A GENERALISED, CONVERGENCE ACCELERATED PRESSURE–BASED SEGREGATED ALGORITHM

V. Pržulj

Key words: CFD, Pressure–Based Method, SIMPLE Algorithm, Finite–Volume, Co–located, Unstructured Grid.

The paper reports on the development and validation of a convergence accelerated SIMPLE–based algorithm. This algorithm addresses the critical issue of pressure–velocity storage and coupling in the pressure–based finite–volume methods. A variant of cell–centred method, with co-located storage, is employed for both incompressible and compressible flows in conjunction with the solution of Reynolds Averaged Navier–Stokes (RANS) equations on arbitrary unstructured polyhedral cells. This method is a kernel of Ricardo’s new generation CFD solver, and some aspects of the method have been presented in [6, 7]

The SIMPLE (Semi–Implicit Method for the Pressure Linked Equations) algorithm and its variants have been a core of the pressure–based finite-volume methods. Since its conception in 1972 the algorithm has evolved to satisfy demands of unstructured grids and to cope with all speed flows. The SIMPLE–like algorithm effectively couples velocity and pressure fields by converting a discrete form of the continuity equation into an equation for the pressure correction. In an iterative procedure, the pressure corrections are used to update the pressure and velocity fields so that the velocity components obtained from the solution of momentum equations also satisfy the continuity equation. This is achieved by introducing the corrections for the velocity components, pressure and density: \( \vec{U}_P = \vec{U}_P^* + \vec{U}_P' \), \( p_P = p_P^* + p_P' \) and \( \rho_P = \rho_P^* + \rho_P' \), respectively. The discretized momentum equations for the cell \( P \) and its face \( j \), \( \vec{U}_P^* \) and \( \vec{U}_j^* \), then provide a link between the velocity and pressure corrections:

\[
\begin{align*}
\vec{U}_P' & = \vec{h}_P' - \frac{V_P}{a_P} \nabla p_P', \quad \vec{h}_P' = \frac{\sum_j a_j \vec{U}_P j + \vec{S}_U}{a_P} \\
\vec{U}_j' & = \vec{h}_j' - \left( \frac{V_P}{a_P} j \right) \frac{\vec{A}_j}{\vec{A}_j \cdot \vec{d}_j} \left( p_P' j - p_P \right) - \left( \frac{V_P}{a_P} j \right) \left[ \nabla p_j' - \frac{\vec{A}_j}{\vec{A}_j \cdot \vec{d}_j} \left( \nabla p_j' \cdot \vec{d}_j \right) \right]
\end{align*}
\]

In the above equations, the velocity corrections from neighbouring cells, represented by \( \vec{h}_P' \) and \( \vec{h}_j' \), are not known during the first correction step. Also, the underlined term which describes
the contribution of the cell-face pressure correction gradient on non-orthogonal grids is not known. These two important contributions to the pressure correction equation are neglected in the SIMPLE, see [4, 5, 1].

The present author and others, [6, 4], introduced additional correction steps to include the latter contribution, also known as the skewness correction. In this work, the pressure correction steps are devised in a novel way in order to take into account velocity corrections from neighbouring cells. Accounting for both neighbour and skewness corrections leads towards the fully implicit algorithm characterised by significantly improved velocity–pressure coupling, and consequently by accelerated convergence.

Figure 1 and Table 1 demonstrate the algorithm capability for two laminar flow benchmark cases, namely for the 2D lid–driven cavity flow from [3], and for the 3D flow in a rotating pipe, [2]. In both cases, the convergence rate (measured by a number of iterations required to obtain normalised residuals below $1 \times 10^{-7}$) can be significantly improved by performing two or more pressure correction steps. Also, an optimal number of pressure corrections exists for which the significant reduction of CPU time is possible. This number varies between two and four – four pressure corrections are typically used for complex industrial applications where poor quality numerical grids are frequently employed. Notably, higher under-relaxation factors

<table>
<thead>
<tr>
<th>Case</th>
<th>Correction steps</th>
<th>2D–Cavity $45^\circ$, $160 \times 160$ cells</th>
<th>Rotating tube, $158111$ polyhedral cells</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\alpha_u$  $\alpha_p$  Iterations  CPU time (s)</td>
<td>$\alpha_u$  $\alpha_p$  Iterations  CPU time (s)</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.9  0.05  1967  54.1</td>
<td>0.8  0.20  196  91.1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.95  0.03  856  34.6</td>
<td>0.9  0.30  99  87.0</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.96  0.02  581  28.3</td>
<td>0.9  0.30  91  124.8</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.98  0.01  793  28.3</td>
<td>0.9  0.30  80  109.7</td>
<td></td>
</tr>
</tbody>
</table>
for the momentum and pressure can be used. More precisely, the sum $\alpha_u + \alpha_p \approx (1.0 - 1.2)$ is a good indicator for their optimal choice.

The full paper will contain detailed description of the present pressure–based algorithm, accompanied with additional application examples.

REFERENCES


