load Formulation

The stiffness of the member of a real truss network is typically dictated by geometry. In this fictitious setting these values need to be created. A desirable property is that edges and elements should not collapse. Thus, a dynamic stiffness is highly desirable, where an increase in resistance is sought as edges and/or elements approach collapse, and decrease in the reverse. Making use of the inverse of the edge length, \( L_e \), does this. Thus the formulation for the spring stiffness is

\[
k_e = \frac{1}{L_e^n}
\]  

and is used for both edge and element springs. While the formulation has been previously noted \([12, 13]\), \( n \) is typically set to 1 \([1, 2, 14, 9, 10, 15, 11]\). However, in this work it was found that a value of 2 produced better results. Thus, stiffnesses for the presented scheme are formulated as

\[
k_e = \frac{1}{L_e^2}
\]  

This change creates a greater difference in relative stiffness which increases the non-linearity of \( K \). The greater stiffness assists in alleviating element collapse around strong field gradients (such as sonic shocks).

Finally the monitor (force) function can be derived. The error across the edge is turned into a fictitious force, acting to compress the edge and thus reduce error. The individual edge forces are then summed around a node to construct the load vector \( f \). While it is expected that gradients are indicative of discontinuities (sonic shocks), Roy \([16]\) noted that curvature tends to have a better effect on smooth fields. As industrial CFD calculations may contain both, a hybrid of the two was selected. The proposed equation for edge force is then given by

\[
f_{ij} = C_c f_{ij,\text{curve}} + C_g f_{ij,\text{grad}}
\]  

with \( f_{ij} \) the total force over the edge \( ij \) and \( f_{ij,\text{curve}} \) and \( f_{ij,\text{grad}} \) the curvature and gradient edge forces respectively. \( C_c \) and \( C_g \) are the curvature and gradient weighting factors, the values for which were determined experimentally in this work (Section 4).

The gradient component of the monitor function is computed simply as

\[
5
\]
\[ f_{ij,\text{grad}} = \left| \frac{\phi_i - \phi_j}{x_i - x_j} \right| t_{ij} \]  

and the curvature function, evaluated at the edge center, by

\[ f_{ij,\text{curve}} = \left| \frac{\phi''}{(1 + \phi'^2)^{\frac{3}{2}}} \right| t_{ij} \]  

where \( \phi \) represents the field being used to estimate error with single and double accent denoting first and second directional derivatives of \( \phi \). For the compressible flow examples, density was used for this purpose. \( L_{ij} \) is the length of the edge \( ij \). To obtain the curvature over the edge, a function \( \phi(l) \) is formulated as a clamped cubic spline:

\[ \phi(l) = a_0 l^3 + a_1 l^2 + a_2 l + a_3 \]  

where \( l \in [0 : L_{ij}] \) and the coefficients \( a_{0-3} \) can be shown to be

\[
\begin{align*}
a_0 &= \frac{2}{L_{ij}^3} (\phi|_0 - \phi|_{L_{ij}}) + \frac{1}{L_{ij}^2} \nabla_{ij} [\phi|_0 + \phi|_{L_{ij}}] \\
a_1 &= \frac{3}{L_{ij}^2} (\phi|_{L_{ij}} - \phi|_0) - \frac{1}{L_{ij}} \nabla_{ij} [2\phi|_0 + \phi|_{L_{ij}}] \\
a_2 &= \nabla_{ij} \phi|_0 \\
a_3 &= \phi|_0
\end{align*}
\]

and \( \nabla_{ij} \) implies the directional derivative in the direction of edge \( ij \).

2.4. Adjustment Factors

An adjustment factor is employed in order to limit nodal displacement and prevent mesh entanglement. The construction of the adjustment factor involves matching the solution for a single node to its shortest distance to collapse with the aid of a safety factor, thus given by

\[ C_{\text{adj}} = \min(C_{i,\text{adj}}) = \min \left( \frac{\min(L_{ip})}{\eta_{sf} \delta x_i} \right) \]  

where the \( \min(C_{i,\text{adj}}) \) implies over all nodes and \( \min(L_{ip}) \) is the shortest distance to element collapse for the elements attached to the node \( i \). Further, \( \eta_{sf} \) is typically set to 1.5 in this work. The new coordinate, \( x'_i \), computed for the entire field is then limited according to

\[ x'_i = x_i + C_{\text{adj}} \delta x_i \]  

with nomenclature previously defined.
2.5. Boundary Refinement

In treating boundary nodes, both an \textit{a priori} and an \textit{a posteriori} approach were taken. The former involves stiffening boundary nodes in the boundary surface normal direction with the addition of normally oriented springs (Figure 2). This does not constrain tangential movement. These new springs are added to the system stiffness matrix by

\[
K' = K + k_{i,n} \begin{pmatrix} N_{i,x} \cdot N_{i,x} & N_{i,x} \cdot N_{i,y} \\ N_{i,y} \cdot N_{i,x} & N_{i,y} \cdot N_{i,y} \end{pmatrix}
\]

with \( N_i \) representing the boundary normal at the node and \( k_{i,n} \) the added spring stiffness. This is typically set to two orders of magnitude higher than any attached spring stiffness. To aid in the above, the boundary normal nodal force component is also removed as

\[
f'_i = f_i - f_i \cdot N_i
\]

with nomenclature previously defined.

Subsequent to the above constraints, nodes may still move off the boundary surface as is the case for curved surfaces. The \textit{a posteriori} component of boundary refinement seeks to correct this by placing the node back on to the boundary surface. For curved or smooth boundary surfaces, 3\textsuperscript{rd} order Bézier curve (2D) or surface (3D) fits are used. The boundary normals required for these are computed from mesh node locations and therefore no explicit additional boundary surface definitions are required.

The method of construction of the Bézier curves and surfaces follows that of Walton and Meek [17]. Nodes that have moved off the surface are placed back to the closest point on the surface. This is done via Newton linearisation and sub-iterations. Additionally, it is noted that corner nodes are not moved and that in 3D, ridges are constrained to move only along the ridge.

Figure 2: Boundary diagram with boundary springs added
3. Refinement Scheme Operation

With the mechanics of the r-refinement scheme presented, discussion turns to integration with the CFD code Elemental™. Once the flow solver has been initialised, the flow solution is converged in a typical manner until it meets a user-specified threshold (typically 3 orders of magnitude drop in residual). At this point the r-refinement scheme is called and a so-called ‘r-refinement pass’ is conducted. At present, the refinement scheme is called a set number of times (referred to as refinement passes). Once the last refinement pass has been made, the flow solution is converged fully.

The refinement pass may be sub-divided into three main sub-functions, shown in Figure 3. The construction of the system stiffness matrix (Sections 2.1 and 2.2) and load vector (Section 2.3) with the a priori modifications (Section 2.5) is performed in step 1. A solution is obtained and the displacement is constrained by \( C_{adj} \) (Section 2.4) in step 2. The nodal co-ordinates are then updated in step 3 and edge lengths and nodal volumes re-computed in step 4. Two iterations of limited Lapacian smoothing are applied in step 5 and a second mesh updating procedure is then carried out in step 6 in which geometric properties (edge length, nodal volumes, etc) are updated as well as the interpolation of solver field variables to the new mesh.
If further r-refinement iterations are required, a limited mesh interpolation is carried out in step 8, where only field variables relevant to the monitor function (density in this work) are interpolated forward. Inverse distance interpolation was employed and always using the original field variables (as received from the flow solver); this to prevent deterioration in monitor function accuracy due to repeated interpolations. The inverse interpolation equation for node $i$ reads
\[
\phi_i = \frac{\sum_j \phi_j^0 |x_i - x_j^0|^2}{\sum_j |x_i - x_j^0|^2} \sum_j |x_i - x_j^0|^2 \neq 0
\]  

(17)

where subscript \( j \) is the index of the nodes connected to \( i \) and superscript 0 is the value as received from the CFD solver.

4. Numerical Test Cases

4.1. Analytical Example

The aim of the analytical tests were to demonstrate the qualitative ability of the scheme to track field phenomena and cluster nodes accordingly. For this purpose analytical fields are prescribed for \( \rho \) on a unit domain with a structured mesh configuration of 64x64 nodes in 2D. The r-refinement scheme was then applied using 5 and 10 passes and the resulting meshes observed for field gradient correlation. In all cases computational times required to perform the passes were circa. 5 seconds on an Intel i7 2.30GHz chip with 8Gb of DDR3 RAM at 800MHz.

The first case is that of a sinusoidal type function aligned with the line \( y = x \). The field equation is given as

\[
\rho(x, y) = 1.5 - 0.5 \cos 4\pi \sqrt{\left(x - \frac{x + y}{2}\right)^2 + \left(y - \frac{x + y}{2}\right)^2}
\]  

(18)

The field is plotted in Figure 4, along with the fabricated error function (based on gradient only).
The adapted meshes are shown in Figure 5. As can be seen, node distribution corresponds well with the error topology plot. In addition, as more passes were applied the agreement improved. Finally, the desired boundary motion was achieved, i.e. with boundary nodes following the trend of non-boundary counterparts. Also apparent was deterioration in element quality (loss of orthogonality of edges). This was expected as the refinement scheme does not explicitly regulate element quality.
For the final analytical case, a twisted flow field inspired by Delzanno et al. [18] is considered. This demonstrates the schemes ability to adapt to very sharp flow features, similar to those found at aerodynamic shocks. Again the monitor function features no curvature components and is purely based on gradient. The equation for the density field is given by

$$\rho(x, y) = \gamma \left(1 + \frac{9}{1 + [10r \cos(\theta - 20r^2)]}\right)$$

(19)

where

$$r(x, y) = \sqrt{(x - 0.5)^2 + (y - 0.5)^2}$$

$$\theta(x, y) = \tan^{-1}\left(\frac{y - 0.5}{x - 0.5}\right)$$

The plots for both the density and error fields are given below. This is followed by the resulting refined meshes after 5 and 10 passes.

Figure 6: Plot of the density field (left) and the gradient based error field (right).
Similar results were obtained to the previous set. Subsequent to r-refinement, the node distribution and error field again corresponded well. Additionally, as the error increased in the outer areas of the domain, so the node density increased. However, element quality was again seen to suffer.

4.2. NACA0012 Test Case

The industrial 2D example was that of the well known NACA0012 aerofoil. The flow was inviscid transonic, with the Mach number and angle of attack being 0.85 and 1.25° respectively. The 2nd order accurate ElementalTM solver was employed [19]. The success of the r-refinement algorithm was quantified via improvement in lift and moment. The target solution, i.e. "zero mesh spacing" solution, was obtained via the grid convergence index methodology [20] with Richardson’s extrapolation.

Various aspects of the r-refinement scheme can be investigated. The most pertinent of these is the effect of varying the ratios $C_c$ and $C_g$ (Section 2.3) on accuracy. Additionally, the effectiveness of the Lapacian smoother (to improve element quality) on both accuracy as well as flow solver reconvergence was assessed together with the effect of resetting the system stiffness matrix. As mentioned, the monitor function was density-based.

4.2.1. Mesh Independence Study

The mesh independence study involved the use of three Delaunay triangulation meshes with increasing node counts. A mesh similar to the coarsest is shown in Figure 8 (referred to as the "fine mesh" in the r-refinement study below), with the density computed on the finest mesh depicted in Figure 9. The lift ($c_l$) and moment ($c_m$) coefficients computed on the various meshes are listed in Table 1.
Figure 8: Close-up of the un-adapted fine NACA0012 mesh.

Figure 9: Plot of the density field on a 72k node mesh for a NACA0012 aerofoil.
Table 1: Results from the various meshes for $c_l$ and $c_m$ in 2D.

<table>
<thead>
<tr>
<th>No.</th>
<th>Nodes</th>
<th>$c_l$</th>
<th>$c_m$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5 380</td>
<td>0.33734</td>
<td>-0.03748</td>
</tr>
<tr>
<td>2</td>
<td>30 115</td>
<td>0.33935</td>
<td>-0.03757</td>
</tr>
<tr>
<td>3</td>
<td>72 858</td>
<td>0.34234</td>
<td>-0.03784</td>
</tr>
</tbody>
</table>

With the above, the mesh independent solution was calculated via Richardson extrapolation. The certainty, or relative certainty, with which the extrapolated variable is judged via the grid convergence index or $GCI$. The results obtained for the values $c_l$ and $c_m$ if using meshes 2 and 3 are given in Table 2.

Table 2: Results from the mesh independence study for $c_l$ and $c_m$.

<table>
<thead>
<tr>
<th>$\zeta_{extrap}$</th>
<th>$c_l$</th>
<th>$c_m$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$GCI$</td>
<td>0.95%</td>
<td>0.34%</td>
</tr>
</tbody>
</table>

Having obtained the target solution, the developed r-refinement technology was applied to two meshes of different resolution. These are referred to as fine and coarse meshes respectively.

4.2.2. Fine Mesh Adaptation

The mesh for the first refinement case was similar to the coarsest mesh used in the mesh independence study. Here, the selection of 'fine' in the title is used to distinguish this mesh from the next case, where the mesh is significantly coarser. For these tests, a specified number of refinement passes were applied as stated before. Each refinement pass occurred when the flow solution had re-converged to $1.0e-3$. Once all refinement passes were complete, the flow solution was converged to $1.0e-5$ to conclude the simulation.

The set-up for the refinement was therefore as follows:

1. Four batches consisting of two 5 and two 10 pass refinements are done.
2. One of the two sub-batches has the Laplacian smoother activated and the other deactivated. This is in the interest of element quality improvement.
3. In each sub batch, 13 individual tests are done with varying $\frac{C_g}{C_c}$ ratios, $\frac{C_g}{C_c} = [0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.25, 1.5]$
4. Initial conditions for the un-adapted mesh are:
   - (a) $c_l = 0.3429$; error of 1.82%
   - (b) $c_m = -0.0385$; error of 1.64%
(c) Computational cost of the unrefined mesh is 545.81 seconds if using a simple Jacobi solver [19].

Computational times for the various cases are reported in Table 3. The simulations were carried out on the University of Cape Town HPC’s Dell C6145 racks. These house AMD Opteron 6274s at 2.2GHz and 2Gb of RAM at 1.3GHz. The typical pass time is presented for each batch and multiplied by the number of passes for a given batch to obtain the total refinement time. Total run time for each batch is given as an average in addition to the standard deviation (with the non-converging cases excluded). The typical refinement time is shown as a percentage of total run time and in the final column of the table, the increase in computational cost relative to the unrefined case is shown (due to repeated solves).

Table 3: Typical computational time for 2D fine mesh refinement (reported in seconds). The last column refers to the factor increase in CPU cost.

<table>
<thead>
<tr>
<th>Batch Name</th>
<th>Pass Time</th>
<th>Tot. Refi. Time</th>
<th>Tot. Run Time</th>
<th>Std. Dev. Run Time</th>
<th>as % of Tot. Run</th>
<th>Inc. in Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 LA</td>
<td>20.03</td>
<td>100.17</td>
<td>1597.66</td>
<td>187.76</td>
<td>6.27</td>
<td>2.93</td>
</tr>
<tr>
<td>5 LD</td>
<td>16.86</td>
<td>84.30</td>
<td>1586.91</td>
<td>228.82</td>
<td>5.31</td>
<td>2.91</td>
</tr>
<tr>
<td>10 LA</td>
<td>18.39</td>
<td>183.91</td>
<td>2658.66</td>
<td>196.89</td>
<td>6.92</td>
<td>4.87</td>
</tr>
<tr>
<td>10 LD</td>
<td>16.13</td>
<td>161.33</td>
<td>2395.13</td>
<td>372.60</td>
<td>6.74</td>
<td>4.39</td>
</tr>
</tbody>
</table>

For the above, the actual refinement computational cost was low where < 10% of full run times were needed to perform the refinement. The increase in total CPU cost, when compared with the unrefined mesh, was well less than the number of passes applied. The increase in overall CPU time was significant (3 to 5 times), but not deemed prohibitive bearing in mind the considerable increase in accuracy achieved (Figure 10). It should be noted that this study served as a first assessment of the refinement technology, and little effort was applied to scheme efficiency optimisation. However, it was found that the refinement threshold (flow solver convergence tolerance at which a sweep is done) had a significant effect on overall scheme efficiency and should be investigated further.
Figure 10: Graph of the error reduction in the coefficient of lift (top) and moment (bottom) vs. the $C_g/C_c$ ratio on the fine NACA0012 mesh. Here, LA and LD respectively denote Laplacian smooth activated and deactivated.

Missing data points indicate tests for which the flow solver failed to re-converge (likely due to deterioration in element quality). As can be seen, this was more common for those batches with increased refinement passes as well as those with a higher gradient weighting. However, no tests failed to have the refined mesh reprocessed, inferring that no invalid elements were created (ElementalTM does checks for this).

The results obtained were, for the most part, satisfactory and a clear demonstration of the proposed scheme’s ability to significantly improve accuracy. Figure 11 shows the general trend of node movement, where nodes moved toward the aerofoil and further clustered around the nose, tail and, importantly, along the shock. For this case, the optimal $C_g/C_c$
ratio band lies from 0.3 to 0.6 if an improvement in both $c_l$ and $c_m$ is sought, where a reduction in error of circa 50\% was achieved. While increasing the weight of the gradient did seem to improve the lift coefficient, the chance of solver rejection was raised, and more importantly, a clearly negative effect on the moment coefficient was observed. This again could be due to element quality deterioration around the leading and trailing edges and is to be investigated in future work. Finally, the effect of increasing the number of passes to 10 did little for accuracy and had a clear negative effect on solver acceptance. This is again suggestive that r-refinement without element quality control has reached a limit.

Figure 11: Adapted mesh for test 10LD for a $C_g/C_c$ ratio of 0.5.

Finally, the effects of the Laplacian smoother with regard to accuracy improvements are only noticeable for high curvature weighted monitor functions and high refinement passes. Figure 12 shows the difference in refinement around the shock. Clustering nodes in the area of shock is important; however, the curvature monitor function tended to cluster nodes on either side. The Lapacian, when applied, relaxed this and so can be attributed with the improved accuracy in this area; however, little effect was observed for $C_g/C_c = [0.3 : 0.5]$. 

18
4.2.3. Course Mesh Adaptation

The second mesh (shown in Figure 13) was significantly coarser than the previous 'fine' mesh. This example is more instructive as the mesh results in a solution which is significantly removed from the mesh independent version. The set up for these tests was similar to that of the fine, with the exception of an absent smoother (as this was found to be of little value). Instead, here the effect of re-setting $\bar{K}^0$ at the beginning of each refinement pass was investigated. This was expected to give better approximations to the non-linear truss stiffnesses. The number of refinement passes was also reduced, this due to experience with the fine mesh case.

The set up for the refinement runs follows with Figure 13 showing the un-adapted mesh:

1. Four batches consisting of two 3 and two 5 pass refinements are done.
2. One of the two sub-batches has $\bar{K}^0$ reset on every refinement pass, while the other has $\bar{K}^0$ set on the first pass only.
3. In each sub-batch, 13 individual tests are done with $C_g/C_c = [0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.25, 1.5]$
4. Initial conditions for the un-adapted mesh are
   (a) $c_i = 0.3116$; error of 7.47%
   (b) $c_m = -0.0298$; error of 21.32%
   (c) Computational cost on the unrefined mesh is 71.95 seconds.
Times for the computations, which were run on the same Dell C6145 cluster as previously, are reported in Table 4.

Table 4: Typical computational time for 2D coarse mesh refinement, reported in seconds.

<table>
<thead>
<tr>
<th>Batch Name</th>
<th>Pass Time</th>
<th>Tot. Refi. Time</th>
<th>Tot. Run Time</th>
<th>Std. Dev. Tot. Run</th>
<th>as % of Run Time</th>
<th>Inc. in Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 OE</td>
<td>1.43</td>
<td>4.30</td>
<td>111.18</td>
<td>12.14</td>
<td>3.87</td>
<td>1.55</td>
</tr>
<tr>
<td>3 OF</td>
<td>1.44</td>
<td>4.32</td>
<td>115.57</td>
<td>10.58</td>
<td>3.74</td>
<td>1.61</td>
</tr>
<tr>
<td>5 OE</td>
<td>1.63</td>
<td>8.16</td>
<td>159.97</td>
<td>16.53</td>
<td>5.10</td>
<td>2.22</td>
</tr>
<tr>
<td>5 OF</td>
<td>1.63</td>
<td>8.17</td>
<td>169.10</td>
<td>18.21</td>
<td>4.83</td>
<td>2.35</td>
</tr>
</tbody>
</table>

Refinement cost as a percentage of total time was small, which is consistent with the results from the fine meshes. There was clearly little difference, cost-wise, in resetting the system stiffness matrix on every pass. Thus, one would clearly favour resetting on every pass due to accuracy improvement, as is discussed shortly.

The graphs showing the reduction in error for the $c_l$ and $c_m$ from the base mesh follows, Figure 14. Here, the abbreviations OE and OF mean ‘on every pass’ and ‘on first pass’, referring to when $\mathbf{K}^d$ was reset. Again, missing data points indicate tests which failed to re-converge and are not reflective of an invalid mesh (but rather deteriorating element quality). Importantly, the increase in overall computational cost was significantly less vs. reduction in error (both $c_l$ and $c_m$) than previously seen. Weighing this against the large increases in accuracy (shown below), significant value addition is deemed demonstrated.
Figure 14: Graph of the error reduction in the coefficient of lift (top) and moment (bottom) vs. the $C_g/C_c$ ratio on the coarse NACA0012 mesh. Here OE and OF respectively denote the resetting of $\bar{K}^0$ on the every and first pass.

The results obtained confirmed those obtained on the fine mesh tests. A $C_g/C_c$ ratio of 0.3 to 0.5 yielded the greatest improvement in accuracy for both $c_l$ and $c_m$ which in this case was circa 90%. As can be seen in Figure 15, the mesh nodes were pulled toward the aerofoil, and started to cluster around the shock. Better approximation of the non-linear system (by the resetting of $\bar{K}^0$) had a significant effect on accuracy, especially over the optimal $C_g/C_c$ ratios. As would be expected, the impact on lower passes was smaller; thus, the effect can be considered cumulative.

One discrepancy between fine and coarse results was the trend of the $c_l$ data. Unlike the fine mesh results, an increase in $C_g/C_c$ ratio did not increase solution accuracy. The exact
reason for this was unclear. Attributing factors could be initial mesh construction (Delaunay vs. paved) or initial mesh resolution. However, this discrepancy was not deemed dire since the aggregate of data is suggestive of consistency in accuracy improvements for $C_g/C_c$ ratios of circa 0.3 to 0.6. Finally, relatively little mesh motion was required (from studying the adaptive meshes) to yield significant improvements in accuracy. This indicates that still further gains are expected due scheme refinement e.g. adding element quality cosmetics.

![Image of adapted meshes](image)

Figure 15: Adapted meshes for test 5OE for a $C_g/C_c$ ratio of 0.5.

4.3. 3D r-refinement Test Case
4.3.1. Mesh Independence Study

For the purpose of the 3D test case, an industrially relevant aerofoil from the FP7 project Future Fast Aeroelastic Simulation Technologies (FFAST) [21] was employed. Again, the curvature and gradient ratios were investigated for improvements in accuracy. The Laplacian smoother was again abandoned since performance in 2D was not adequate to justify continued use. For these tests, the system stiffness matrix was reset on every refinement pass.

The flow was again inviscid, the angle of attack was set to $2.0^\circ$ and a Mach number of 0.8 was applied. Air properties were taken at cruise altitude. The calculation of the $c_m$ number differs somewhat from the two dimensional case. The standard formulation in 3D, taken from [22], is of the form

$$c_m = \frac{M}{P_{dy}} \frac{A_{pl}}{l_{ch}}$$  \hspace{1cm} (20)

where the dynamic pressure $P_{dy}$, planform area $A_{pl}$ and moments $M$ are easily computed. The characteristic chord length is $l_{ch}$. For the purposes of this work the aerofoil was assumed
to have a constant taper and sweep. This is not strictly true, but since validation is of little concern the discrepancy can be ignored. The characteristic length $l_{ch}$ of the aerofoil was calculated via Etkin and Reid [23]:

$$l_{ch} = \frac{2l_{ch,r}}{3} \frac{1 + \Lambda + \Lambda^2}{1 + \Lambda}$$

(21)

where

$$\Lambda = \frac{l_{ch,t}}{l_{ch,r}}$$

In the above, $l_{ch,r}$ and $l_{ch,t}$ denote the root and tip chord lengths. The value of the characteristics length was found to be 8.253m. The computed density field is presented in Figure 16 with the mesh independence results reported in Table 5.

Figure 16: Density field plot for the FFast aerofoil from the above (top) and below (bottom).
Table 5: Results from the various meshes for $c_l$ and $c_m$ on the FFAST aerofoil.

<table>
<thead>
<tr>
<th>No.</th>
<th>Nodes</th>
<th>$c_l$</th>
<th>$c_m$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>270 869</td>
<td>0.28856</td>
<td>0.20056</td>
</tr>
<tr>
<td>2</td>
<td>380 934</td>
<td>0.29081</td>
<td>0.20242</td>
</tr>
<tr>
<td>3</td>
<td>573 520</td>
<td>0.30116</td>
<td>0.21205</td>
</tr>
</tbody>
</table>

The mesh independent solution was again found as previously done. Table 6 shows the resulting extrapolated variables and GCIs.

Table 6: Results from the mesh independence study for $c_l$ and $c_m$ in on the FFAST aerofoil.

<table>
<thead>
<tr>
<th></th>
<th>$c_l$</th>
<th>$c_m$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_{extrap}$</td>
<td>0.2876</td>
<td>0.1999</td>
</tr>
<tr>
<td>GCI</td>
<td>0.41 %</td>
<td>0.43 %</td>
</tr>
</tbody>
</table>

The grid used for the refinement study had a node count of 155k and was constructed from tetrahedrons. The refinement threshold was again set to $1.0e - 3$. The set-up for the refinement was as follows:

1. Three batches consisting of 3, 5 and 7 pass refinements are done.
2. $K^0$ is reset on every refinement pass.
3. In each sub batch 13 individual tests are done with varying $\frac{C_G}{C_G^0}$ namely, $\frac{C_G}{C_G^0} = [0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.25, 1.5]$
4. Initial conditions for the un-adapted mesh are
   (a) $c_l = 0.27712$; error of 3.64%
   (b) $c_m = 0.1889$; error of 5.45%
   (c) Computational cost of the unrefined mesh is 6 hours 16 minutes.

Computational costs were found to again exhibit an overall increase with increasing number of sweeps. However, no batch reported to have increased total cost by more than double (whether measured by time or flow solver iteration). Typical refinement costs however constituted circa 30% of total run time. This was due to the increased complexity of the CFD solver’s volume calculations in 3D.
Figure 17: Graph of the error reduction in the coefficient of lift (top) and moment (bottom) vs. the $C_g/C_c$ ratio on the FFAST aerofoil.

The achieved reductions in error are depicted in Figure 17. When compared with the 2D results, a number consistencies emerge. Similar to the 2D cases, 5 passes showed the best overall improvements in accuracy. This was 40% in both lift and moment at a $C_g/C_l$ ratio of 0.5. Surprisingly though, accuracy peaked at a $C_g/C_l$ ratio of unity. This is to be further investigated as part of future work.

4.4. 3D Mesh Deformation Test Case

Mesh deformation testing was done primarily to demonstrate that the developed scheme could be used for large mesh deformation. For this purpose, a 200k node FFAST mesh was used, and subjected to an unrealistically large tip deflection of two thirds of the span.
(see Figure 18). This was done while resolving the flow field via Elemental™. For mesh movement purposes, the above displacement was broken into three equal steps. No invalid elements were created and the flow solver continued convergence. The volumetric deformed mesh and a close up of the tip is shown in Figures 19 and 20 respectively. The calculation was conducted on a Intel i7 2.30GHz chip with 8Gb of DDR3 RAM at 800MHz.

Figure 18: Un-deformed and deformed FFAST aerofoil

Figure 19: Deformed volumetric mesh
5. Conclusion

If continued improvements to CFD computational efficiency (accuracy vs. computational cost) are to be sought, mesh adaptation will need to be improved. An r-refinement scheme was investigated with this purpose. The ball-vertex truss networked variant was utilised as the mechanism through which to perform mesh adaptation. The formulation entails attaching fictitious springs to nodes, the stiffness of which are a function of the inverse edge lengths. Forces were applied across edges to be informative of discretization error. Both field gradients and curvature were utilised for this purpose. Nodes on boundaries were allowed to move, for which automated feature identification was employed. The above resulted in a novel and industrially relevant refinement scheme. The entire scheme was implemented into Elemental™’s compressible flow solver. Additional novelty resulted from improved approximation of non-linear displacements of the truss network.

For purposes of evaluation, the developed methodology was applied to analytical as well as industrial flow problems. The latter involved transonic flow in both 2D and 3D where reductions in error of 50% and 40% were respectively achieved on coarse and fine meshes. This was achieved for all cases using a gradient to curvature error estimation ratio of 0.5. Additional computational cost was mainly due additional CFD solver work required on account of re-solving after mesh adaptation. This was, however, of a factor of circa 3 at worst. Scheme robustness for the 0.5 ratio was demonstrated throughout. Finally, the developed technology was proven effective for large 3D mesh deformations, further demonstrating significant industrial potential for CFD applications.

6. Acknowledgements

The conducted research has been supported by the Department of Science and Technology and the National Research Foundation through the South African Research Chair in...
Further acknowledgement is given to the University of Cape Town’s ICTS High Performance Computing team who provided computational resources for numerous test cases. http://hpc.uct.ac.za

References


