Part C: Final Project Proposal

High-order-accurate methods for simulation of fluid flow problems, are among the highly pursued research frameworks in Computational Fluid Dynamics, due to their potential to reduce the computational effort required for a given level of solution accuracy. My current thesis project is to generalize our in house high order turbulent viscous flow finite volume solver[2] to handle three dimensional geometries.

Conventional mesh generators create linear cells near the curved boundaries. However, high order numerical methods must account for the curved boundary in order to maintain their order of accuracy. For an isotropic mesh curving the faces on the boundary is sufficient. On the other hand, for anisotropic meshes, which are common in turbulent flow simulations, the mesh deformation should be propagated inside the domain to prevent unacceptable self intersecting elements.

To convert a three-dimensional anisotropic linear element mesh to a boundary conforming curved mesh, a solid mechanics analogy is used. The initial linear mesh is modeled as an elastic solid. When the boundary of the solid is deformed to match the curved boundary, the internal deformation of the solid will create the desired conforming curved mesh.

The equations modeling the elastic solid, i.e. Navier Equations, are written as[4]:

$$\nabla \cdot \sigma = 0$$

Where σ is the symmetric three dimensional stress tensor and related to the displacement vector through the formulas :

$$\begin{aligned}
\sigma_{11} &= (2\mu + \lambda)u_x + \lambda v_y + \lambda w_z \\
\sigma_{22} &= \lambda u_x + (2\mu + \lambda)v_y + \lambda w_z \\
\sigma_{33} &= \lambda u_x + \lambda v_y + (2\mu + \lambda)w_z \\
\sigma_{12} &= \mu u_y + \mu v_x \\
\sigma_{13} &= \mu u_z + \mu w_x \\
\sigma_{23} &= \mu v_z + \mu w_y
\end{aligned}$$

Where u, v and w denote the components of the displacement vector, μ and λ are the Lame's coefficients, and the subscripts denote partial differentiation. The Lame's coefficients are among the properties of the solid, and will be considered constant throughout the domain.

My proposed project is to solve this system of equations in three dimensions using the finite element method, as a starting step for a code that can convert linear mixed element three-dimensional meshes into curved boundary conforming ones. Based of our current two-dimensional mesh curving code we know that the Galerkin Method with Cubic Lagrange Elements is a suitable approach to solve this system of equations, and that affine mappings will work for all the mesh elements, as the boundary of the initial domain is the actual linear mesh.

Initially I was hoping to use libMesh[3] for the project, but it seems that incorporating cubic elements in libMesh is more challenging than I thought. So I will probably code up the solver myself in C++ and use Petsc[1] to solve the linear system of equations resulting from the discretization.

The following stages are proposed for the project:

WU For the warm up stage, I will solve a simpler scalar version of the problem. A Poisson problem with a manufactured solution in a simple geometry, e.g. box, cylinder, or sphere, would be a suitable candidate. It should be noted that linear mappings for boundary elements are enough for our purpose, and will be used for a cylinder or a sphere. Although the final goal is to use cubic elements, it is wise to implement linear and quadratic elements as well for verification purposes.

A An anisotropic mesh consisting of only tetrahera will be curved at this stage. As numerically describing arbitrary curved boundaries can become quite challenging, a simple geometry of two concentric spheres will be considered. Also, the method of manufactured solutions will be used to verify that the Navier Equations are being solved correctly.

B Two paths can to be followed here:

In case I write the code myself I will consider generalizing the code to handle mixed element meshes for this part, as they are quite a must in solving turbulent Navier-Stokes Equations. I will add support for hexahedra in this part, and if things go well five faced prisms and pyramids will also be implemented.

In case I can get libMesh to work and use it for this project, using different elements will be quite easy, thus I will curve a more complicated and practical geometry like a three-dimensional airfoil for this part of the project. Defining the three-dimensional geometry and the boundary conditions for the equations are the main challenges of this part.

References

[1] S Balay, S Abhyankar, M Adams, J Brown, P Brune, K Buschelman, V Eijkhout, W Gropp, D Kaushik, M Knepley, et al. Petsc users manual revision 3.5. Technical report, Technical report, Argonne National Laboratory (ANL), 2014.

[2] Alireza Jalali and Carl Ollivier Gooch. Higher-order unstructured finite volume methods for turbulent aerodynamic flows. In *22nd AIAA Computational Fluid Dynamics Conference*, page 2284. 2015.

[3] Benjamin S Kirk, John W Peterson, Roy H Stogner, and Graham F Carey. libmesh: a c++ library for parallel adaptive mesh refinement/coarsening simulations. *Engineering with Computers*, 22(3-4):237–254, 2006.

[4] W Michael Lai, David H Rubin, David Rubin, and Erhard Krempl. *Introduction to continuum mechanics*. Butterworth-Heinemann, 2009.