The Deformation Method for Grid Adaptation

James Grisham[1], Nandakumar Vijayakumar[1], Ben Hildebrand[2]
Chen Xi[2]

[1]Mechanical and Aerospace Engineering
University of Texas at Arlington

[2]Department of Mathematics
University of Texas at Arlington

MATH 5392 Final Presentation, May 2014

Outline

1. Introduction
   - Motivation
   - The process
   - Goals
2. Computational Tools
   - Pointwise
   - Fluent
   - Deformation method code
3. Implementation
   - Test case
   - Detailed process
   - Results
4. Wrapping up

Why adapt?

- For numerical solution of partial differential equations, an adequate discretization (or mesh) is required.
- Using adaptive grids, the quality and accuracy of the numerical solution can be improved.

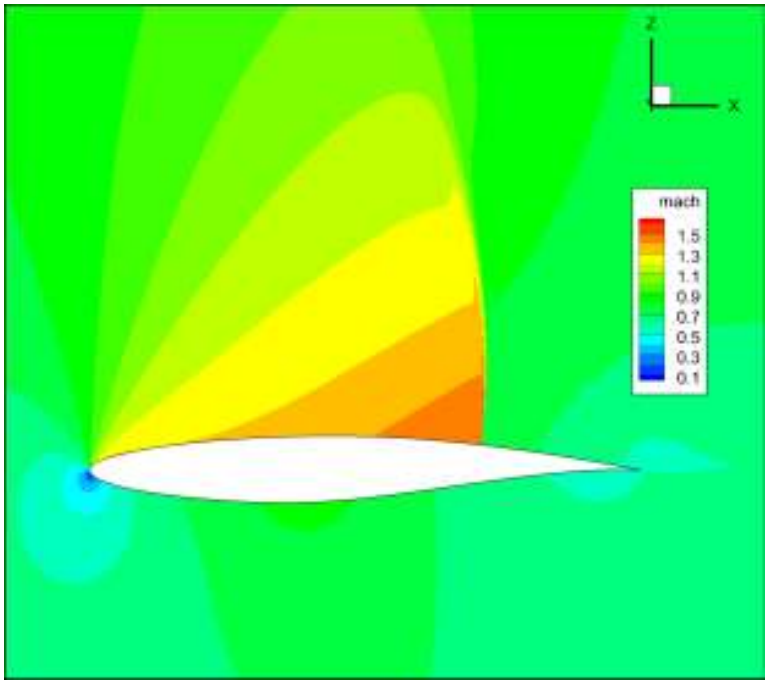


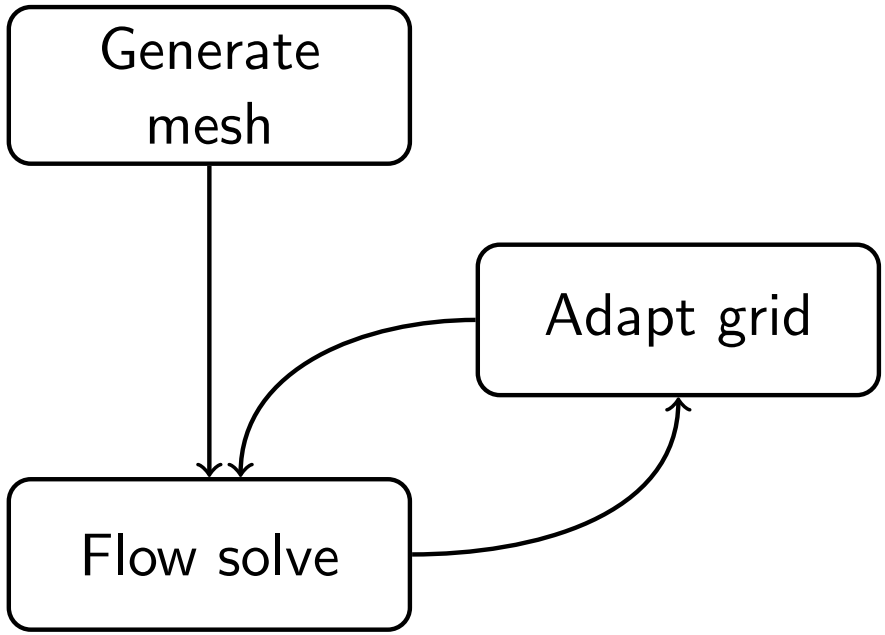
Figure: Contours of Mach number.

Why use the deformation method?

- Many moving grid techniques are capable of producing negative volumes (e.g., linear elasticity).
- The deformation method is mathematically rigorous, unlike the physics-based analogies.
- It has been mathematically proven that the deformation method will not produce negative cell volumes. However, there are always issues with the numerics.
- The changes in the point distribution are propagated throughout the entire mesh.

Explanation of the adaptation process

1. Generate geometry and mesh.
2. Perform flow solve.
3. Adapt grid.
4. Re-run flow solve to check quality of deformed grid.



Deformation method process

- Form monitor function. In this case,

$$f_0(\mathbf{x}) = \frac{1}{1 + C|\nabla p(\mathbf{x})|} \tag{1}$$

where C is a constant that controls the intensity of the adaption.

- Make the monitor function a function of time.

$$f(\mathbf{x}, t) = 1 - t + t\, f_0(\mathbf{x}) \tag{2}$$

where $0 \leq t \leq 1$.

Deformation method process, cont.

- Solve Poisson's equation with Neumann BCs ($\nabla w \cdot \hat{\mathbf{n}} = 0$).

$$\nabla^2 w = -\frac{\partial}{\partial t}\left(\frac{1}{f(\mathbf{x}, t)}\right) \tag{3}$$

- Compute node velocities ($\mathbf{V}$).

$$\mathbf{u} = \nabla w \tag{4}$$

and

$$\mathbf{V} = f(\mathbf{x}, t)\,\mathbf{u} = f(\mathbf{x}, t)(\nabla w) \tag{5}$$

- Solve ODEs for new node coordinates.

$$\frac{d\mathbf{x}}{dt} = \mathbf{V} \tag{6}$$

Goals

- Implement the deformation method for 2D structured grids.
- Perform inviscid and viscous flow solves to use as baseline data for adaptation.
- Adapt to different flow features in different speed regimes (i.e., boundary layers and shock waves).
- Compare deformation-based adaptation with ANSYS Fluent's local refinement method.

Responsibilities

- James Grisham
  - Code development.
  - Grid generation.
- Nandakumar Vijayakumar
  - Grid generation.
  - CFD simulations.
  - Fluent adaptation.
- Ben Hildebrand
  - Poisson solver.
  - RK2 ODE solver using MATLAB.
  - Understanding the math.
- Chen Xi
  - Understanding the math.

Grid generation

- A series of Pointwise Glyph script were developed to generate viscous and inviscid O- and C-grid topologies for an airfoil.
- Glyph script was also developed to generate grid for a supersonic duct case.
- Grids were exported in CGNS (CFD General Notation System) format.

Flow solve

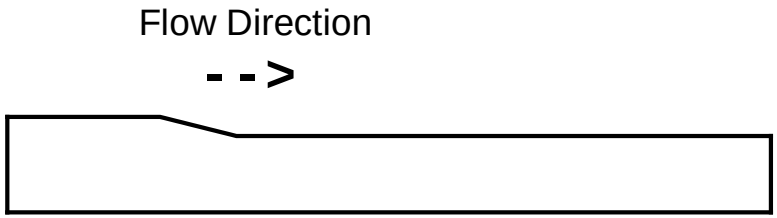
- ANSYS Fluent was used to perform inviscid and viscous flow solves.
- Plans to use a NASA Langley-developed flow solver named CFL3D in the near future.

Deformation method code

- General code was developed using C++.
- Reads and writes CGNS-formatted grid and solution files.
- Object-oriented approach with a general C++ class that contains all the necessary methods.
- OOP allowed for stand-alone testing of each method.

Example test case

- Supersonic duct case was used to test the code.
- $M = 3.5$



Initial Mesh

- 150 $\times$ 80 structured grid

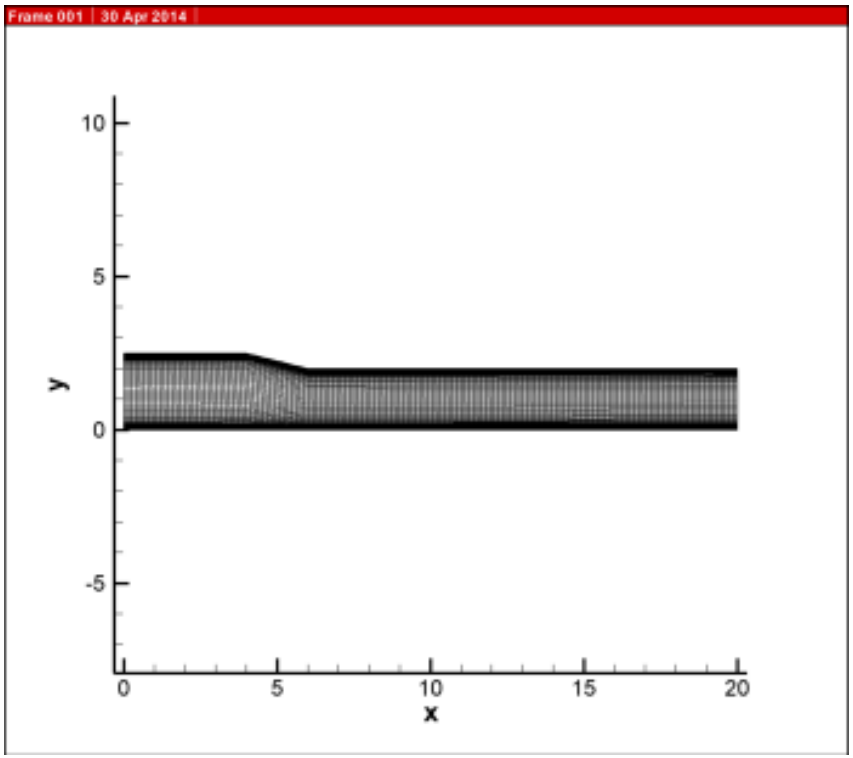


Figure: Initial grid.

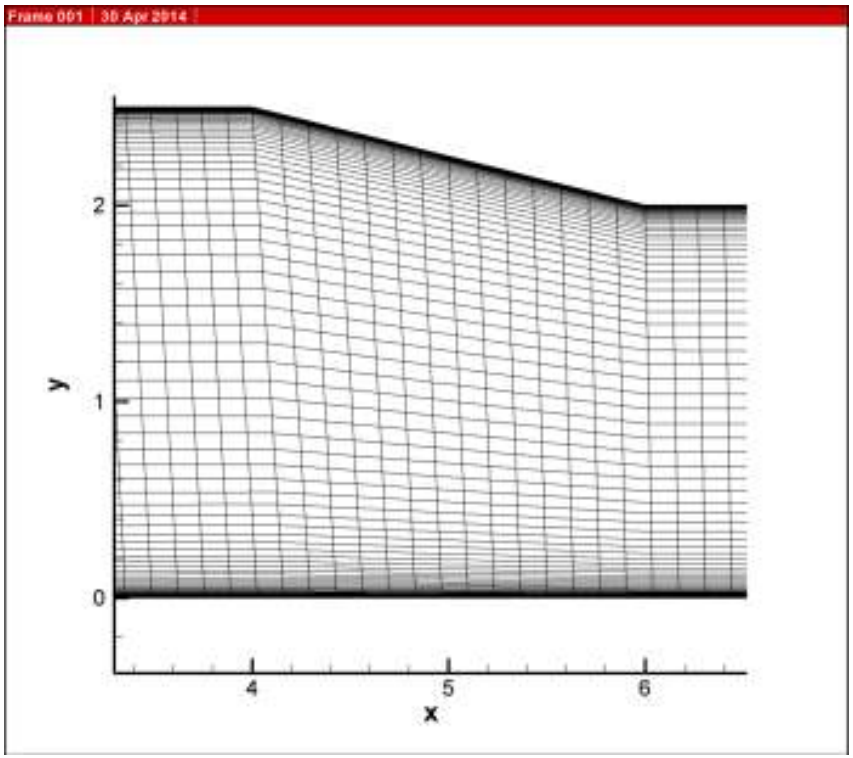


Figure: Initial grid close-up.

Fluent output

- Interesting flow features.
- Reflecting oblique shocks.
- Shock-boundary layer interaction.

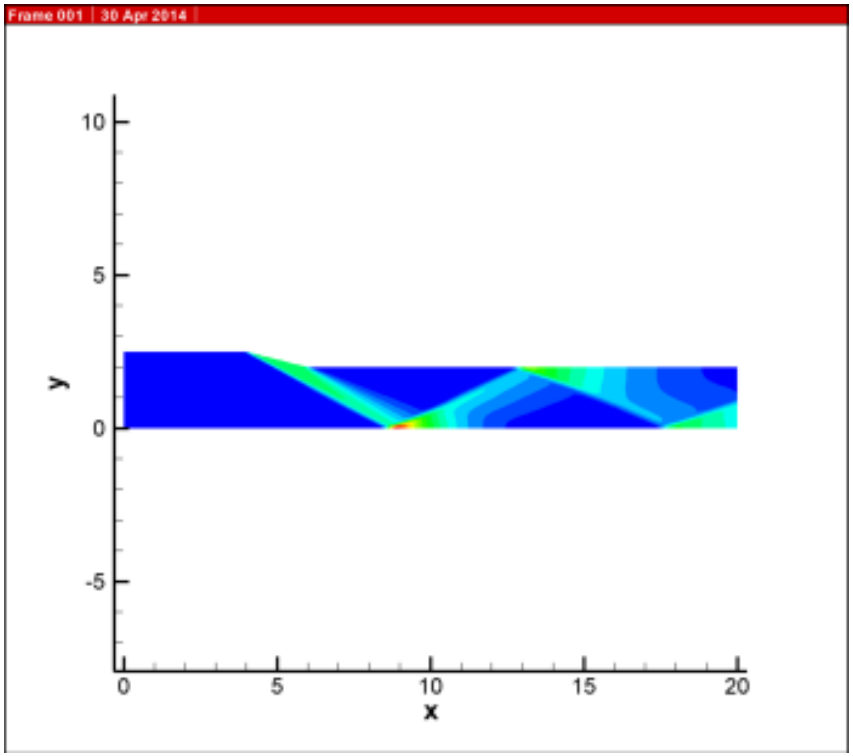


Figure: Contours of static pressure.

Importing grid and data

- C++ code is linked with the CGNS libraries so that binary files can be read and written.
- Majority of pre-processing codes, flow solvers, and post-processing codes are capable of reading and writing CGNS files.
- Fluent writes flow solve data in an unstructured format.
- Method is included in general class that re-orders the grid one time as a pre-processing step.

Computing gradient of pressure

- Gradient of pressure must be computed to form the monitor function given by Eq.(1).
- Accomplished using the finite difference method with the following inverse transformation for nonuniform grids:

$$x = x(\xi, \eta) \tag{7a}$$

$$y = y(\xi, \eta) \tag{7b}$$

- Following metrics are computed:

$$\frac{\partial x}{\partial \xi}, \frac{\partial x}{\partial \eta}, \frac{\partial y}{\partial \xi}, \frac{\partial y}{\partial \eta}$$

Computing gradient of pressure, cont.

- Jacobian determinant is then computed:

$$J = \frac{\partial(x, y)}{\partial(\xi, \eta)} = \begin{vmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{vmatrix} \tag{8}$$

- Partial derivative of some variable ϕ with respect to x is then given by

$$\frac{\partial \phi}{\partial x} = \frac{1}{J}\left(\frac{\partial \phi}{\partial \xi}\frac{\partial y}{\partial \eta} - \frac{\partial \phi}{\partial \eta}\frac{\partial y}{\partial \xi}\right) \tag{9}$$

Gradient results

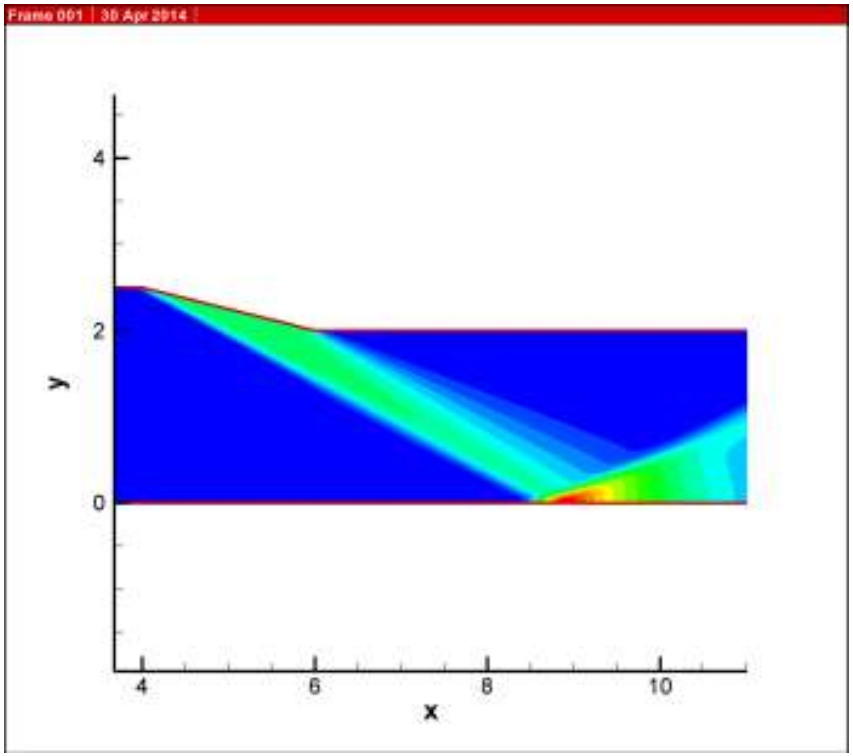


Figure: Contours of p.

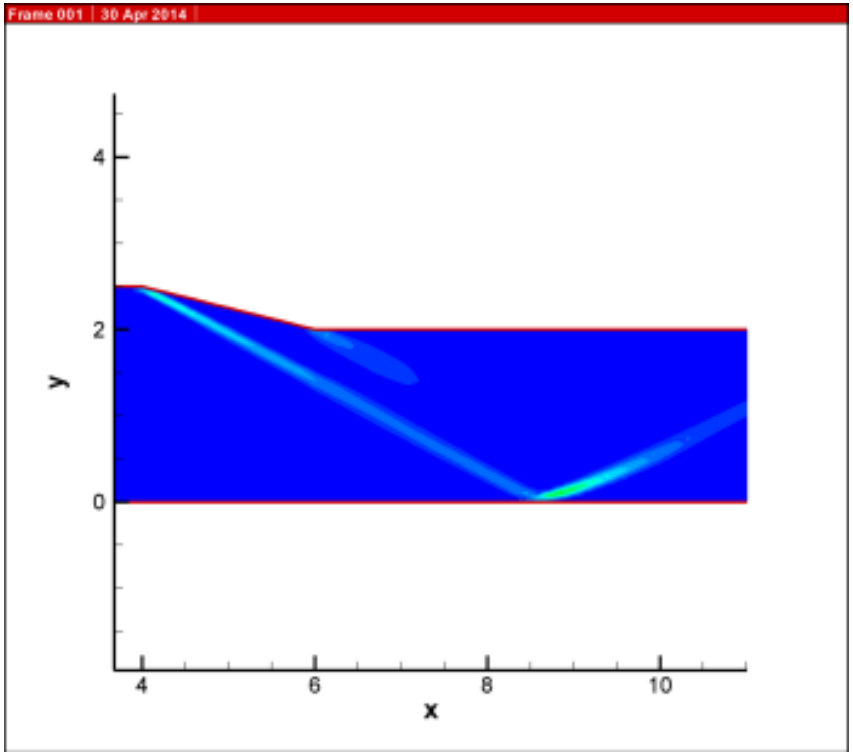


Figure: Contours of $|\nabla p|$.

Forming monitor function

- Monitor function is formed as follows:

$$f_{i,j} = \frac{1}{1 + C|\nabla p_{i,j}|} \quad (10)$$

- Monitor function is small where gradients are large.
- Used to control cell size.

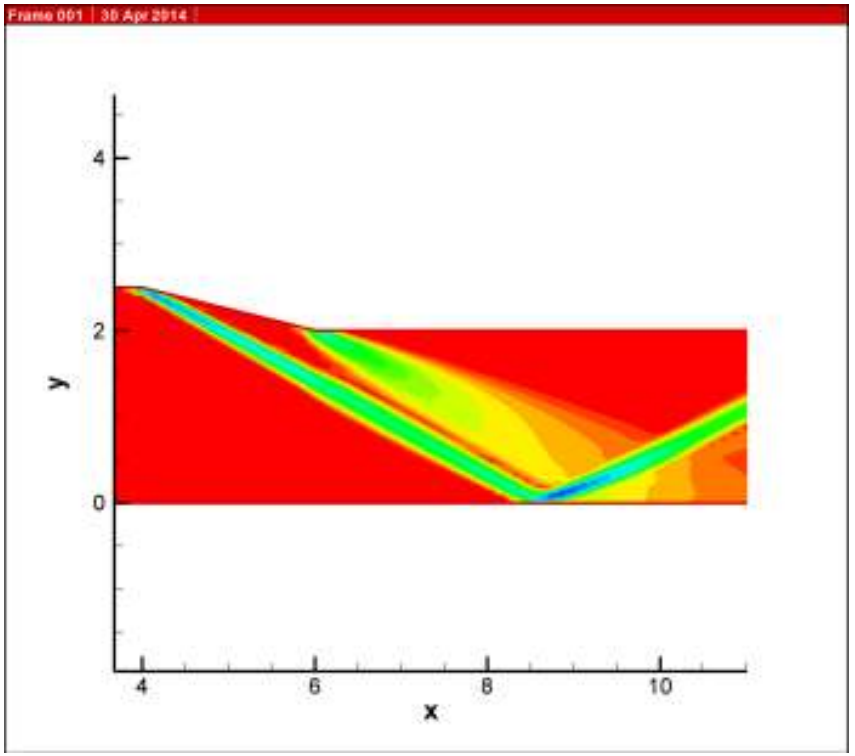
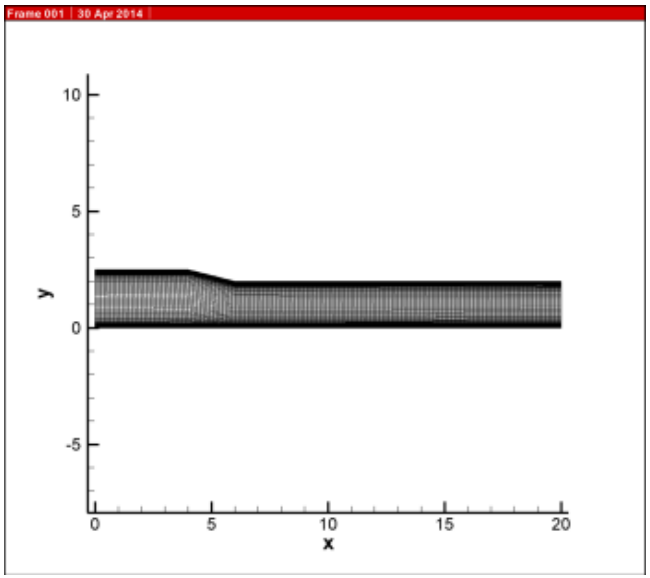
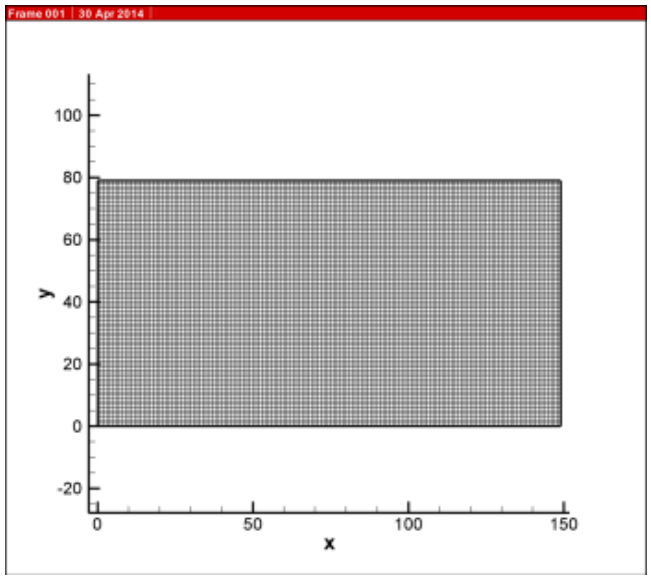


Figure: Contours of the monitor function.

Transformations

- Process can be thought about using three grids: the original nonuniform grid in the physical plane, a uniform grid in the computational plane that will be deformed, and a uniform grid in the computational plane that remains unchanged.
- Monitor function is computed on the nonuniform grid.
- Deformation takes place in the computational plane.





Monitor function on transformed grid

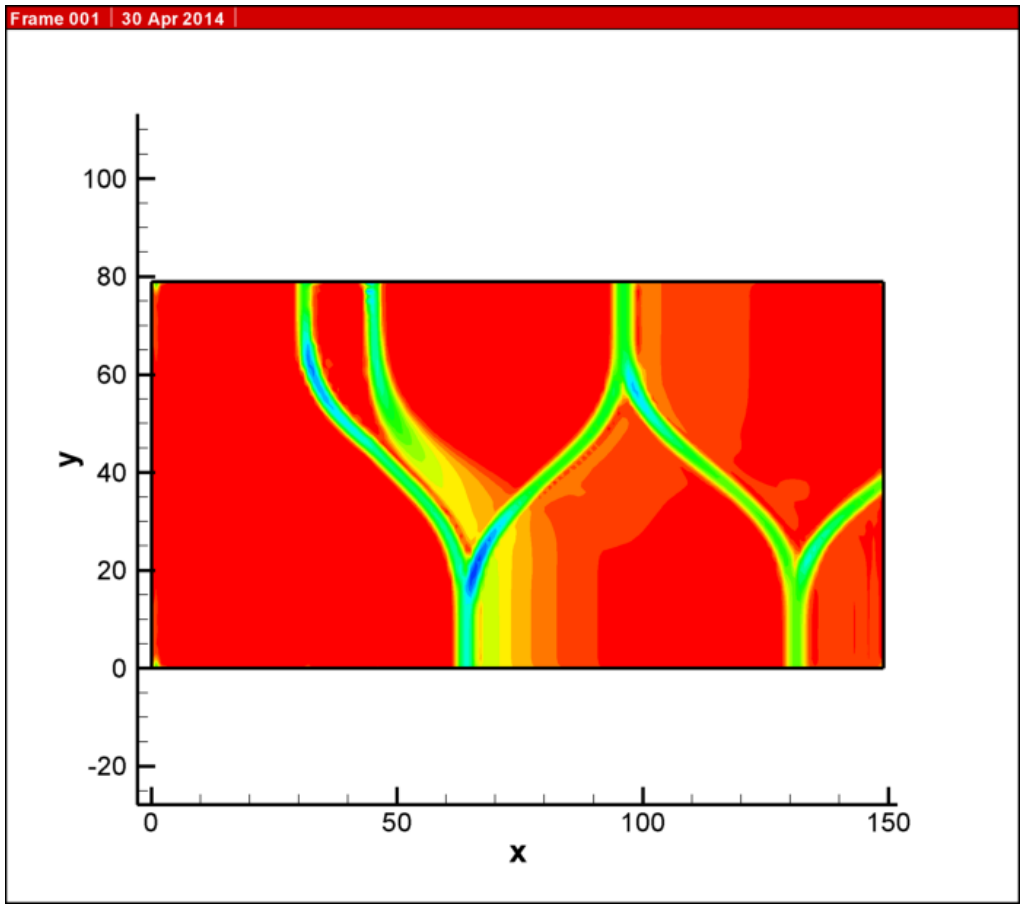


Figure: Monitor function in the computational plane.

Monitor function as a function of time

Choose number of timesteps and loop over time.

$$\nabla^2 w = -\frac{\partial}{\partial t}\left(\frac{1}{f}\right)$$

- Monitor function must be evaluated at two times so that a finite difference representation of the time derivative can be computed.

$$f^n = 1 + t^n - t^n f_0 \tag{11a}$$

$$f^{n+1} = 1 + t^{n+1} - t^{n+1} f_0 \tag{11b}$$

- Monitor functions at each time must then be normalized on the uniform grid.

Normalization

- The monitor functions at each time are normalized so that the sum of each cell area is equal to the total area of the domain as follows:

$$F(x,y,t) = \frac{\iint \frac{1}{f(x,y,t)}\, dx\, dy}{A} f(x,y,t) \qquad (12)$$

  where F is the normalized monitor function.
- Integral is accomplished using first-order accurate Gaussian quadrature.

Forming RHS of Poisson's equation

Now, the RHS of Poisson's equation can be formed using first-order accurate forward differencing.

$$-\frac{\partial}{\partial t}\left(\frac{1}{f}\right) \approx -\frac{\frac{1}{F^{n+1}} - \frac{1}{F^{n}}}{\Delta t} \tag{13}$$

Next, Poisson's equation must be solved on the uniform grid.

Solution of Poisson's equation

- Neumann boundary conditions are enforced using ghost nodes.

$$\nabla w \cdot \hat{\mathbf{n}} = 0 \tag{14}$$

- Solution is accomplished using SOR.
- General difference equation is given by

$$w_{i,j}^{k+1} = \frac{1}{4}\left(w_{i+1.j}^{k} + w_{i-1,j}^{k+1} + w_{i,j+1}^{k} + w_{i,j-1}^{k+1} - h^2 g_{i,j}\right) \tag{15}$$

$$w_{i,j}^{k+1} = w_{i,j}^{k} + \omega(w_{i,j}^{k+1} - w_{i,j}^{k}) \tag{16}$$

where k is the iteration level, ω is the relaxation parameter, and $g_{i,j}$ is the RHS of the original Poisson equation.

Solution of ODEs

- Node velocities are determined as follows:

$$\mathbf{u} = \nabla w \tag{17}$$

$$\frac{dx}{dt} = V_x = F^n u_x \tag{18a}$$

$$\frac{dy}{dt} = V_y = F^n u_y \tag{18b}$$

- Solution accomplished using Euler's method:

$$x^{n+1} = x^n + V_x \Delta t \tag{19a}$$

$$y^{n+1} = y^n + V_y \Delta t \tag{19b}$$

- Continue looping until $t = 1$.

Transforming back to physical domain

- Uniform grid in computational domain has been deformed.
- Deformed grid must be transformed back to the physical domain.
- Accomplished by interpolating the transformation.

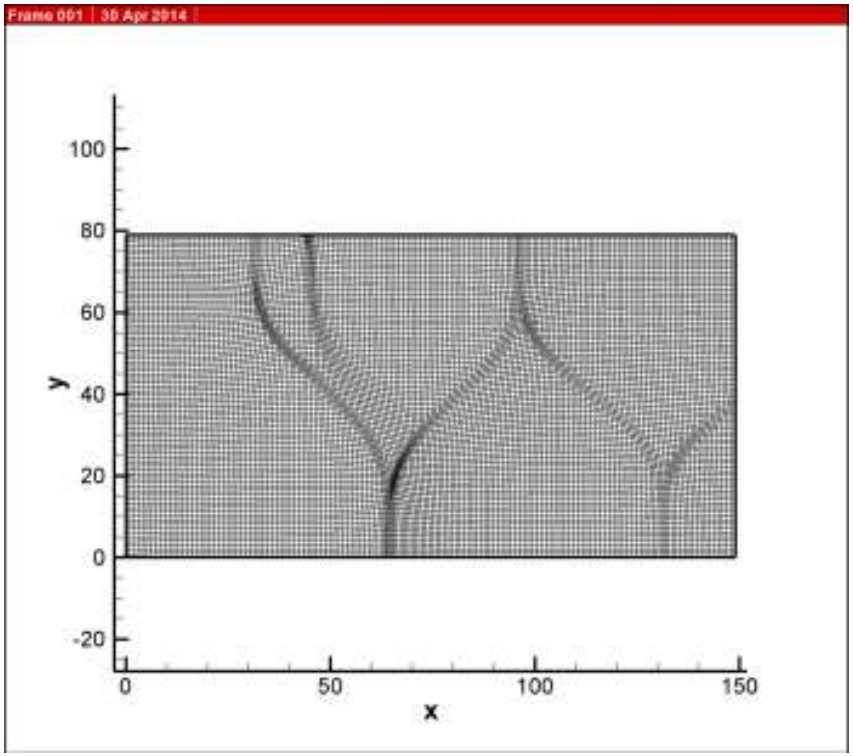


Figure: Deformed grid in the computational plane.

Transforming back to physical domain, cont.

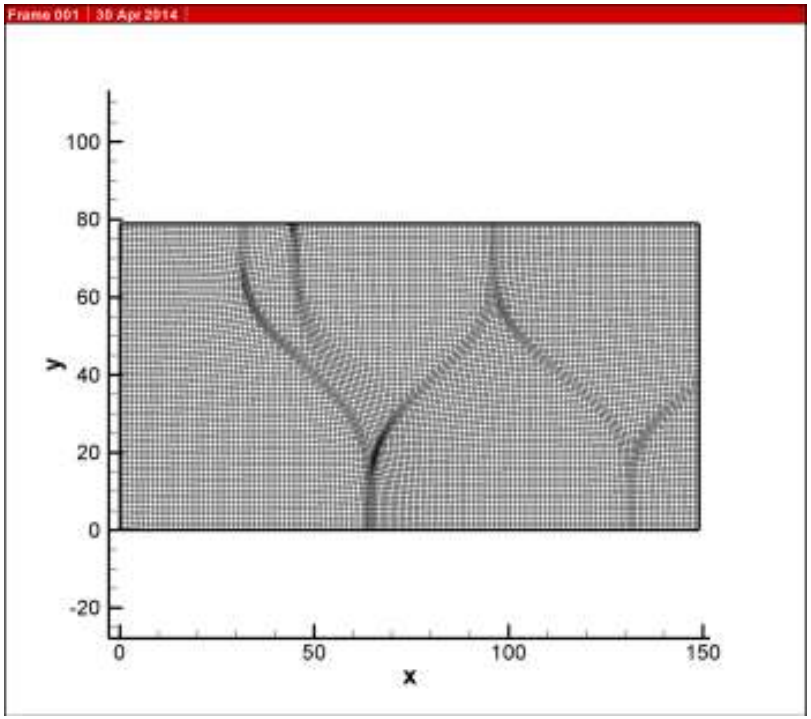


Figure: Deformed grid in the computational plane.

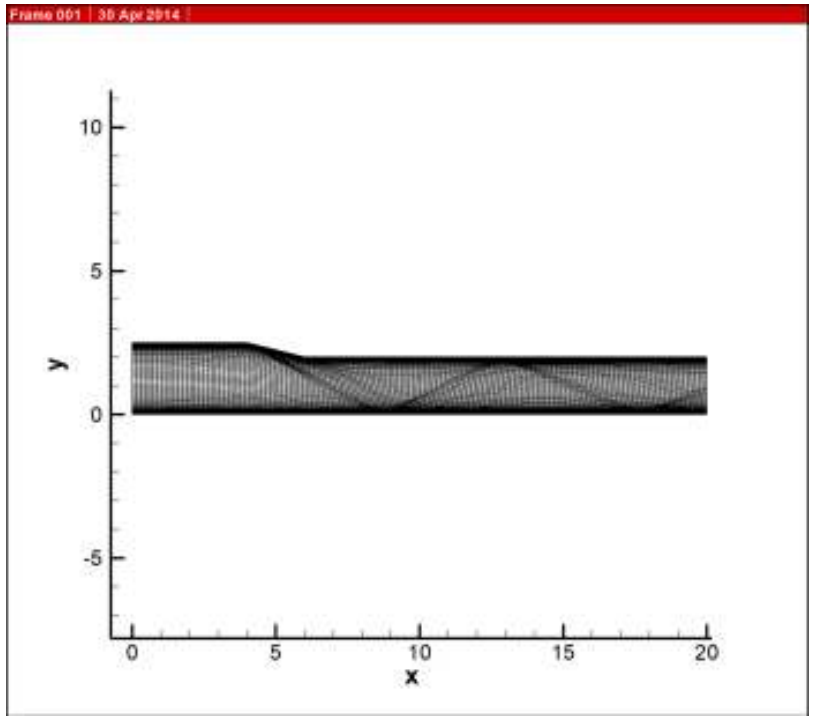
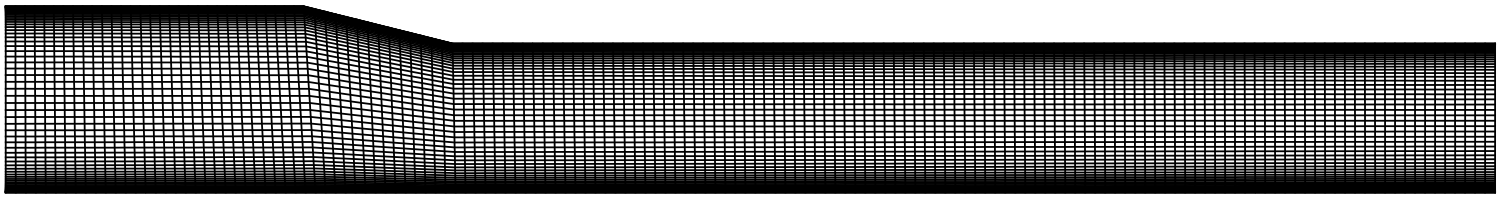
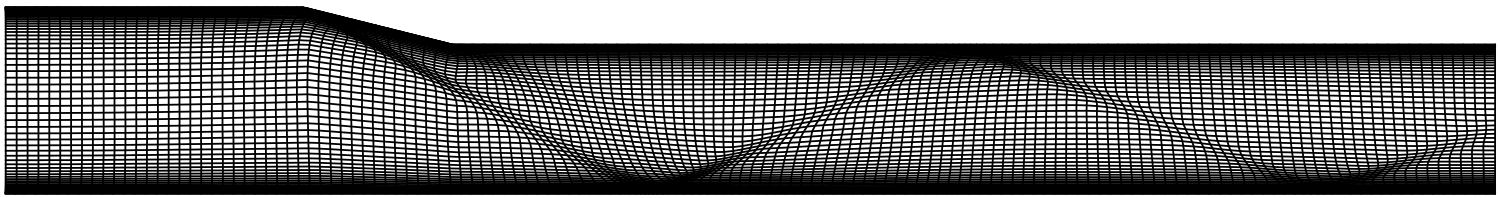


Figure: Deformed grid in the physical plane.

Comparison between initial and deformed





Comparison between initial and deformed, cont.

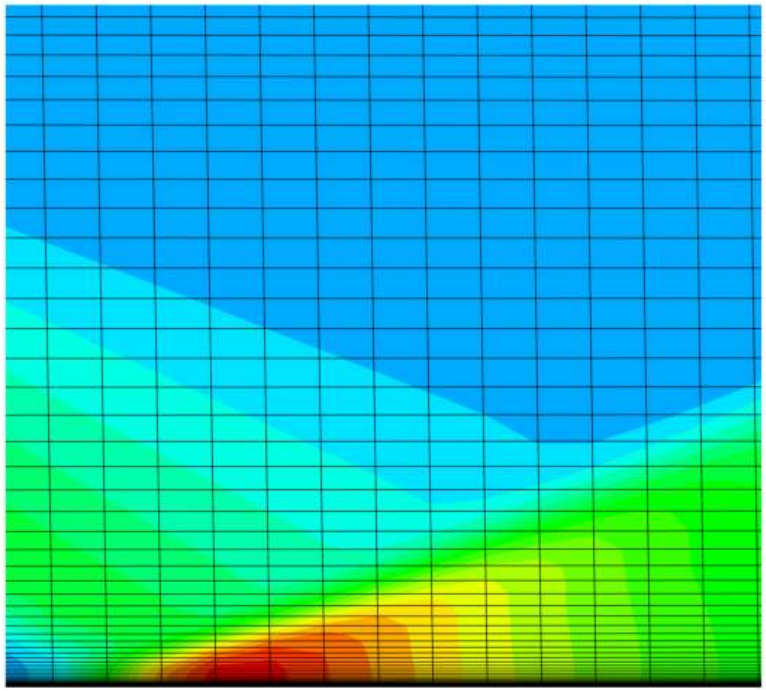


Figure: Contours of static pressure and the initial mesh.

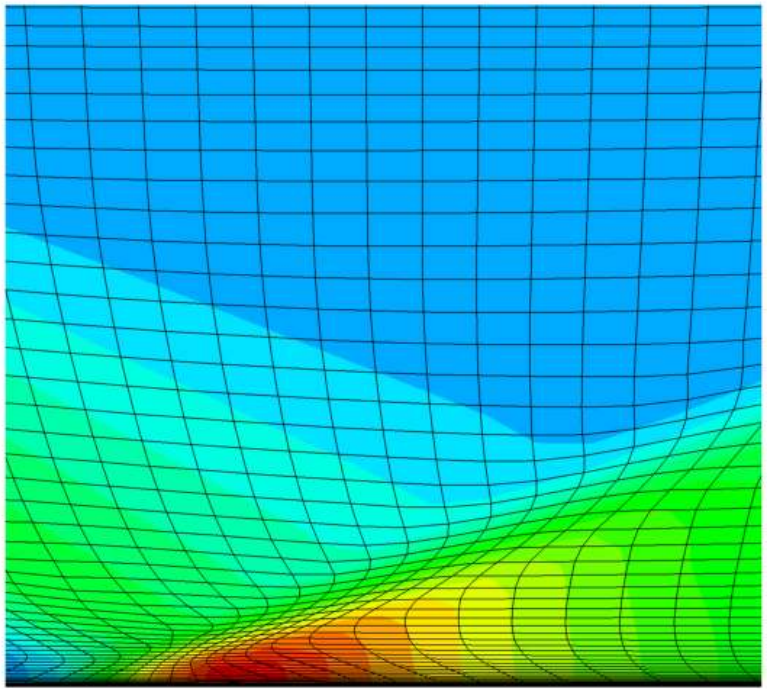


Figure: Contours of static pressure and the deformed mesh.

Summary

- A general C++ code was developed that uses the deformation method to adapt 2D, structured grids.
- Code is extensible, easily modifiable and could easily be applied to image processing.
- Code reads and writes CGNS files that are compatible with most CFD software solutions.
- Deformation method guarantees positive cell volumes (dependent upon numerics).

Potential issues

- Deformation method creates skewed cells that may cause issues with the flow solver.
- Could solve another elliptic equation to improve the skewness as a post-processing step.
- Even easier, could load the grid into Pointwise and use their elliptic smoothing capabilities.

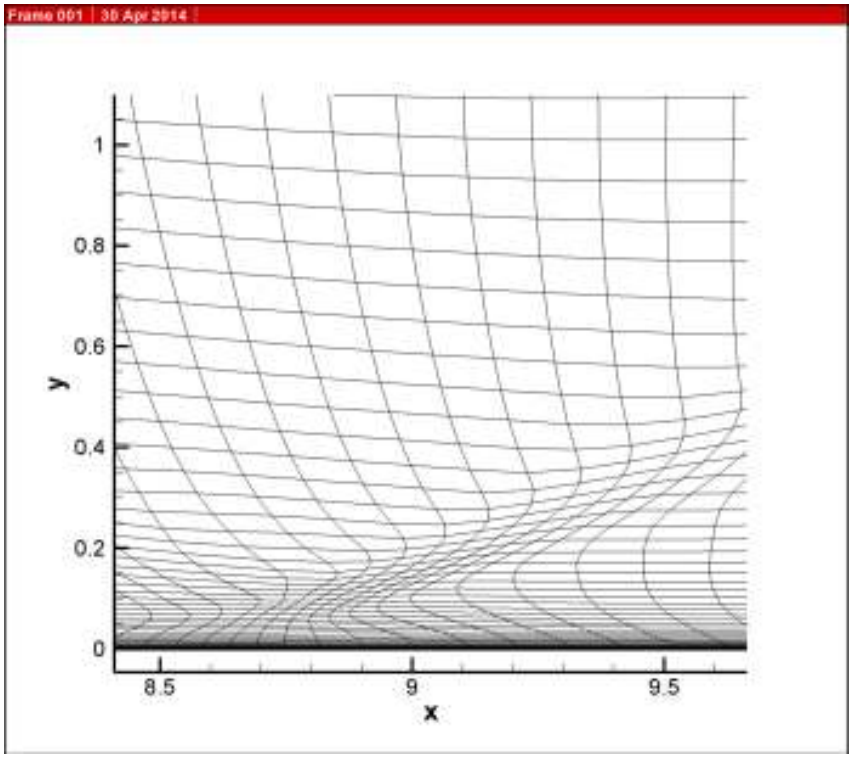


Figure: Close-up of deformed grid near shock.

Next steps

- Actually run the deformed grid in Fluent.
- Apply the code to a C-grid and O-grid (Fluent is causing an issue).
- Develop code that adapts unstructured grids using FEM.
- Develop code for 2D moving meshes and compare with spring analogy code.
- Use the current code in conjunction with CFL3D to study shock-boundary layer interation.

Questions?