OpenFOAM Workshop 2014:

Effects of grid quality on solution accuracy

Written by J. Rhoads
July 3, 2014

General Information

The 9th OpenFOAM Workshop was held in Zagreb, Croatia from June 23-26, 2014. Around 250 participants in the workshop took part in a series of presentations, tutorials, and special interest groups, with the intent on furthering the collective understanding of both OpenFOAM and CFD in general. Of particular interest was the significant discussion in regard to preprocessing–both in terms of grid generation and grid effects on the solution.

Grid Generation

Grid generation remains a significant challenge for CFD. There are several open source solutions that were discussed in varying levels of detail:

- blockMesh - Block structured grids for simple geometries
- snappyHexMesh - Automated hex-core poly mesher with some inflation layer capabilities
- cfMesh - Automated hex-core poly mesher that allows for retaining edges and can subdivide first layer of cells to create pseudo boundary layer mesh
- foamyHexMesh - Additional automated poly mesher

Interestingly, all of the open source tools for complex geometries use some sort of cut-cell approach (for automation purposes) in place of a bottom-up approach similar to that used in Pointwise. These methods are commonly favored because of their relative stability/robustness, their ability to handle arbitrarily complex geometries, and their scaling properties when run in parallel using distributed memory.

However, all open source solutions for complex geometries sacrifice control over the surface mesh and boundary layer cells in favor of automation. Each of the three automated methods have some form of boundary layer tools, but they all suffer from limitations including overall boundary layer thickness, lack of control over initial cell height, and limited robustness of inflation layers near complex features (such as concave corners). Further, all of these automated solutions use discrete representations of the underlying geometry, inherently limiting the surface resolution to that of the initial representation. That is, surface curvature information is lost, making grid refinement studies problematic.

Several attendees voiced the opinion that while open source tools are suitable (and indeed quite useful) for rough calculations, some precise, high fidelity simulations require more control

over the mesh than open source solutions can currently provide. In such applications, accurately capturing the physics of the boundary layers is paramount. Having sudden changes in cell orientation, cell thickness, or cell type can introduce significant numerical errors (as discussed following section), which will destroy the integrity of the boundary layer and can compromise the overall accuracy of the CFD solution. For such applications, a bottom-up approach with exact control over the elements near the viscous surfaces is sometimes preferred in order to reduce grid-induced effects in the simulation results.

Grid Effects on Solution Accuracy

Intuitively, the method of discretization can have a clear impact on the accuracy of any simulation. Subdividing the simulation volume with highly skewed, non-orthogonal differential elements will produce a solution with significant errors because of the numerics employed to approximate the differential equations being studied. From the discussions at the workshop, there were two primary error-inducing grid effects that were commonly discussed, non-orthogonality and skewness. However, flow alignment can also have an effect on the simulation accuracy, and will be discussed briefly.

Test Case

The error arising from grid effects can easily be seen by considering a simple 2D test case considering purely advective transport of a passive scalar (such as temperature when neglecting buoyancy and thermal diffusion). This test case was used to investigate both flow-aligned and skewness effects. Once the solution on the domain has been calculated, a local error at position (x, y) can be computed as

$$\mathrm{Err}(x, y) = \frac{T_{sim}(x, y) - T(x, y)}{\langle T(x, y) \rangle} \cdot 100,$$

and a global error can be taken to be the standard deviation of this error,

$$\sigma = \sqrt{\frac{1}{N} \sum_{i,j} \left(\mathrm{Err}(x_i, y_j) \right)^2},$$

which provides a single number to quantify the total error in the solution.

It should be noted that a non-linear gradient was necessary in order to see these grid effects, as a linear profile will be perfectly represented by linear interpolations, thereby introducing no further error. However, in physically relevant flows, many non-linear gradients exist due to shear in the flow.

Flow Misalignment

When orthogonal cells are perfectly aligned with the flow, only interpolation errors are present in the solution since all face quantities can be accurately calculated from the cell centers. However, when the flow passes at an angle relative to these orthogonal cells, a numerical mixing process occurs, resulting in artificial numerical diffusion.

These results suggest that flow misalignment introduces non-negligible numerical diffusion, which can increase the error in the solution by almost an order of magnitude above the baseline interpolation errors.

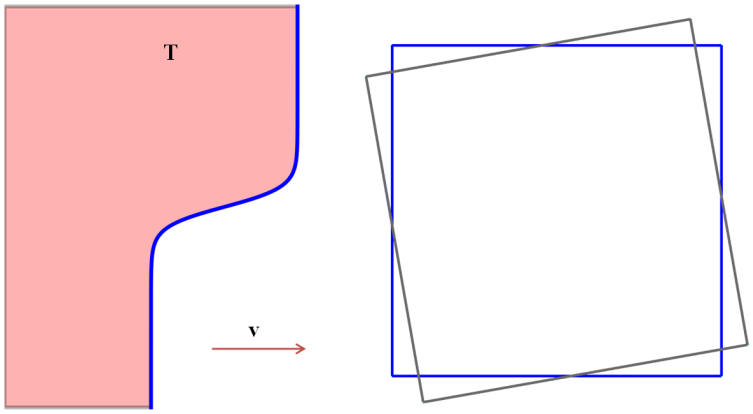


Figure 1: A profile for the passive scalar was applied at the inlet boundary condition, and the simulation domain was rotated with respect to the flow direction. The evolution of the passive scalar across the domain was then tracked to provide a measure of the numerical diffusion.

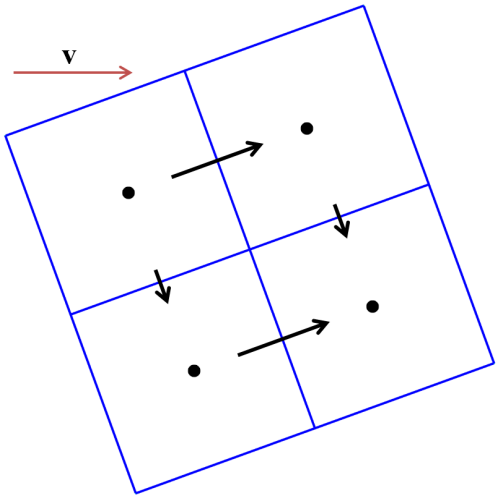


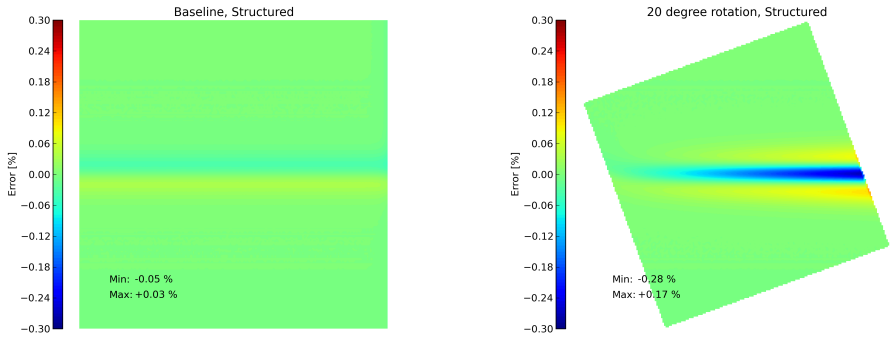
Figure 2: Grid misalignment leads to a simple geometrically imposed preferential direction due to the projected area.

Non-Orthogonality

Non-orthogonality is a significant contributor to error in OpenFOAM because of the way that gradients are calculated at cell faces. By default, gradients at cell faces are calculated by taking the difference between values at neighboring cell centers and dividing by the appropriate distance. This results in an approximation for the gradient in the direction of the vector connecting the two cell centers. However, when a term being computed in the system requires the dot product of the gradient with the face area, this can introduce significant error due to the misaligned face normal and cell-center vectors, as illustrated in Figure 4.

This term is especially critical for diffusive terms, since the discretization of the Laplacian leads to

$$\mu\nabla^2\phi \rightarrow \int_V \nabla\cdot(\mu\nabla\phi)\ \mathrm{d}V \rightarrow \sum_f \mu_f \mathbf{S}_f\cdot(\nabla\phi)_f\,,$$



(a) (b)

Figure 3: Solution error for orthogonal mesh at (a) 0 degrees and (b) 20 degrees with respect to the flow shows appreciable error when not flow-aligned.

where ϕ represents a quantity being solved for and μ is some diffusivity (be it viscosity, thermal diffusivity, etc.), and $\mathbf{S}_f$ represents the directional area of each face, f.

It is possible to correct for this grid effect by increasing `nNonOrthogonalCorrectors` in the OpenFOAM `fvSolution` dictionary. In practice, this begins to contribute significant errors when the angle between the face normal and cell-center vector is above 70 or 80 degrees. However, the simplistic test case outlined above does not suffer from non-orthogonality effects since the equations being modeled do not contain a Laplacian term.

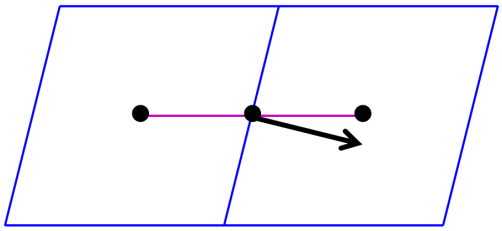


Figure 4: Non-orthogonality arises from misalignment between face normal vectors and the vector connecting adjacent cell centers (which represents the approximation for the gradient across the shared face).

Skewness

Skewness can introduce a significant numerical diffusion and is of particular concern in OpenFOAM due to the methods used to compute fluxes between adjacent cells. The `skewCorrected` method can help mitigate skewness effects to some degree, but simulated experiments show that while this decreases the errors somewhat, it does not completely eliminate the grid effects from the solution.

Skewness is of particular importance for convective derivatives, as these derivatives require the value on the face to be computed when calculating the flux to the adjacent cell. This can be seen in the discretization of the convective term,

$$(\mathbf{v}\cdot\nabla)\,\phi \rightarrow \int_V \nabla\cdot(\mathbf{v}\phi)\ \mathrm{d}V \rightarrow \sum_f \mathbf{S}_f\cdot\left(\mathbf{v}_f\phi_f\right)$$

where $\mathbf{v}$ is the fluid velocity, ϕ is again your variable, and $\mathbf{S}_f$ is defined as before.

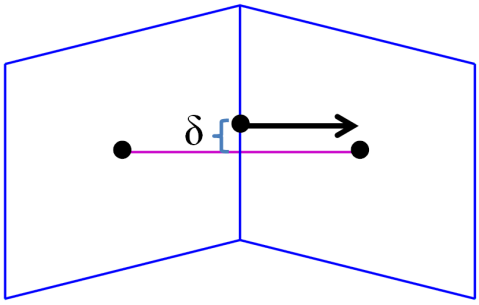


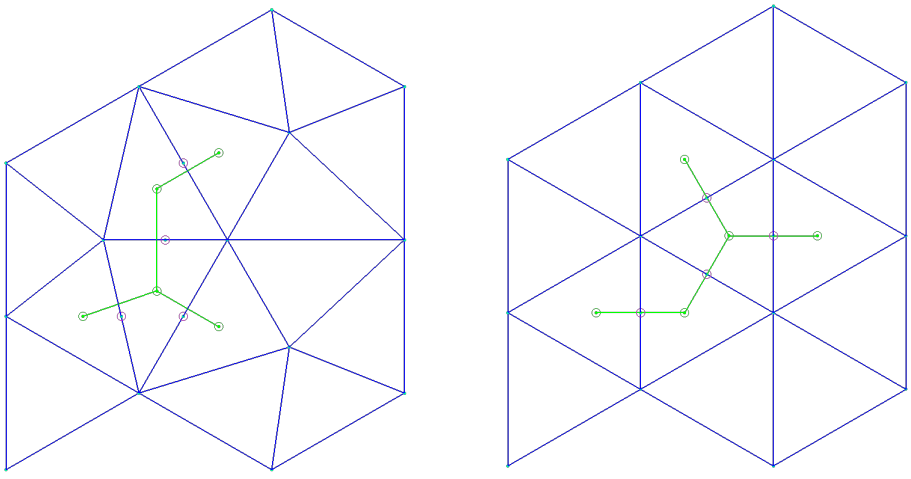
Figure 5: Skewness arises from discrepancies between the location of the face center and the location of the cell center vector.

The skewness, as it pertains to this discussion, can be easily calculated for each face by considering the ratio, ξ, of the distance δ between the actual face center and the interpolated face center locations and the distance between adjacent cell centers, D. That is

$$\xi = \frac{\delta}{D}.$$

The larger ξ the larger the error introduced by the grid skewness.

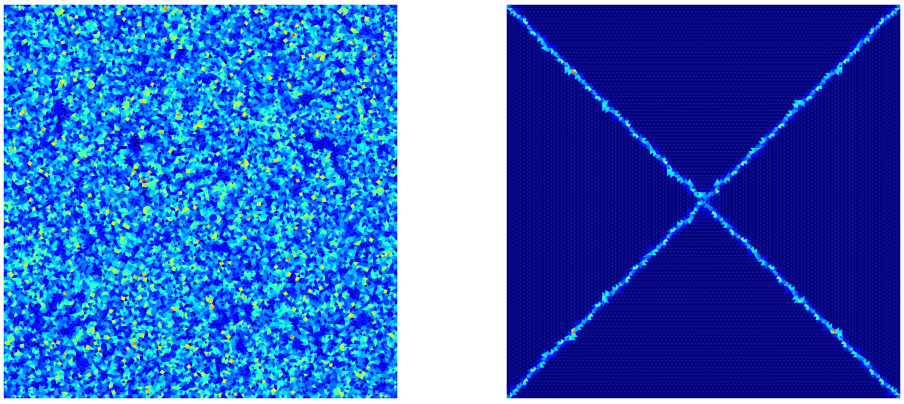
The effect of skewness is most clearly demonstrated by comparing two triangular meshes generated with different algorithms: Delaunay and Advancing Front. Delaunay meshes consist of irregularly shaped triangles arising from the Delaunay criteria regarding the inclusion of neighboring vertices in a circumscribed circle. By contrast, Advancing Front meshes create regular, equilateral triangles from the domain edges until the front collides with another from an opposing wall, at which point irregular triangles are inserted to match the two fronts. For this purpose, the majority of Advancing Front meshes have zero skewness, while almost all cells in a Delaunay mesh have non-zero skew. This can be seen in Figure 6 by carefully noting the location of the face centers relative to the location where the cell center vectors cross the face.



(a) (b)

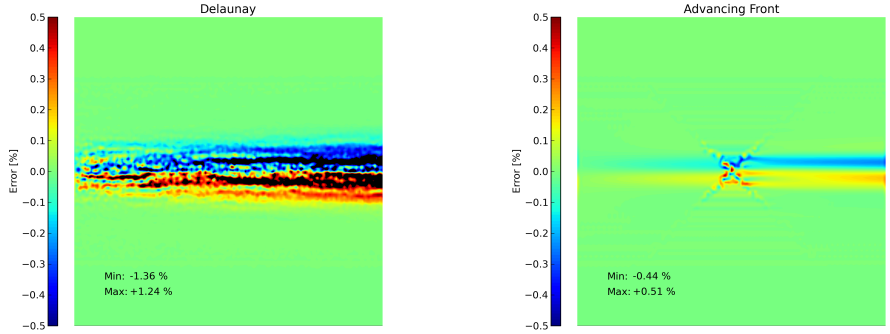
Figure 6: Delaunay (a) and Advancing Front (b) triangular domains.

For comparison, the maximum skewness can be calculated for each cell. Comparing a square domain populated with these two algorithms as seen in Figure 7 along with the solutions from these grids in Figure 8, it is clear that the Delaunay mesh has considerably more skewness and consequently higher numerical diffusion. However, it should be noted that due to the somewhat random orientation of each of the triangles in Delaunay meshes, they are far less sensitive to flow alignment, as seen in Figure 9. The effects of skewness are likely the primary reason tetrahedral grids tend to be highly diffusive in OpenFOAM with the default solvers.



(a) (b)

Figure 7: Calculated maximum skewness for (a) Delaunay and (b) Advancing Front triangular domains.



(a) (b)

Figure 8: Calculated error of solution on (a) Delaunay and (b) Advancing Front triangular domains.

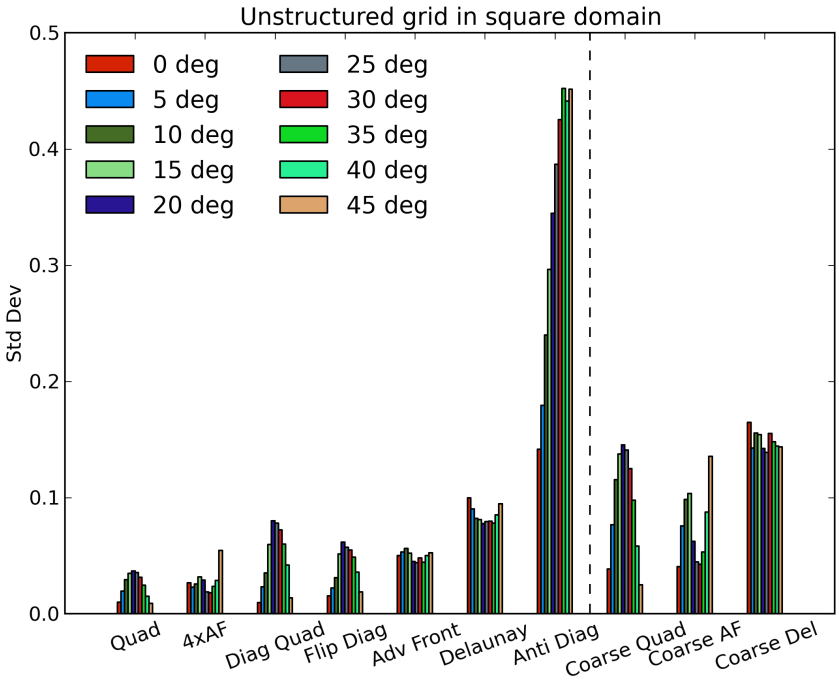


Figure 9: The effects of flow alignment on the solution accuracy can be seen to vary considerably based on the type of grid being considered.

Best Practices

1. Use orthogonal, flow-aligned cells whenever possible
2. Hexahedral elements are preferred over tetrahedral elements due to their three face pairs that contain information about fluxes in all three dimensions
3. Use skewness corrected schemes (`skewCorrected linear`, `skewLinear`, etc.) on tetrahedral or contorted hexahedral meshes
4. set `nNonOrthogonalCorrectors` to 1 when non-orthogonality is present in the grid and monitor convergence
5. Advancing front surface meshes should produce prismatic layers with less skewness (and therefore less numerical diffusion
6. Avoid sudden jumps in cell orientation, particularly in regions of high gradients

Conclusions

Numerical diffusion is a significant concern for OpenFOAM solvers, particularly when considering triangular meshes. However, implementing more advanced algorithms that rely on additional neighbors to compute higher-order interpolations of the face gradients should reduce the error at the cost of adding slightly more computational steps. In short, triangular meshes will

work in OpenFOAM, but the presence of numerical diffusion must be taken into account when interpreting the results.

Further work investigating three-dimensional diffusion on tetrahedral grids may shed more light on the problem at hand and suggest ways to improve the quality of solutions obtains from tetrahedral grids. It should be noted that tetrahedra suffer from a potentially limiting geometric constraint when computing higher-order interpolations of gradients. That is to say, it is possible for all of the face neighbors of a tetrahedral element to lie within the same plane. In such a situation, including the face neighbors will not provide any information about the gradient in the third dimension. For this reason, it may be necessary to not only include face-neighbors, but edge neighbors as well.

Because of the formulations in OpenFOAM, hex-dominant or polyhedral meshes have fewer restrictions in terms of the numerics. However, the flow field in the boundary layer must also be accurately captured by the grid, which poses a challenge to top-down preprocessing approaches. As commonly concluded in CFD, the solution comes down to the mesh.

Additional Resources

- `http://sourceforge.net/projects/openfoam-extend/files/OpenFOAM_Workshops/`
- `http://openfoamwiki.net/index.php/Handy_links`
- `http://openfoamworkshop.org/index.html`