UNIVERSITY OF CAMBRIDGE

CUSF Flow Simulation Workshop

Jack Brewster

CUED 14/06/2018

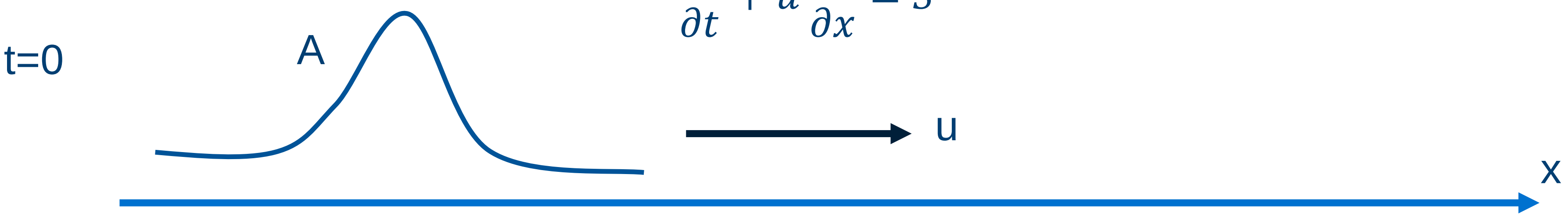
Overview

1. Introduction
2. Theory: Finite Volume (FV) methods
3. OpenFoam: Supersonic wedge
4. OpenFoam: Supersonic sphere
5. Theory: Finite Element (FE) methods
6. FEniCS: Cylinder flow
7. Future

The governing equations in fluid mechanics can often be recast as an advection equation.

- The governing equations in fluid mechanics can often be written as:

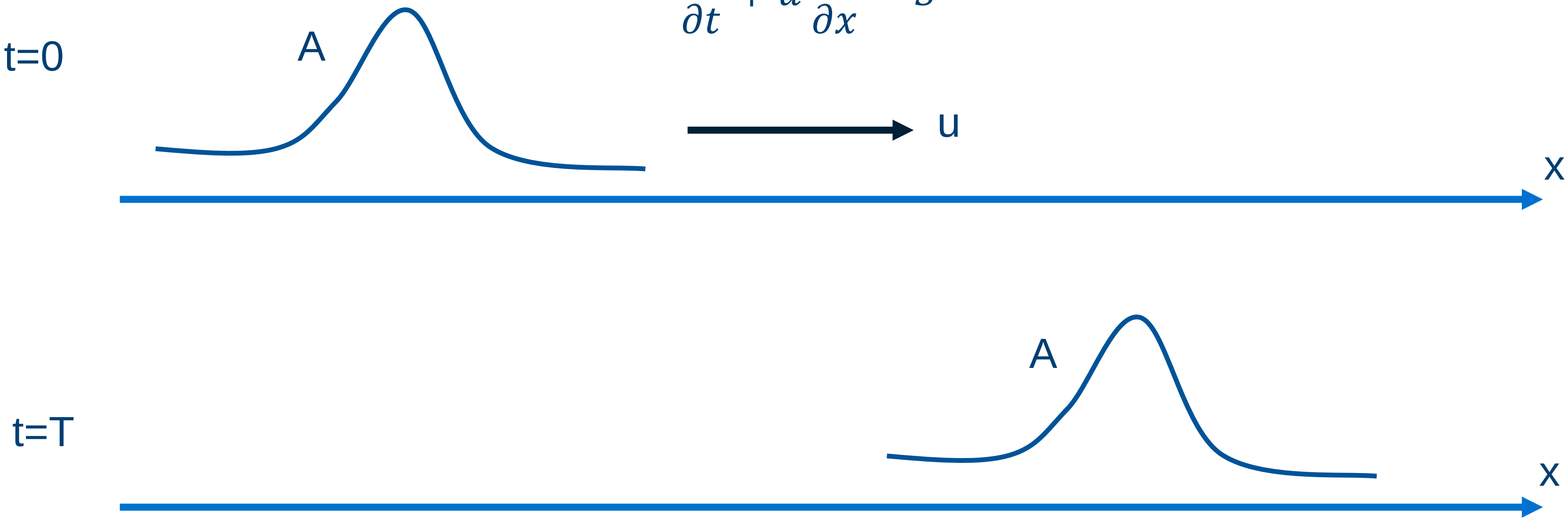
$$\frac{\partial A}{\partial t} + u\frac{\partial A}{\partial x} = S$$



The governing equations in fluid mechanics can often be recast as an advection equation.

- The governing equations in fluid mechanics can often be written as:

$$\frac{\partial A}{\partial t} + u\frac{\partial A}{\partial x} = S$$



There are three families of methods for solving PDEs.

1. Finite Difference

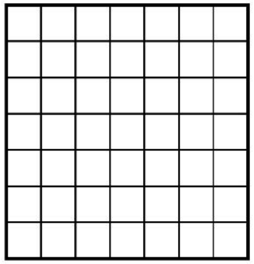
2. Finite Volume

3. Finite Element

There are three families of methods for solving PDEs.

1. Finite Difference – the intuitive approach

$$\frac{\partial A}{\partial t} \approx \frac{A_{t+\delta t} - A_t}{\delta t}, \qquad \frac{\partial A}{\partial x} \approx \frac{A_{x+\delta x} - A_x}{\delta x}$$



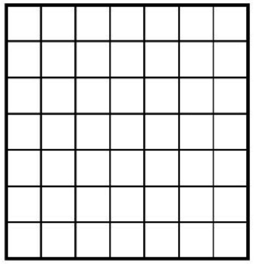
2. Finite Volume

3. Finite Element

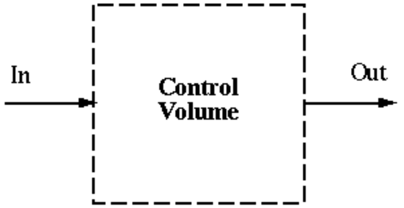
There are three families of methods for solving PDEs.

1. Finite Difference – the intuitive approach

$$\frac{\partial A}{\partial t} \approx \frac{A_{t+\delta t} - A_t}{\delta t}, \qquad \frac{\partial A}{\partial x} \approx \frac{A_{x+\delta x} - A_x}{\delta x}$$



2. Finite Volume – the engineer's approach



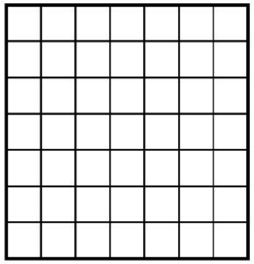
The Open Source CFD Toolbox

3. Finite Element

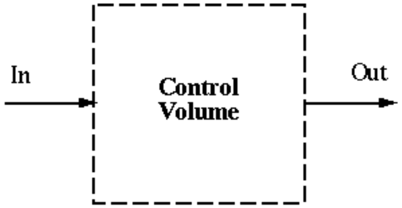
There are three families of methods for solving PDEs.

1. Finite Difference – the intuitive approach

$$\frac{\partial A}{\partial t} \approx \frac{A_{t+\delta t} - A_t}{\delta t}, \qquad \frac{\partial A}{\partial x} \approx \frac{A_{x+\delta x} - A_x}{\delta x}$$



2. Finite Volume – the engineer's approach



ANSYS

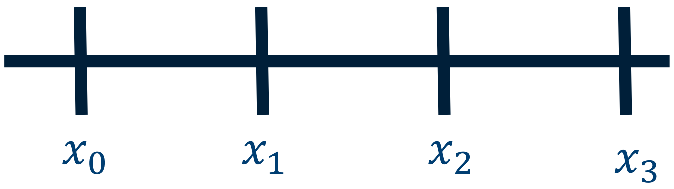
OpenFOAM
The Open Source CFD Toolbox

3. Finite Element – the mathematician's approach

FEniCS project

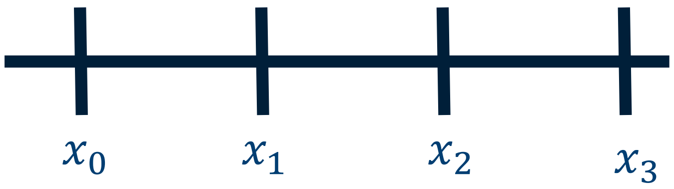
The current generation of commercial CFD solvers are all Finite Volume.

- Define a series of cells:



The current generation of commercial CFD solvers are all Finite Volume.

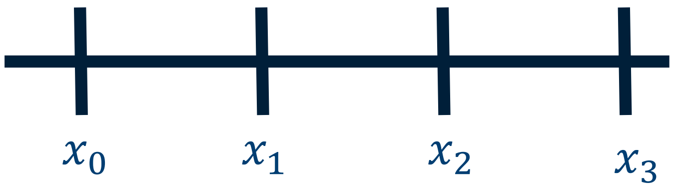
- Define a series of cells:



- Re-write governing equation as: $\frac{\partial A}{\partial t} + \frac{\partial}{\partial x}(Au) = S$

The current generation of commercial CFD solvers are all Finite Volume.

- Define a series of cells:



- Re-write governing equation as: $\frac{\partial A}{\partial t} + \frac{\partial}{\partial x}(Au) = S$

- Integrate over a cell: $\int_{x_0}^{x_1} \frac{\partial A}{\partial t}\, dx + [Au]_{x_0}^{x_1} = \int_{x_0}^{x_1} S\, dx$

The current generation of commercial CFD solvers are all Finite Volume.

- Define a series of cells:

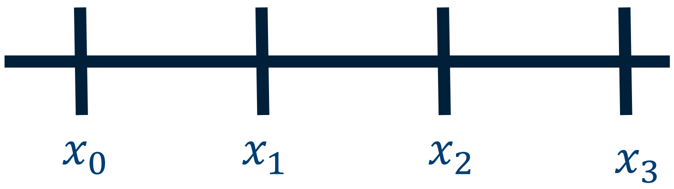
- Re-write governing equation as: $\frac{\partial A}{\partial t} + \frac{\partial}{\partial x}(Au) = S$

- Integrate over a cell: $\int_{x_0}^{x_1} \frac{\partial A}{\partial t}\, dx + [Au]_{x_0}^{x_1} = \int_{x_0}^{x_1} S\, dx$

- Divide by cell volume to get average: $\frac{\partial \bar{A}}{\partial t} = \frac{1}{\Delta x}[Au]_{x_1}^{x_0} + \bar{S}$

The current generation of commercial CFD solvers are all Finite Volume.

- Define a series of cells:



- Re-write governing equation as: $\frac{\partial A}{\partial t} + \frac{\partial}{\partial x}(Au) = S$

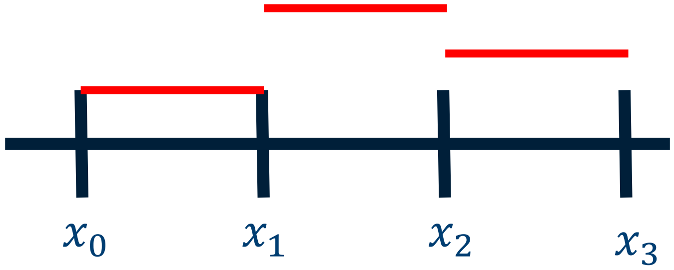
- Integrate over a cell: $\int_{x_0}^{x_1} \frac{\partial A}{\partial t} \, dx + [Au]_{x_0}^{x_1} = \int_{x_0}^{x_1} S \, dx$

- Divide by cell volume to get average: $\frac{\partial \bar{A}}{\partial t} = \frac{1}{\Delta x} [Au]_{x_1}^{x_0} + \bar{S}$

- Replace the time derivative with finite difference: $\frac{\bar{A}_{t+\delta t} - \bar{A}_t}{\delta t} = \frac{1}{\Delta x} [Au]_{x_1}^{x_0} + \bar{S}$

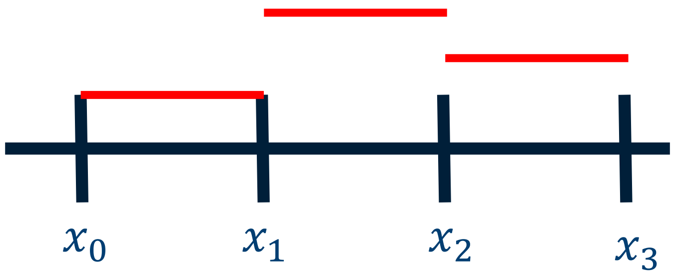
Solving in this manner is called cell-centred finite volume.

- We interpret $\bar{A}$ as the value at the cell centre

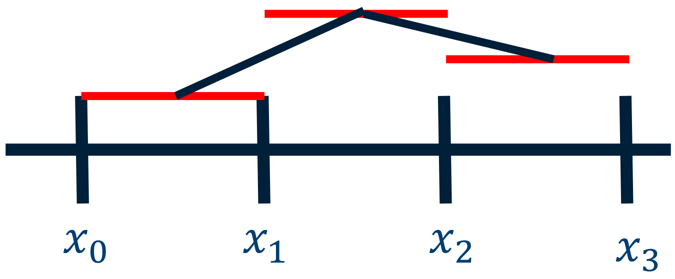


Solving in this manner is called cell-centred finite volume.

- We interpret $\bar{A}$ as the value at the cell centre

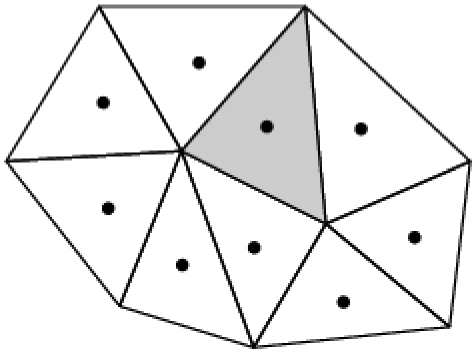


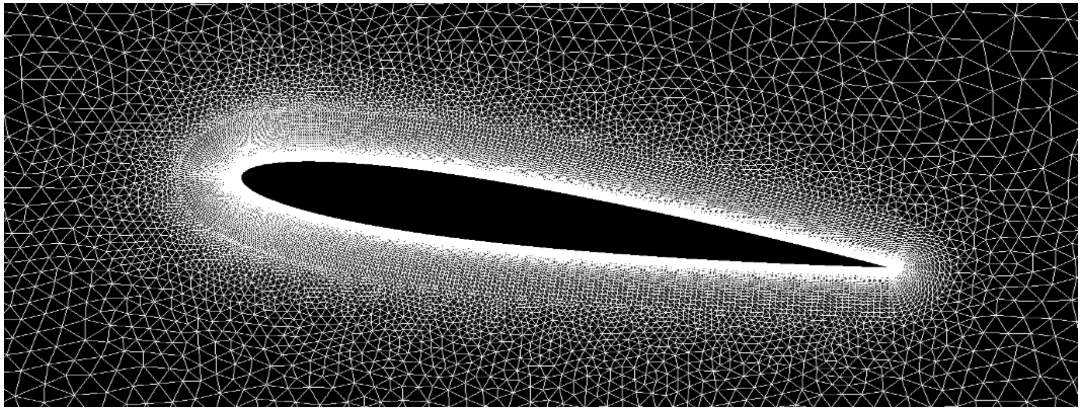
- When plotting we interpolate between the cells



This readily extends to 2 and more dimensions.

- In 2 or 3 dimensions the governing equation becomes: $\frac{\partial A}{\partial t} + \nabla.(A\boldsymbol{u}) = S$
- And the flux balance becomes $\frac{\partial \bar{A}}{\partial t} = \frac{1}{\text{vol(cell)}} \int_S \; A\boldsymbol{u}.\boldsymbol{n}\, dS + \bar{S}$



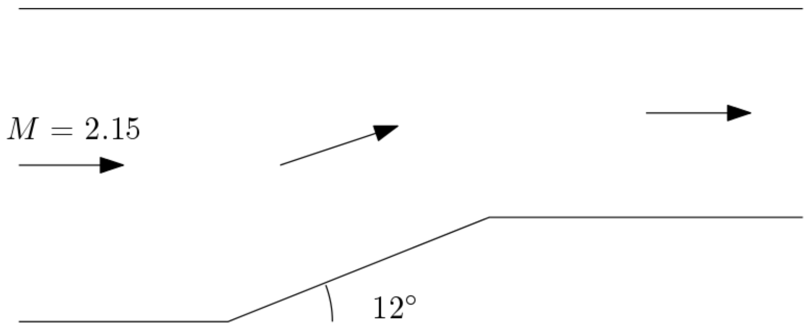


What is OpenFoam?

- OpenFoam is a series of tools and Finite Volume solvers:
- Meshing tools:
    - `blockMesh`
    - `snappyHexMesh`
- Solvers:
    - `rhoCentralFoam`
    - `sonicFoam`
    - `icoFoam`

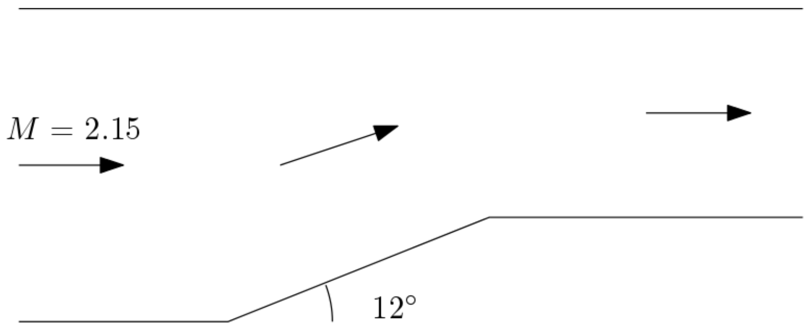
OpenFoam: Supersonic wedge

- Going to look at inviscid supersonic flow

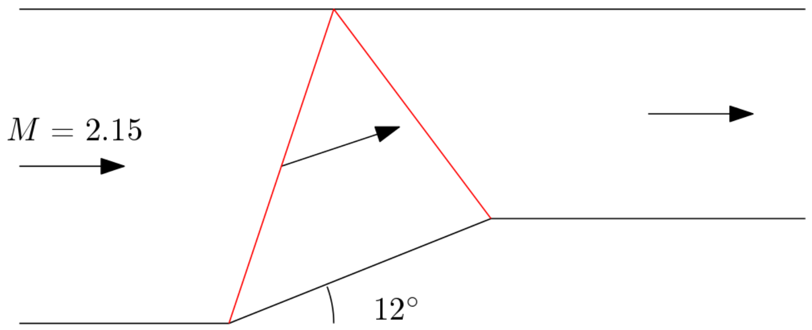


OpenFoam: Supersonic wedge

- Going to look at inviscid supersonic flow

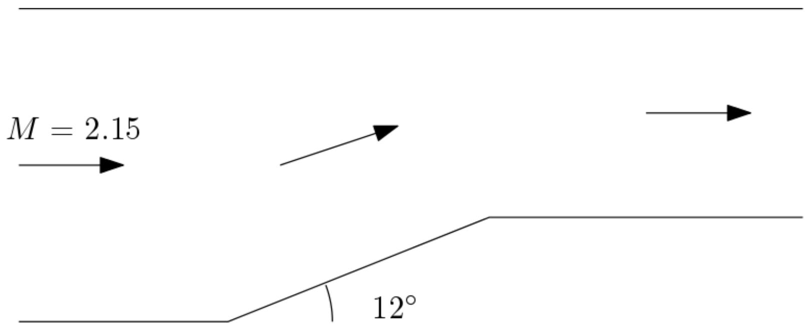


- Will set up a series of shock waves

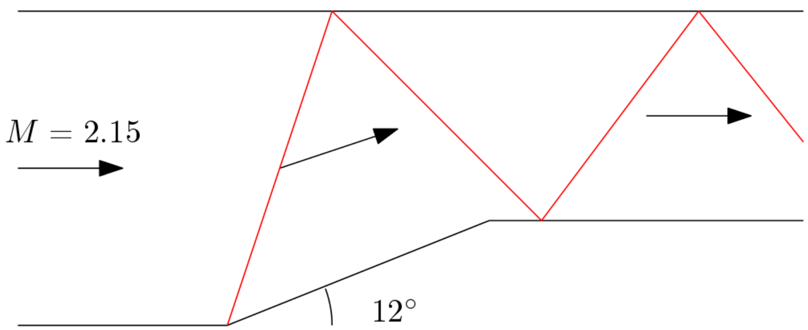


OpenFoam: Supersonic wedge

- Going to look at inviscid supersonic flow

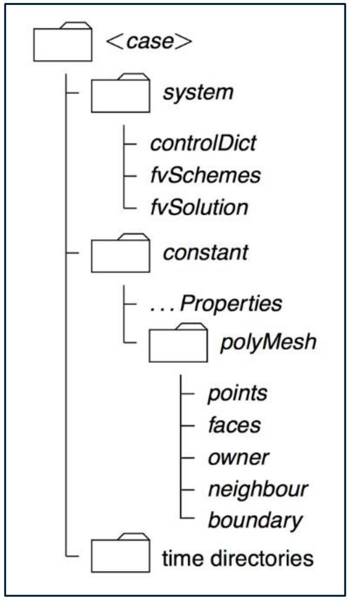


- Will set up a series of shock waves



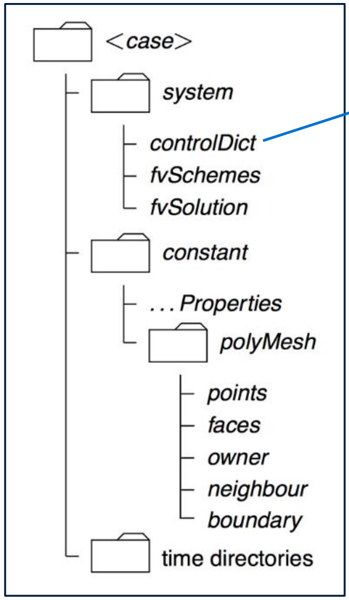
Implementing the wedge.

- `git clone https://github.com/jb803/workshop.git`
- Go into `Reflection/incomplete`
- The OpenFoam case structure:



Implementing the wedge.

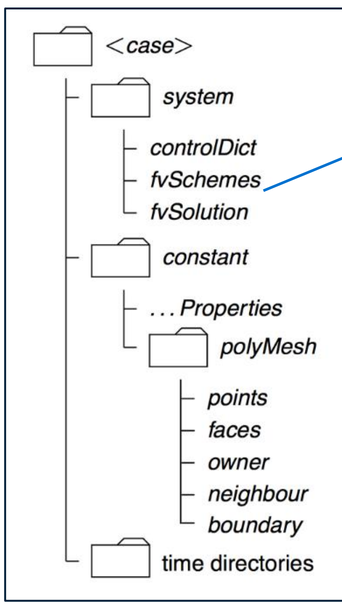
- `git clone https://github.com/jb803/workshop.git`
- Go into `Reflection/incomplete`
- The OpenFoam case structure:



Controls size of timesteps.
How often to write output etc.

Implementing the wedge.

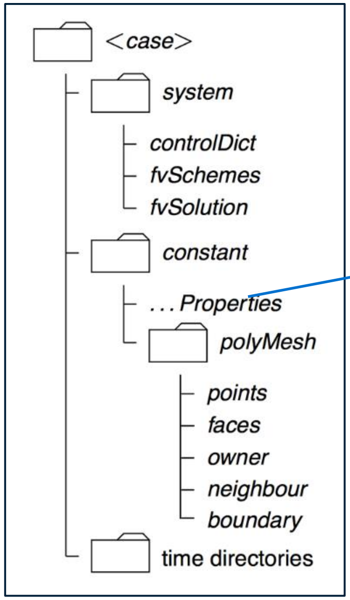
- `git clone https://github.com/jb803/workshop.git`
- Go into `Reflection/incomplete`
- The OpenFoam case structure:



How to calculate fluxes. **Very** solver specific. Copy from the examples directory.

Implementing the wedge.

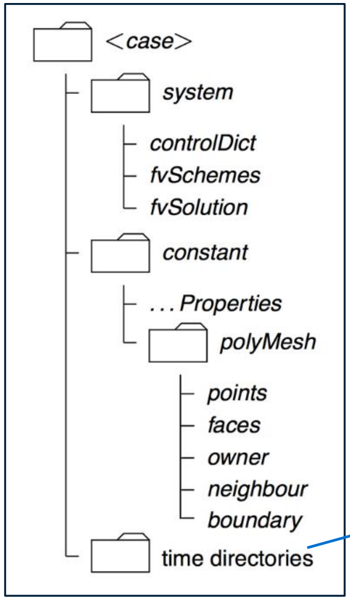
- `git clone https://github.com/jb803/workshop.git`
- Go into `Reflection/incomplete`
- The OpenFoam case structure:



Gas properties, mesh data etc.

Implementing the wedge.

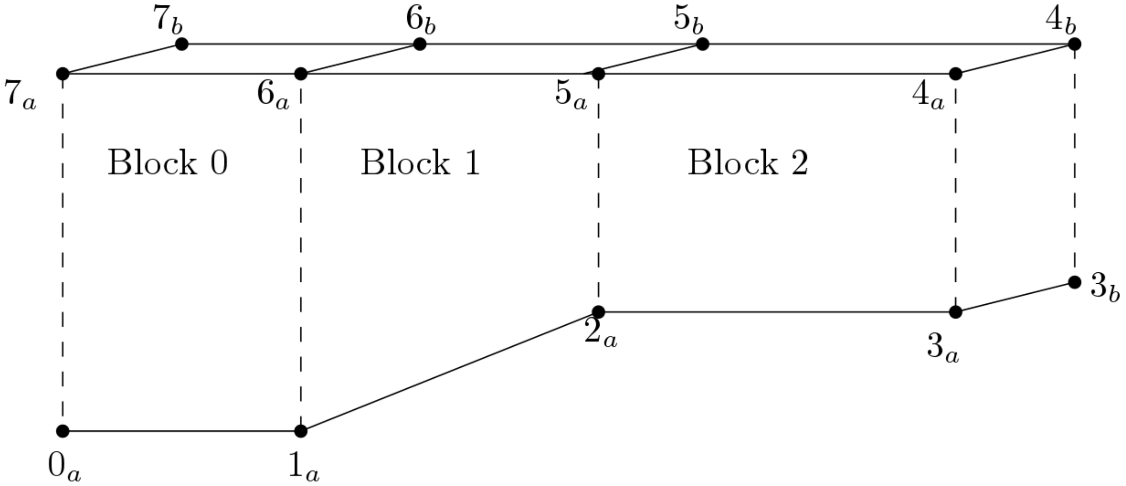
- `git clone https://github.com/jb803/workshop.git`
- Go into `Reflection/incomplete`
- The OpenFoam case structure:



Values of the velocity, pressure and temperature fields at each timestep.

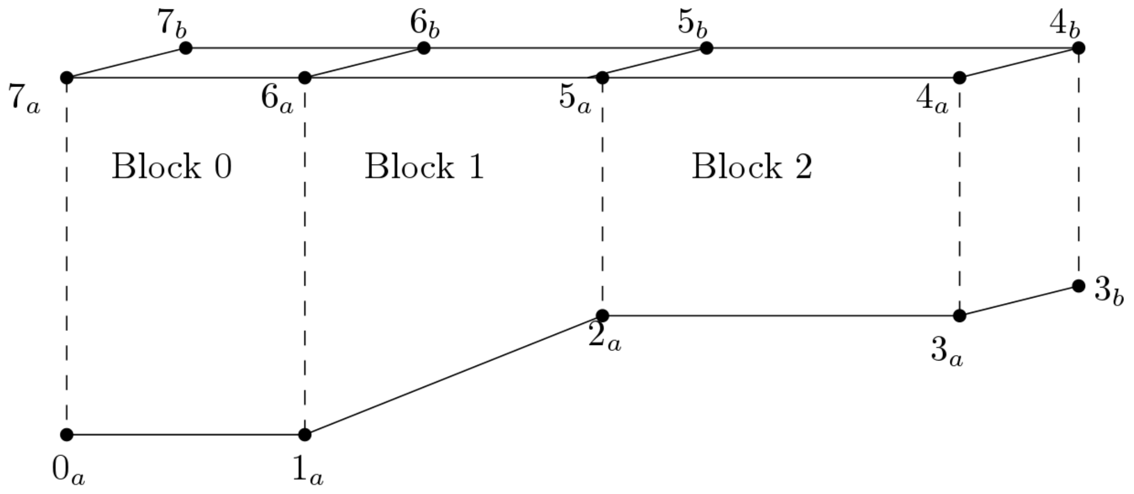
Creating the mesh.

- The mesh is made from a series of 'blocks'



Creating the mesh.

- The mesh is made from a series of ‘blocks’



- To create this mesh we will use the `blockMesh` tool.

Interactive: Editing BlockMeshDict

Creating the mesh.

- We generate the mesh with: `blockMesh`
- We check the mesh with: `checkMesh`
- We visualise with: `paraFoam`

Interactive: Timestep 0

- $a = \sqrt{\gamma R T}$

Interactive: Running `rhoCentralFoam`

- Edit `controlDict`

- Run `rhoCentralFoam`

Interactive: Visualisation with `paraFoam`

- Run `paraFoam`

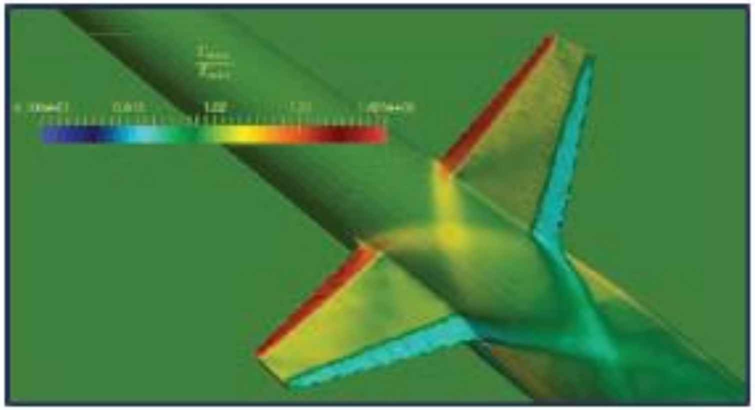
Interactive: Bonus questions

1. What happens when you increase the mesh resolution?
2. What happens when you decrease the mesh resolution?
3. What is the largest Courant number you can achieve?
4. What happens if you adjust the slope of the wedge?
5. How could we improve the mesh?
6. What is the exit Mach number?

OpenFoam: Supersonic sphere

- Historically, CUSF has been more interested in external flows

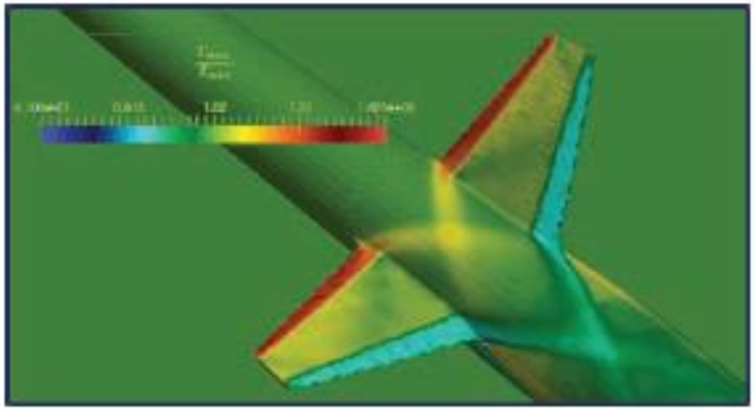
e.g



OpenFoam: Supersonic sphere

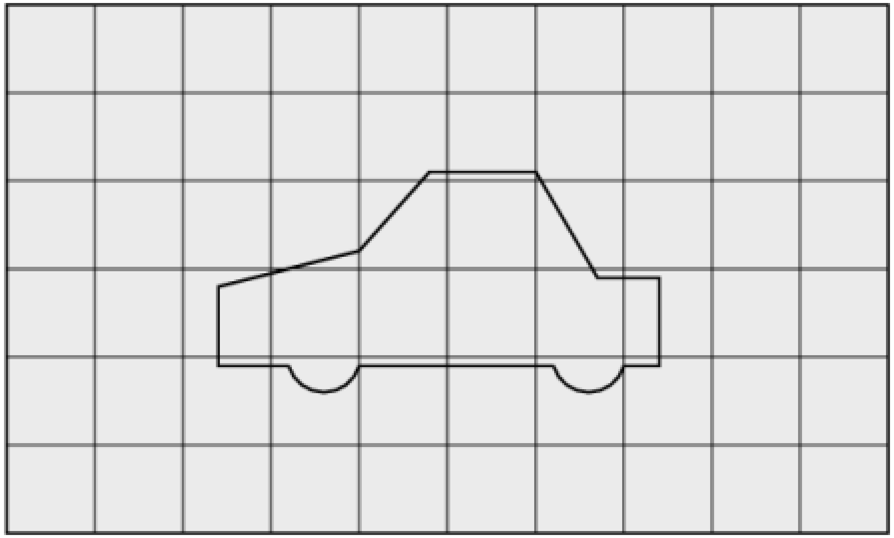
- Historically, CUSF has been more interested in external flows

e.g

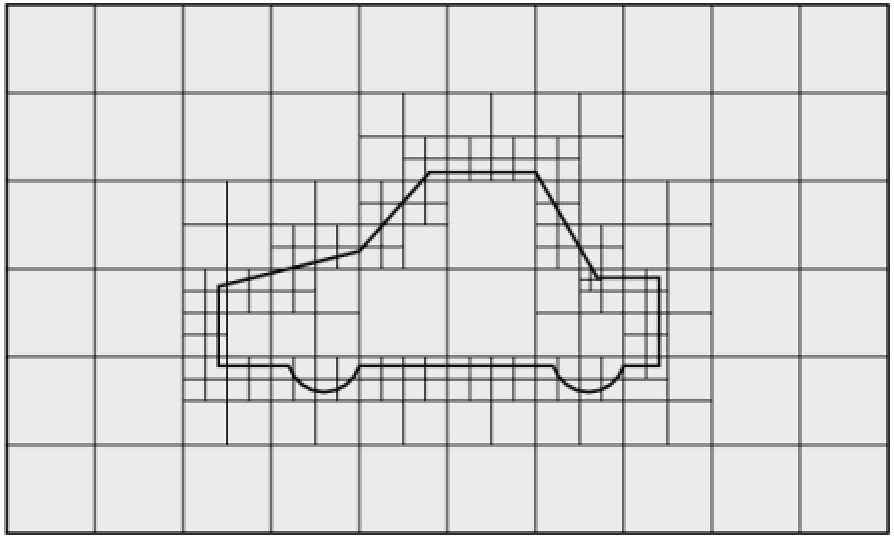


- This requires a more elaborate meshing process: `snappyHexMesh`

- Take a block mesh and an STL file

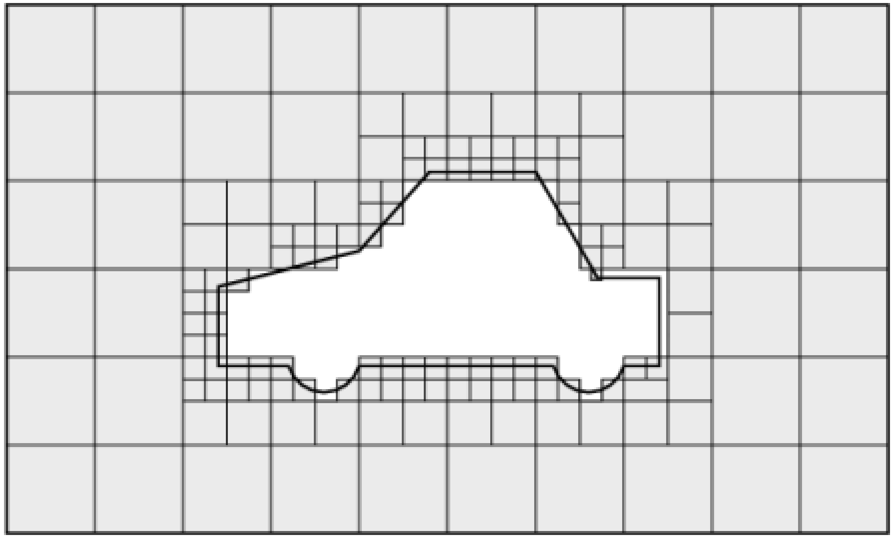


- Take a block mesh and an STL file
- Castellate the mesh



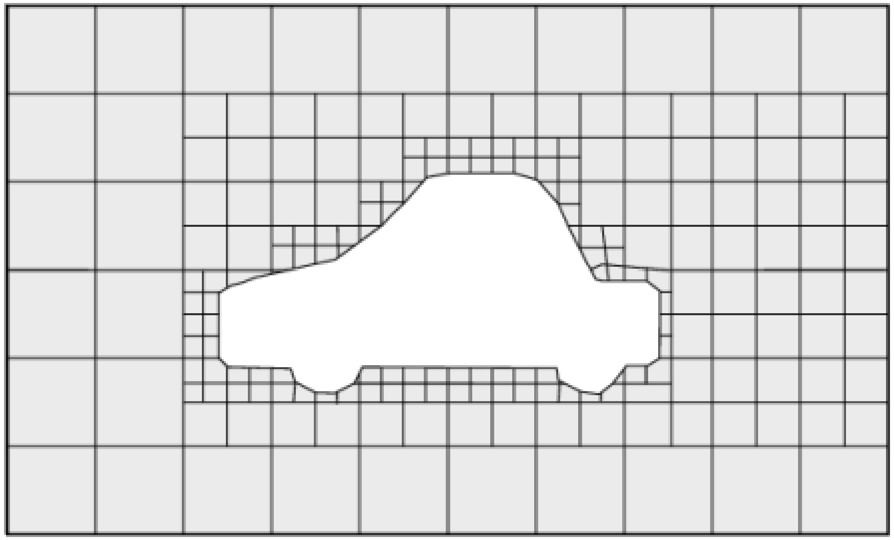
snappyHexMesh

- Take a block mesh and an STL file
- Castellate the mesh
- Remove the interior cells



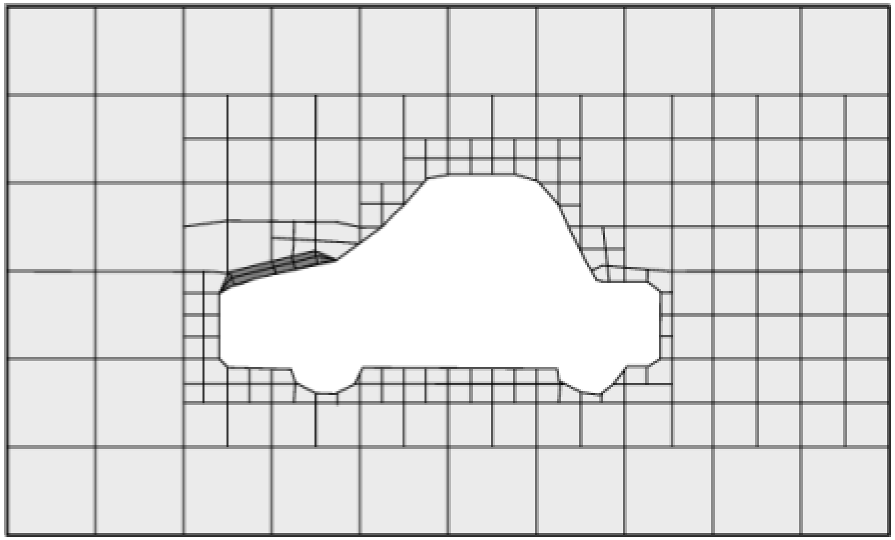
snappyHexMesh

- Take a block mesh and an STL file
- Castellate the mesh
- Remove the interior cells
- ‘Snap’ to the STL surface



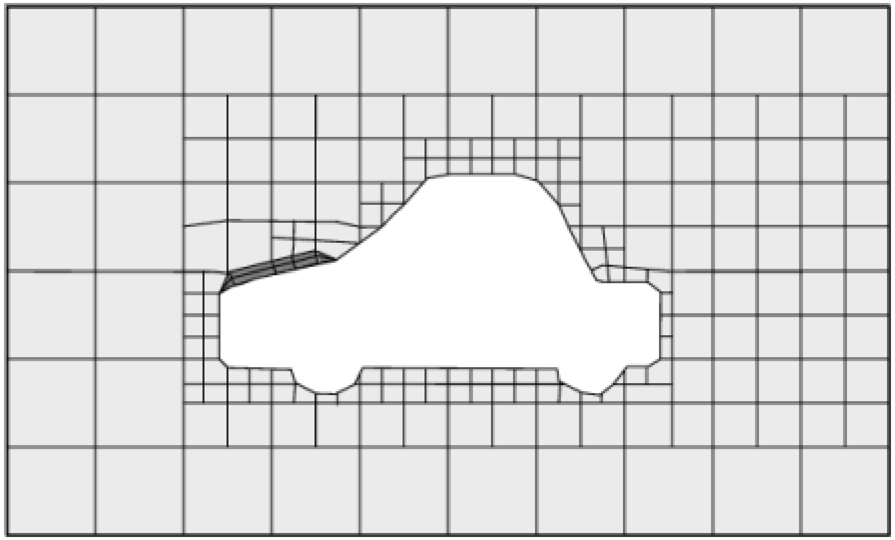
snappyHexMesh

- Take a block mesh and an STL file
- Castellate the mesh
- Remove the interior cells
- ‘Snap’ to the STL surface
- Insert boundary layers



snappyHexMesh

- Take a block mesh and an STL file
- Castellate the mesh
- Remove the interior cells
- ‘Snap’ to the STL surface
- Insert boundary layers



Interactive: `snappyHexMesh`

- Controlled by `snappyHexMeshDict`
- Before running we must:
  - `blockMesh`
  - Prepare the STL: `extractSurfaceFeatures`
  - `snappyHexMesh`
- To look at the mesh: `paraFoam`

- When happy with the mesh: `snappyHexMesh -overwrite`

Interactive: `sonicFoam`

- `sonicFoam` is a bit more robust for this problem.
- The `fvScheme` and `fvSolution` files are different to those of `rhoCentralFoam`
- Run with `sonicFoam`
- Visualise with `paraFoam`

Interactive: Bonus questions

- What do you notice about how long it takes to get a solution?
- What’s happening in the wake of the cylinder?
- If you use rhoCentralFoam (copy fvScheme and fvSolution from the reflection case) what happens?
- What’s the largest timestep you can do with sonicFoam?
- Let sonicFoam run for a long time: do you notice anything about the time per iteration?

The future of CFD will be with Finite Element

- Finite element naturally handles higher order approximations within cells e.g. quadratic, cubic

- Finite elements are well suited for finding **adjoints** which is becoming an import design tool.

How does it work?

- Consider: $\frac{d^2 f}{dx^2} - 1 = 0$ with $f = 0$ on the boundaries of the domain.

How does it work?

- Consider: $\frac{d^2 f}{dx^2} - g(x) = 0$ with $f = 0$ on the boundaries of the domain.
- We can multiply be an arbitrary function $v(x)$: $\frac{d^2 f}{dx^2} v - vg(x) = 0$

How does it work?

- Consider: $\frac{d^2 f}{dx^2} - g(x) = 0$ with $f = 0$ on the boundaries of the domain.
- We can multiply be an arbitrary function $v(x)$: $\frac{d^2 f}{dx^2} v - vg(x) = 0$
- We can then integrate: $\int \frac{d^2 f}{dx^2} v - vg(x)\, dx = 0$

How does it work?

- Consider: $\frac{d^2 f}{dx^2} - g(x) = 0$ with $f = 0$ on the boundaries of the domain.
- We can multiply be an arbitrary function $v(x)$: $\frac{d^2 f}{dx^2} v - vg(x) = 0$
- We can then integrate: $\int \frac{d^2 f}{dx^2} v - vg(x)\, dx = 0$
- We realise: $\frac{d^2 f}{dx^2} v = \frac{d}{dx}\left(\frac{df}{dx} v\right) - \frac{df}{dx}\frac{dv}{dx}$

How does it work?

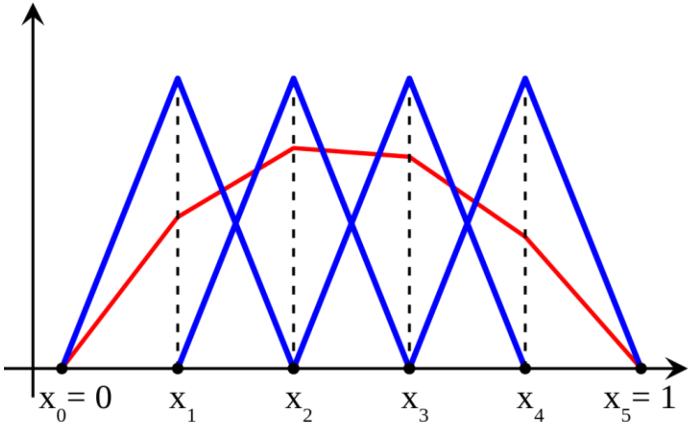
- Consider: $\frac{d^2 f}{dx^2} - g(x) = 0$ with $f = 0$ on the boundaries of the domain.
- We can multiply be an arbitrary function $v(x)$: $\frac{d^2 f}{dx^2} v - vg(x) = 0$
- We can then integrate: $\int \frac{d^2 f}{dx^2} v - vg(x)\, dx = 0$
- We realise: $\frac{d^2 f}{dx^2} v = \frac{d}{dx}\left(\frac{df}{dx} v\right) - \frac{df}{dx}\frac{dv}{dx}$
- The integral becomes: $\int -\frac{df}{dx}\frac{dv}{dx} - vg(x) dx = 0 + b.terms$

How does it work?

- The boundary terms need to be dealt with cleverly but we will ignore them here.

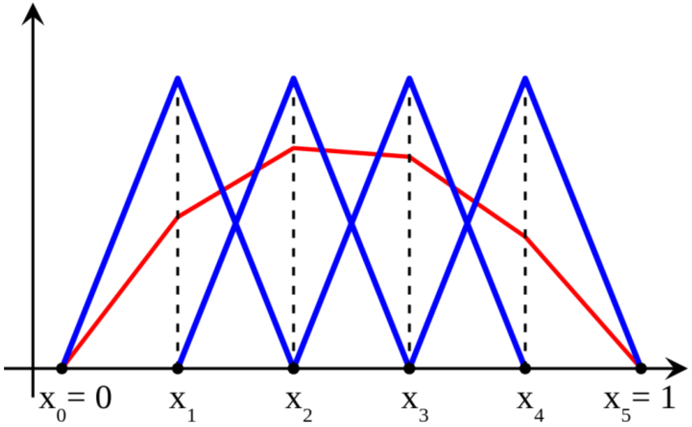
How does it work?

- The boundary terms need to be dealt with cleverly but we will ignore them here.
- We approximate our function using a series of degrees of freedoms and basis polynomials.



How does it work?

- The boundary terms need to be dealt with cleverly but we will ignore them here.
- We approximate our function using a series of degrees of freedoms and basis polynomials.



- $\rightarrow f = \sum F_i \phi_i$ and so $\sum \int -F_i \frac{d\phi_i}{dx}\frac{dv}{dx} - vg(x)dx = 0$

How does it work?

- If f is a solution then $\sum \int -F_i \frac{d\phi_i}{dx}\frac{dv}{dx} - vg(x)dx = 0$ must be true for all possible v

How does it work?

- If f is a solution then $\sum \int -F_i \frac{d\phi_i}{dx}\frac{dv}{dx} - vg(x)dx = 0$ must be true for all possible v
- Therefore we can set $v = \phi_j$ for all j

How does it work?

- If f is a solution then $\sum \int -F_i \frac{d\phi_i}{dx}\frac{dv}{dx} - vg(x)dx = 0$ must be true for all possible v
- Therefore we can set $v = \phi_j$ for all j
- This gives:$\sum_{i,j} \int -F_i \frac{d\phi_i}{dx}\frac{d\phi_j}{dx} - \phi_j g(x)dx = 0$

How does it work?

- If f is a solution then $\sum \int -F_i \frac{d\phi_i}{dx}\frac{dv}{dx} \; - vg(x)dx = 0$ must be true for all possible v
- Therefore we can set $v = \phi_j$ for all j
- This gives:$\sum_{i,j} \int -F_i \frac{d\phi_i}{dx}\frac{d\phi_j}{dx} \; - \phi_j g(x)dx = 0$
- This is a matrix equation for the vector containing all the F_i

Why is this useful?

- For certain conditions, the FE solution is **proven** to be the best solution for a given mesh and set of basis functions.
- We can choose very high order basis functions e.g. cubic (spectral element)
- We've relaxed the smoothness requirements on the solution

Why is this useful?

- For certain conditions, the FE solution is **proven** to be the best solution for a given mesh and set of basis functions.

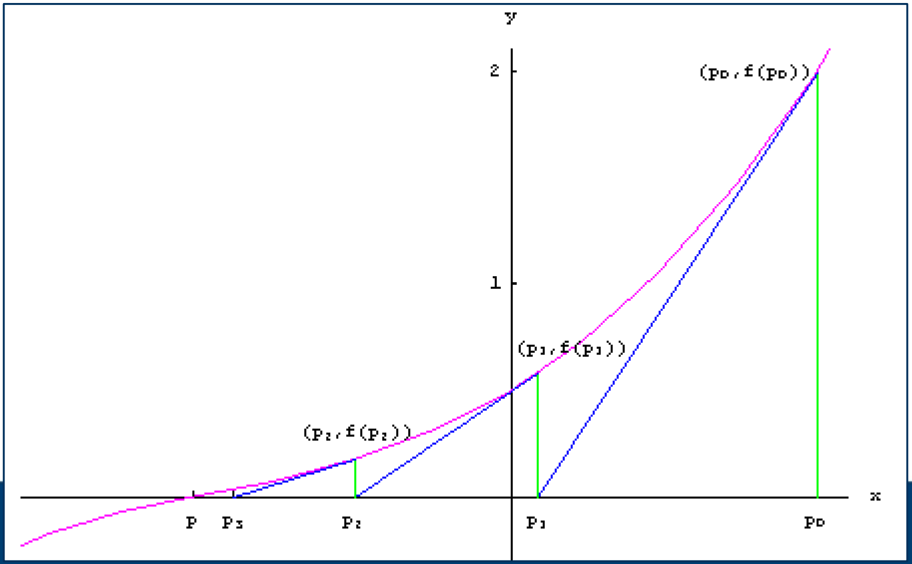
- We can choose very high order basis functions e.g. cubic (spectral element)

FEniCS: a finite element package

- FEniCS provides a nice **flexible** way of solving PDEs by FE

FEniCS: a finite element package

- FEniCS provides a nice **flexible** way of solving PDEs by FE
- We’re going to solve the full Navier-Stokes at low Reynolds number with a Newton-Method



Interactive

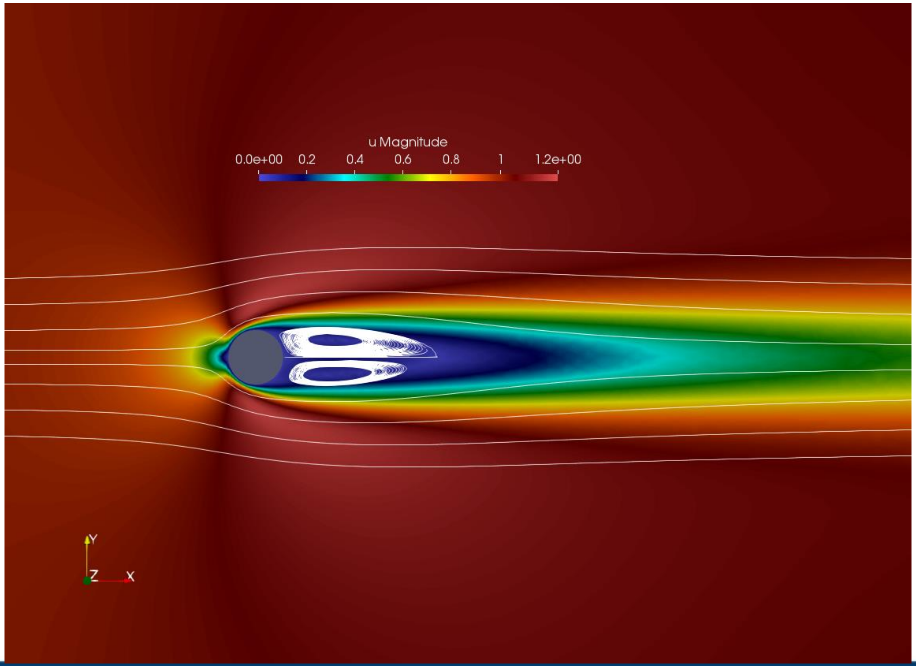
- Run `python FEDemo.py`

- Visualise the flow with `paraview`

Interactive

- Run `python FEDemo.py`

- Visualise the flow with `paraview`



- If we could inject momentum anywhere – where should we do it? How would you find out?

Interactive: Adjoint

Interactive: Adjoint

- $J = -\nabla \boldsymbol{u} . \nabla \boldsymbol{u}$ therefore $\delta J = -2\nabla \boldsymbol{u} . \nabla \delta \boldsymbol{u}$
- The perturbation to the flow satisfies: $\boldsymbol{u} . \nabla \delta \boldsymbol{u} + \delta \boldsymbol{u} . \nabla \boldsymbol{u} + \nabla \delta p - \nu \nabla^2 \delta \boldsymbol{u} = \delta f$
- We multiply by an adjoint state, $\boldsymbol{u}^+$, and apply divergence theorem (integrate by parts)

- Gives $\delta J = \delta f . \boldsymbol{u}^+$ provided $\boldsymbol{u}^+$ is divergence free and satisfies:

$$-\boldsymbol{u} . \nabla \boldsymbol{u}^+ + \boldsymbol{u}^+ . \nabla^T \boldsymbol{u} - \nabla p^+ - \nu \nabla^2 \boldsymbol{u}^+ = +2\nu \nabla^2 \boldsymbol{u}$$

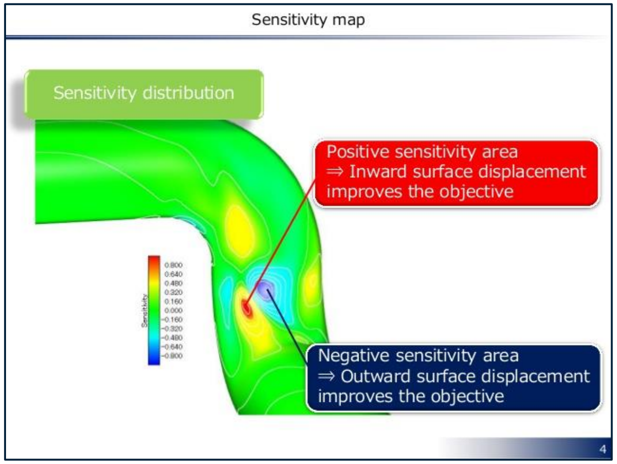
- In finite elements the discrete representation of this is found by transposing the Jacobian

Future

- OpenFoam and Ansys are implementing adjoint solvers at the moment.

- Adjoints are the current big thing in CFD.

- Finite element methods and their close cousin spectral elements are set to be the future of CFD. Higher order polynomials can do a lot more with a smaller number of DOFs



https://www.youtube.com/watch?v=GLe3j0I11_k