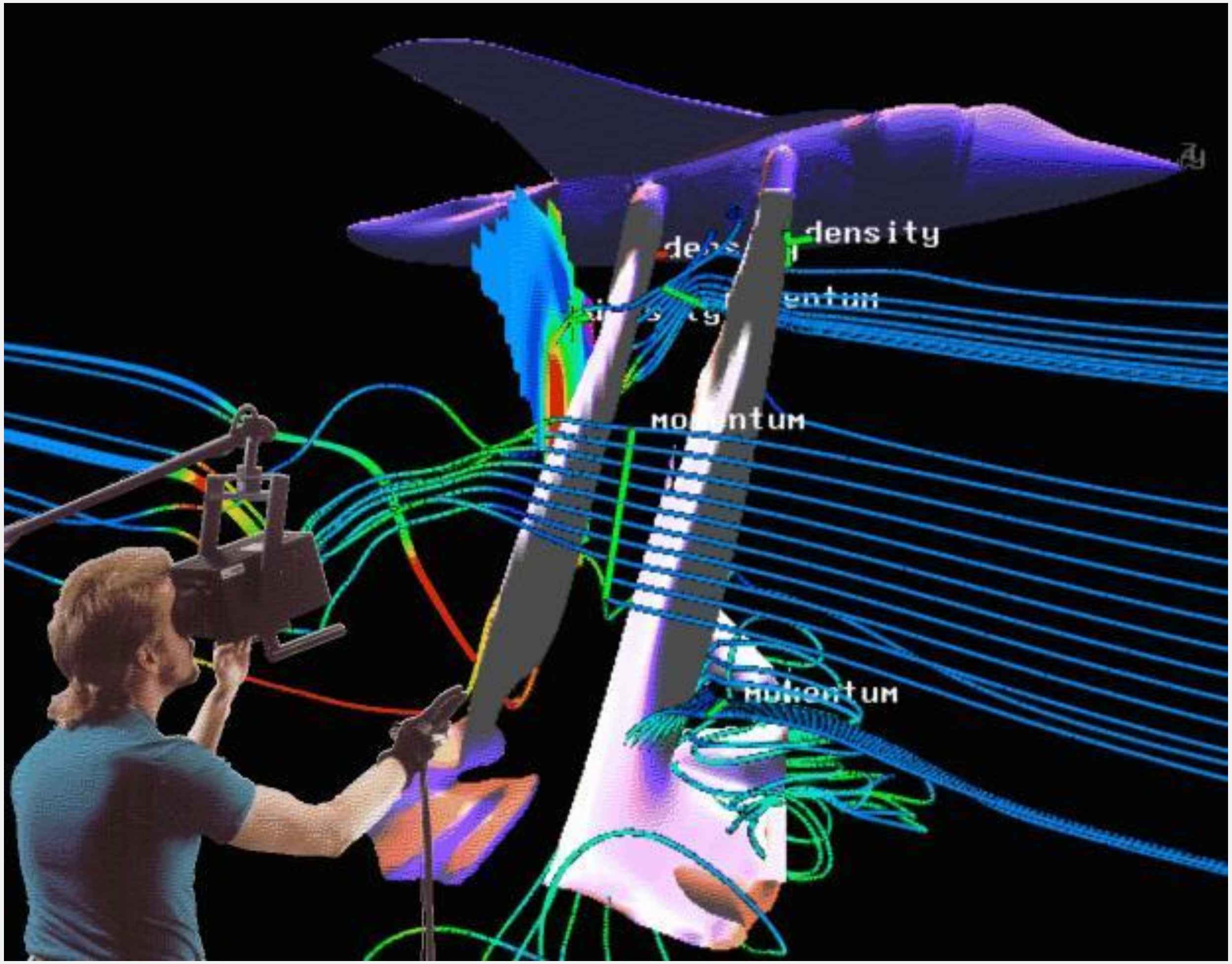
Data Visualization

Vector Field Visualization - Visualizing Flow
Part 1: Experimental Techniques
Particle based Techniques

Applications of Vector Field Visualization

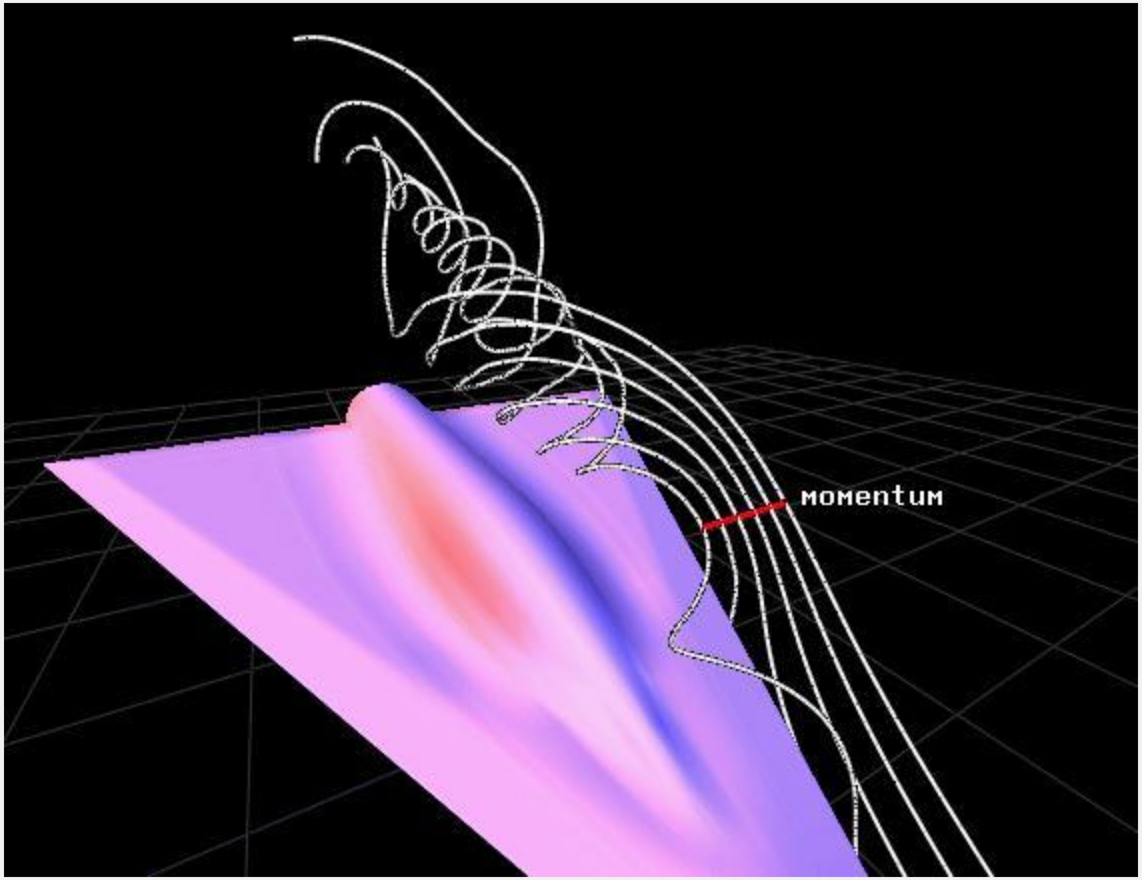
- The major application area is Computational Fluid Dynamics where we wish to visualize the velocity field within a volume, or on a surface
- ... but has applications to any other discipline where **flow** is involved
  - for example, the flow of population in social sciences
- Important distinction between **steady** and **unsteady** (time-dependent) flow

Virtual Windtunnel



The Virtual Windtunnel is a VR facility developed by NASA for aircraft testing

Visualizing Flow over Aircraft Wing



Experimental and Computational Fluid Dynamics

- Experimental fluid dynamics:
  - aim to get impression of flow around a scale model of object (eg smoke in wind tunnel)
  - disadvantages in cost, time and integrity
- Computational fluid dynamics
  - simulation of the flow (Navier-Stokes equations)
  - visualization of the resulting velocity field so as to mimic the experimental techniques

Experimental Flow Visualization - Adding Foreign Material

- Time lines
  - row of small particles (hydrogen bubbles) released at right angles to flow - motion of 'line' shows the fluid flow
- Streak line
  - dye injected from fixed position for period of time - tracer of dye shows the fluid flow
- Path line
  - small particles (magnesium powder in liquid; oil drops in gas) - velocity measured by photographing motion with known exposure time

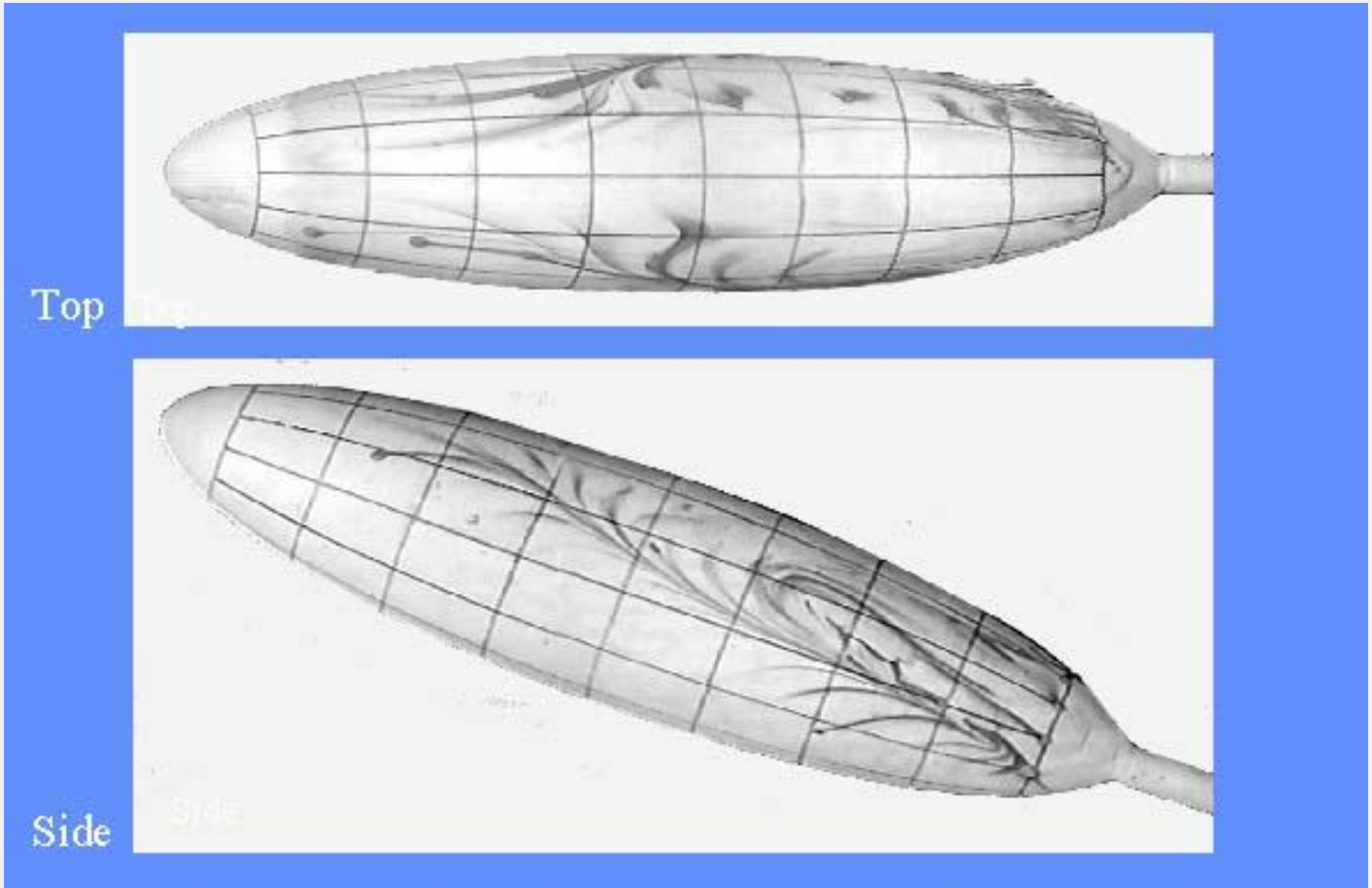
Experimental Flow Visualization - Other Techniques

- Visualization of flow field on surface of object achieved by fixing tufts at several points on surface - orientation of threads indicates direction of flow

Notice distinction:

- tufts show flow past a fixed point (Eulerian)
- bubbles etc show the flow from point of view of a floating object (Lagrangian)

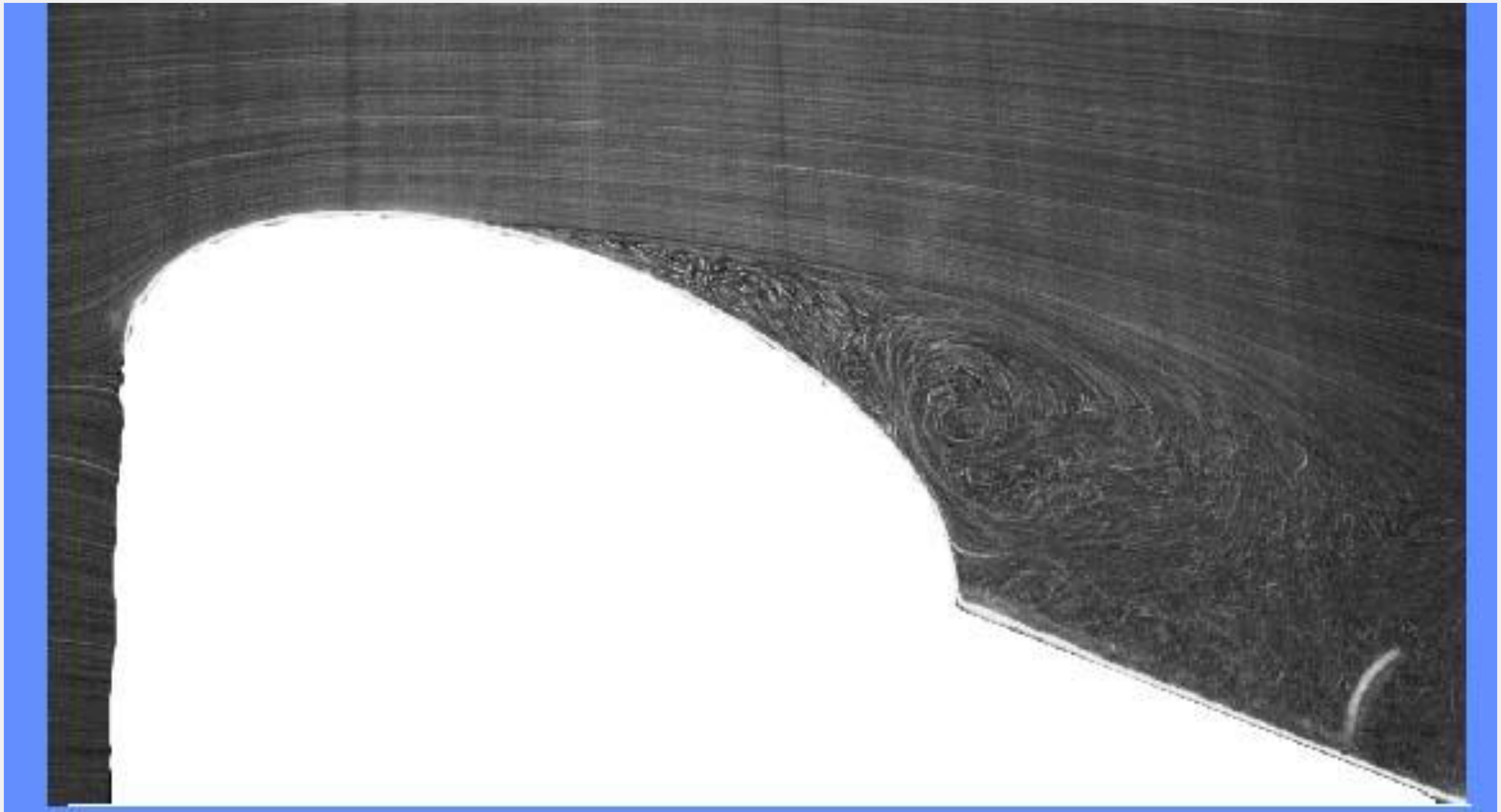
Experimental Visualization Example - Poster Paint in Water



Experimental surface flow pattern for 1-4 model

see www.flometrics.com

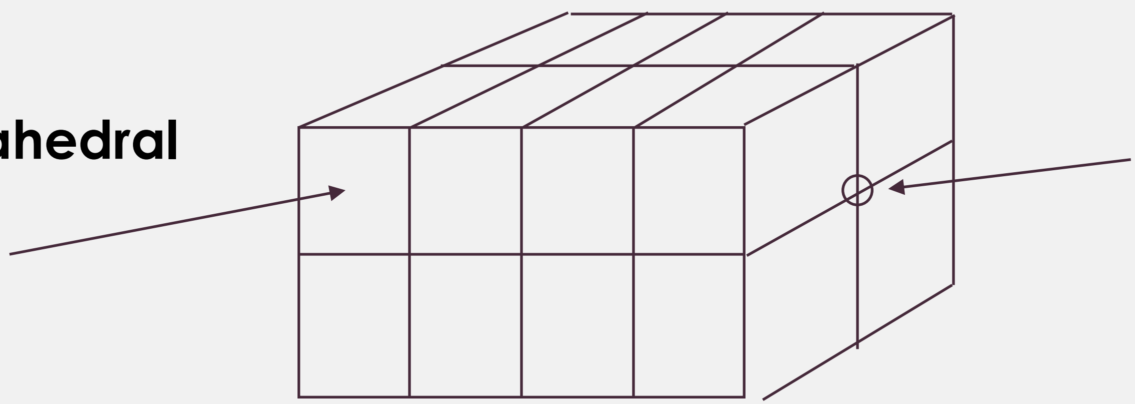
Experimental Visualization - Particles Illuminated by Laser Sheet Light



Experimental sheet light particle paths in symmetry plane for 1-2 model

Computer Flow Visualization

- Now look at methods for computer aided flow visualization
- Assume initially velocity field given on 3D Cartesian grid

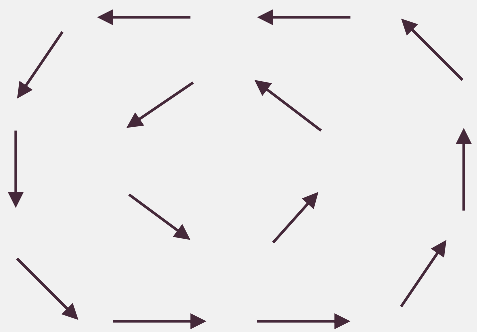


Using Scalar Techniques

- Sometimes it is useful to derive scalar quantities from the velocity field
- For example, velocity magnitude
  - speed = sqrt $(v_x^2 + v_y^2 + v_z^2)$
- How would these be visualized?

Arrows

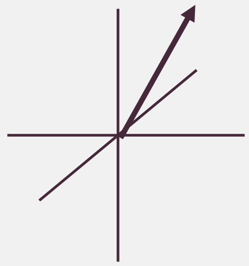
- Very simple technique
- Arrow drawn at each grid point showing direction and size of velocity



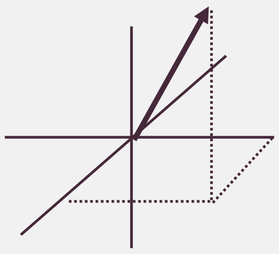
This works effectively enough in 2D

Arrows

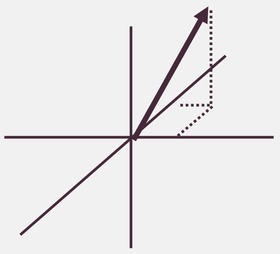
- But in 3D it suffers from perception problems:



Is it this?



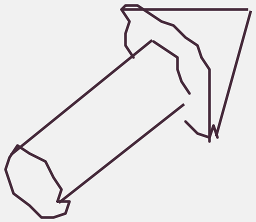
or this?



Of course the picture quickly gets cluttered too

Arrows

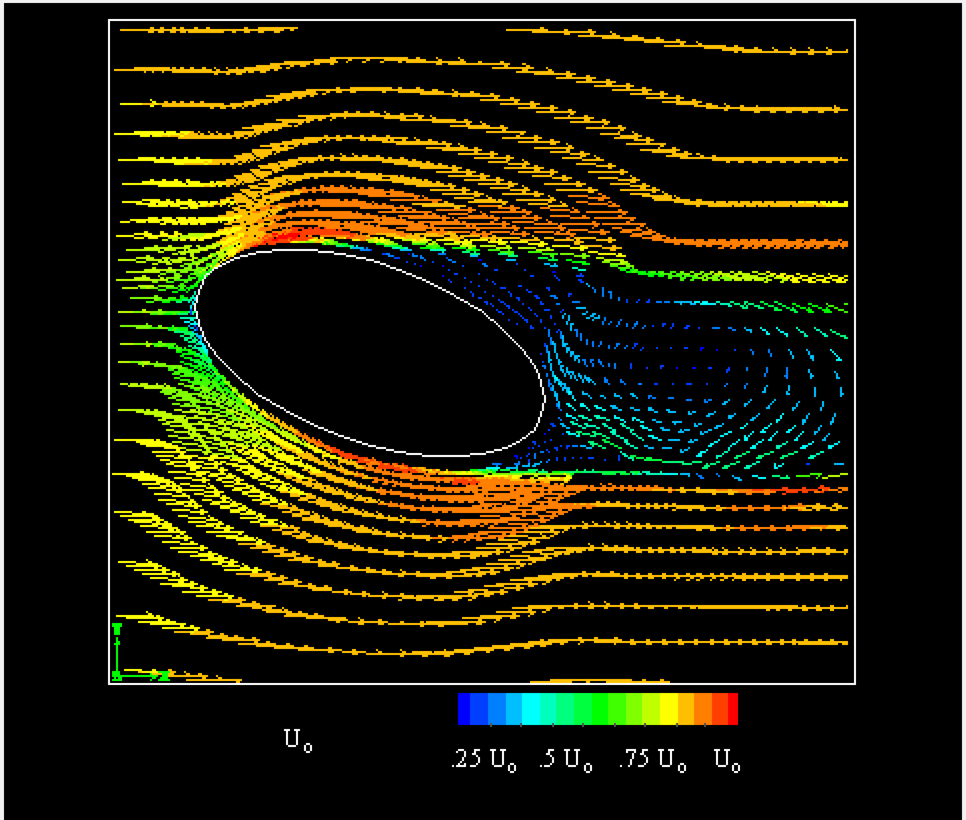
- Arrows can be used successfully in 3D as follows:
  - by slicing the volume, and attaching arrows (with shadow effects) to the slice plane - this gives a hedgehog effect
  - by giving more spatial cues - drawing arrows as true 3D objects



but clutter again a problem!

{BTW - Eulerian or Lagrangian?}

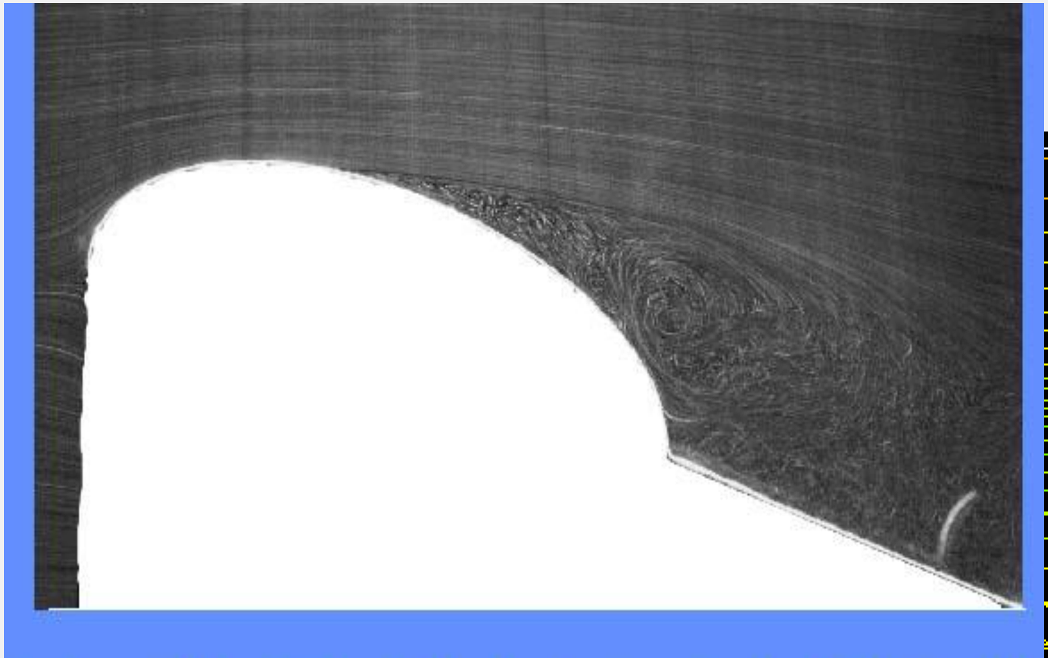
CFD simulation of laser example



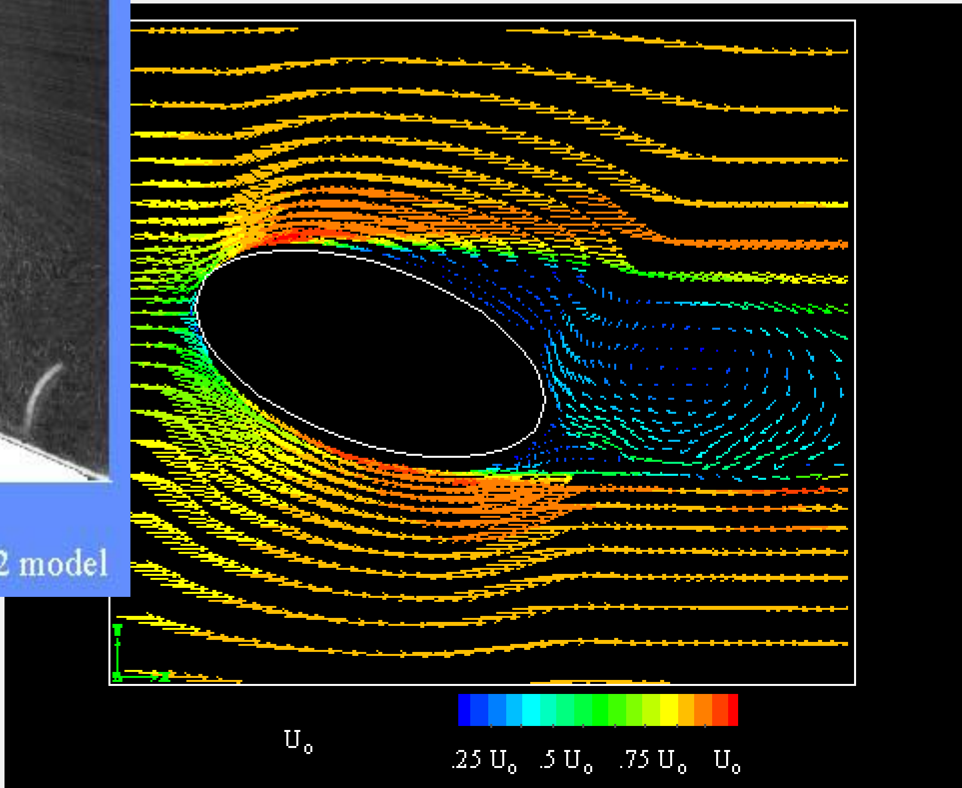
Computed symmetry plane velocity vectors for 1-2 model

Flometrics - see www.flometrics.com

Comparison of Experimental and Computational Visualization

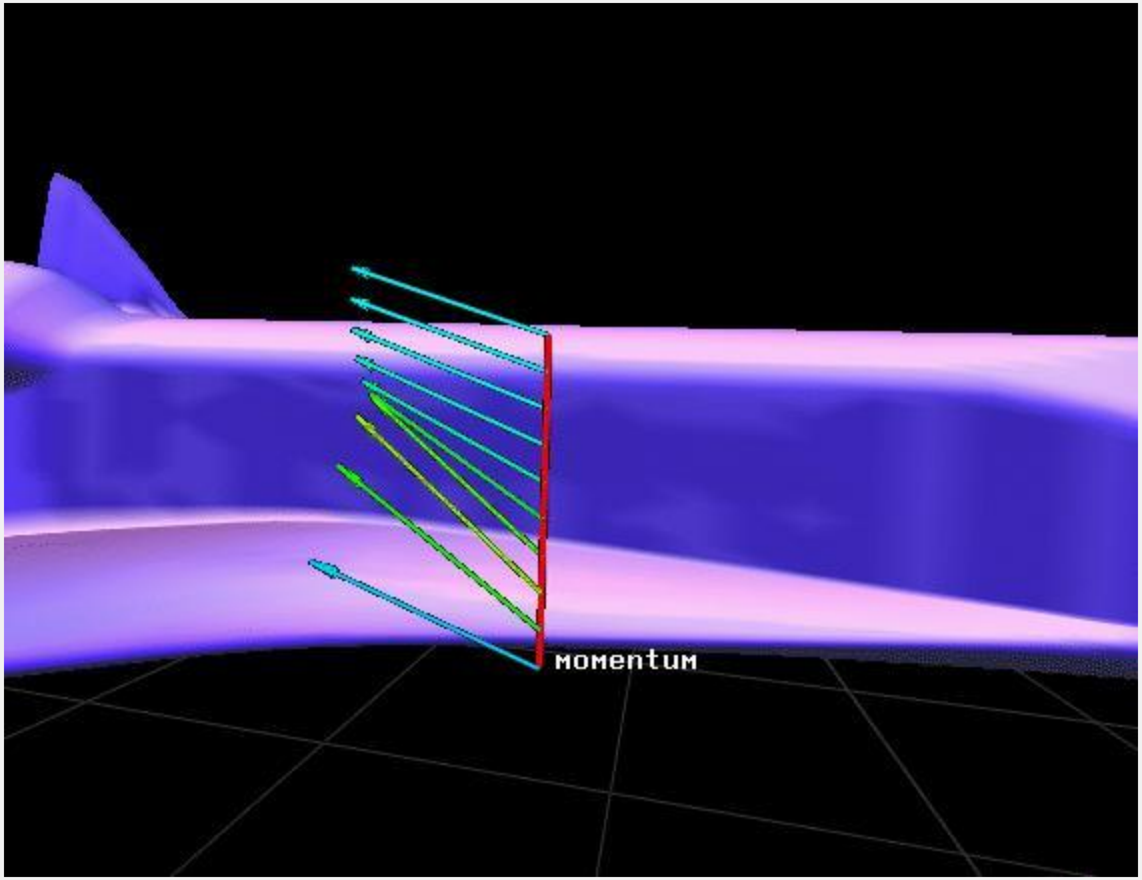


Experimental sheet light particle paths in symmetry plane for 1-2 model



Computed symmetry plane velocity vectors for 1-2 model

Tufts



Particle Traces

- This is analogous to experimental path lines - we imagine following the path of a weightless particle - cf a bubble
- Suppose initial position - *seed point* - is

$$(x_0, y_0, z_0)$$

- The aim is to find how the path

$$(x(t), y(t), z(t))$$

develops over time
- Also called *particle advection*

Particle Traces

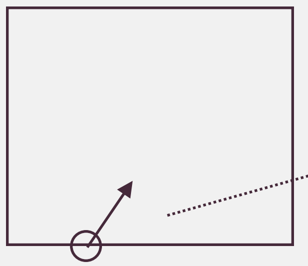
- Motion of a particle is given by:

$$dx/dt = v_x;\ dy/dt = v_y;\ dz/dt = v_z$$

- three ordinary differential equations with initial conditions at time zero:

$$x(0) = x_0;\ y(0) = y_0;\ z(0) = z_0$$

In 2D, we have:



(v_x,v_y) is rate of change of position

(x_0,y_0)

Particle Tracing - Numerical Techniques for Integrating the ODEs

- Simplest technique is Euler's method:

$$dx/dt = (\ x(t+\Delta t) - x(t))\ /\ \Delta t = v_x(\underline{p}(t))$$

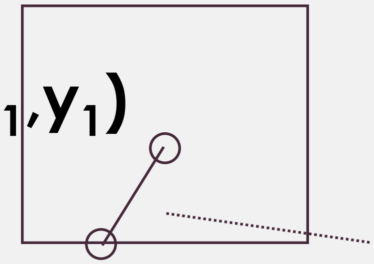
$\underline{p}=(x,y,z)$

hence

$$x(t+\Delta t) = x(t) + \Delta t . v_x(\underline{p}(t))$$

- Similarly, for y(t) and z(t)

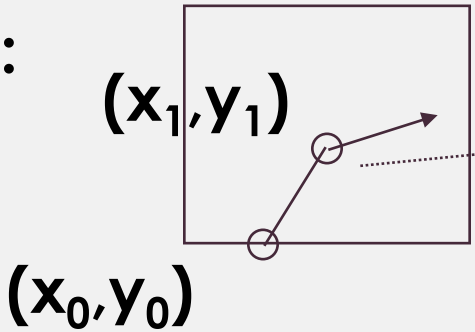
In 2D, we have:



Particle Tracing - Interpolation

- As the solution proceeds, we need velocity values at interior points
- (v_x, v_y, v_z) is calculated at current point (x,y,z) - by trilinear interpolation for example.

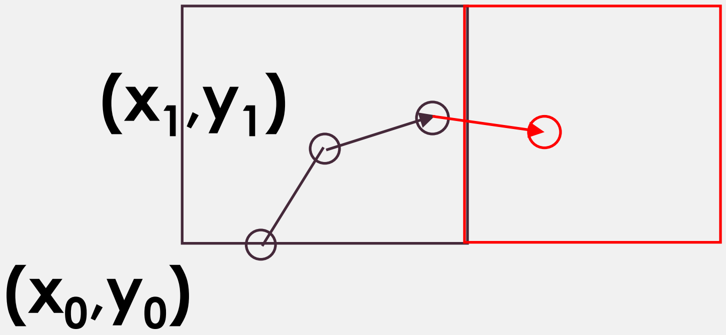
In 2D, we have:



(v_x,v_y) found by interpolation (bilinear)

Particle Tracing - Point Location

- **When we leave one cell, we need to determine which cell the new point belongs to**



-this is quite straightforward for Cartesian grids

Particle Tracing - Algorithm

```
find cell containing initial position
                                        {point location}
while particle in grid
            determine velocity at current position
                                         {interpolation}
            calculate new position
                                           {integration}
            find cell containing new position
                                        {point location}
endwhile
```

Improving the Integration

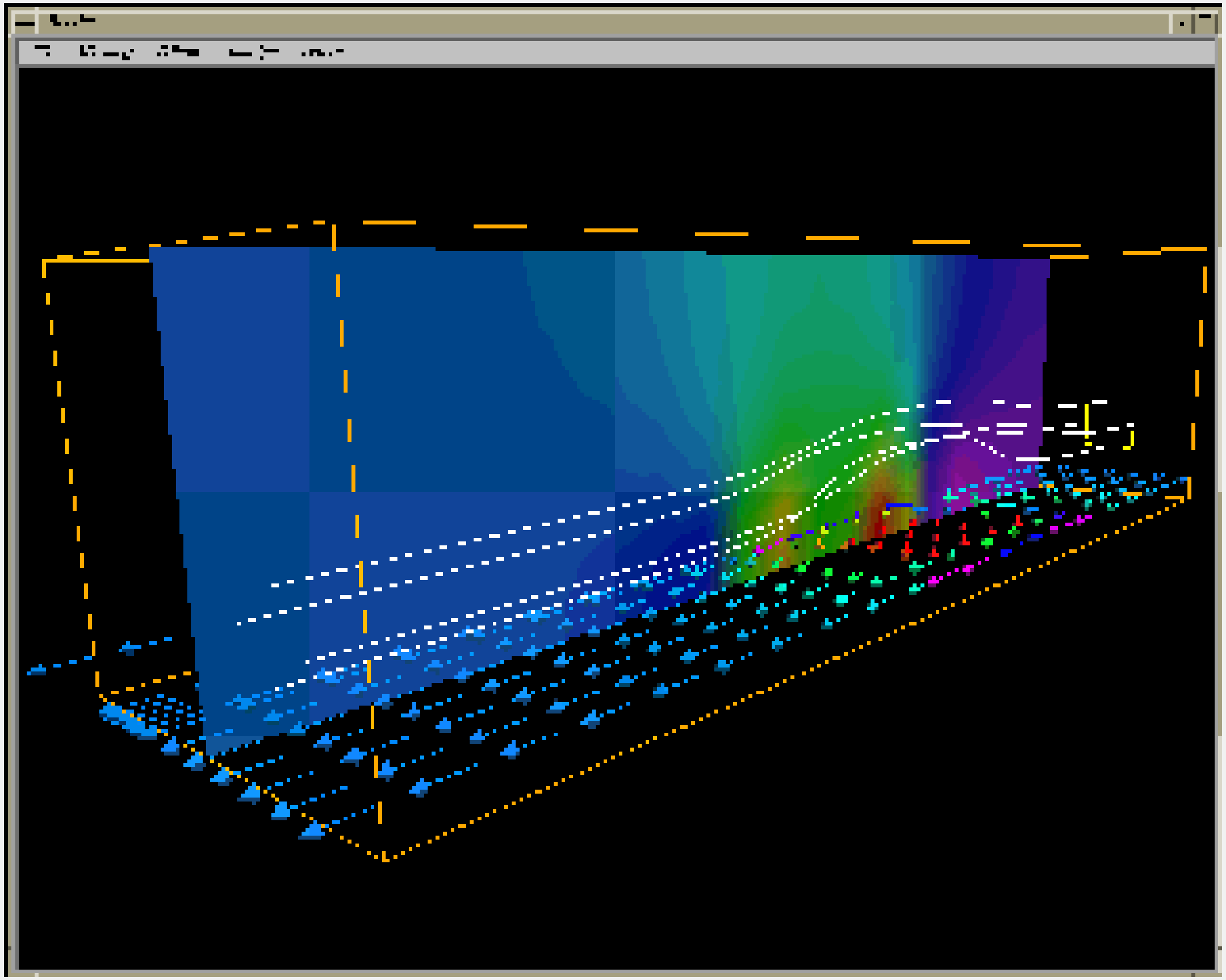
- Euler's method is inaccurate (unless the step size Δt is very small)
- Better is Runge-Kutta:
  - $x^* = x(t) + \Delta t.v_x(\underline{p}(t))$ (and for y*, z*)
  - $x(t+\Delta t) = x(t) + \Delta t.\{ v_x(\underline{p}(t)) + v_x(\underline{p}^*)\}/2$ (and for y,z)
- This is Runge-Kutta 2nd order - there is also a more accurate 4th order method
- There is another source of error in particle tracing
  - what?

Rendering the Particles - and Rakes

- Particles may be rendered as
  - points - but are there better representations?

- It is common to use a *rake* of seed points, rather than just one - rake can be line, circle, even an area...

Particle Advection Example - Flow Around a Moving Car



Created using IRIS Explorer

Streak Lines and Time Lines

- Streak lines
  - release continuous flow of particles for a short period
- Time lines
  - release a line of particles at same instant and draw a curve through the positions at successive time intervals

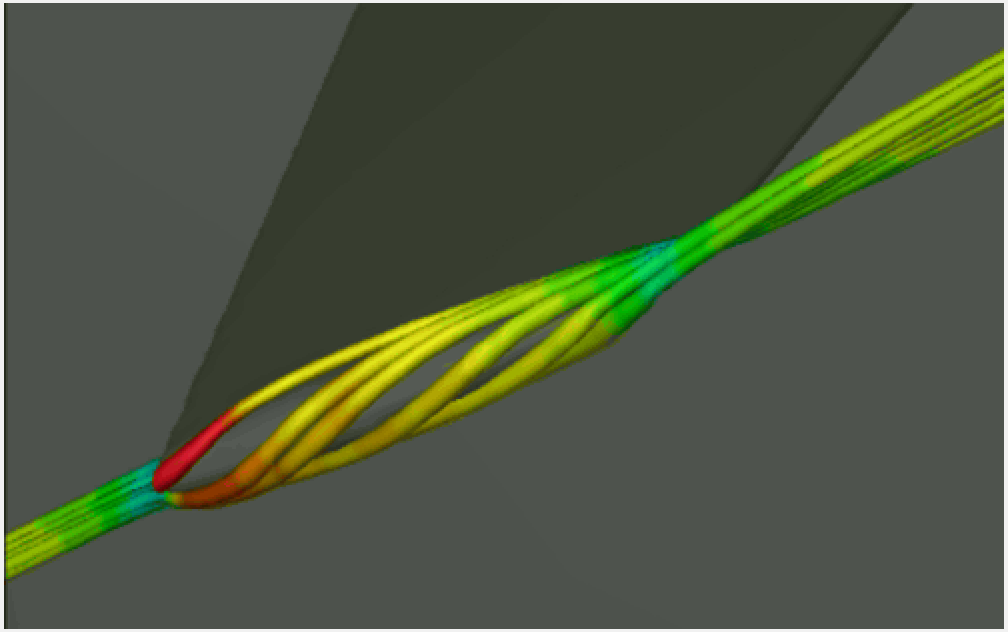
Stream Lines

- Mathematically, stream lines are lines everywhere tangential to the flow
- For a steady flow - what is the relation between stream lines and particle traces?

Rendering Streamlines

- In 3D, curves are hard to understand without depth cues
- Ideas used include:
  - stream ribbons - each streamline drawn as a thin flat ribbon, showing twist; or two adjacent streamlines connected into ribbon, showing twist and divergence
  - tubes

Streamlines Example



Streamlines drawn as tubes - by K Ma of ICASE (see www.icase.edu)

Steady Flow Visualization

- Streamlines and stream ribbons best for flow direction
- Particle traces best for flow speed
- Note that derived quantities are also visualized:
  - flow speed as 3D scalar field
  - vorticity field as 3D vector field
  - vorticity magnitude as 3D scalar field

  (Vortex = rotational flow about axis

  vorticity = vector product of velocity and its gradient)

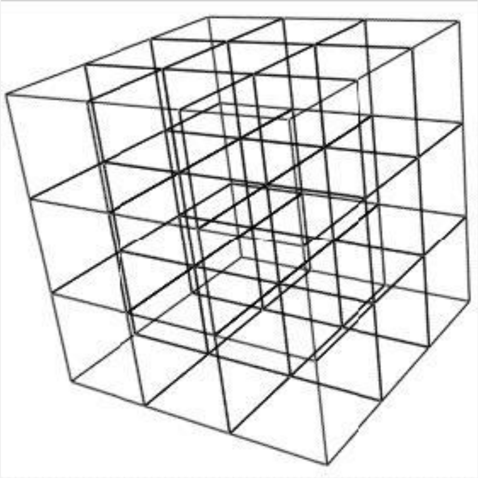
Unsteady Flow Visualization

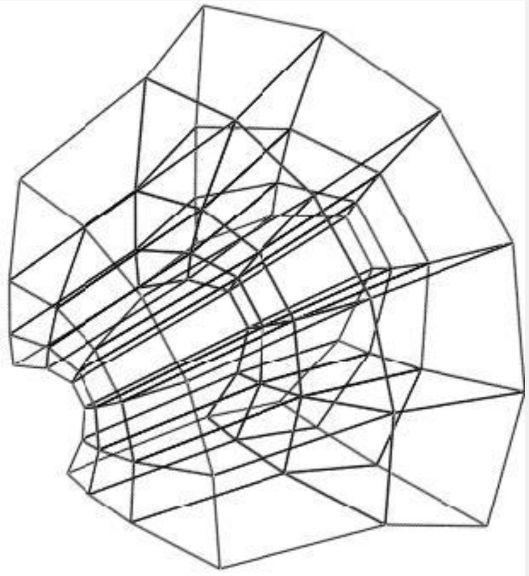
- Recent research interest has been in the more complex case of unsteady flows, where velocity depends on time
- Particle traces, streak lines and time lines can all be used
- Streak lines seem to give the best results
- Nice applet at:

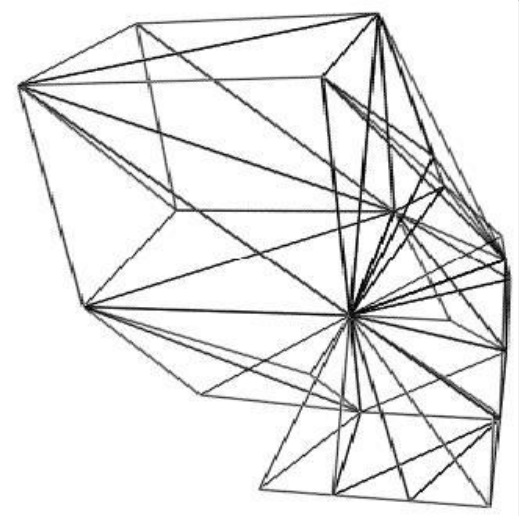
http://widget.ecn.purdue.edu/~meapplet/java/flowvis/Index.html

Different Types of Grid

- Rectilinear
- Curvilinear
- Unstructured







Curvilinear Grids

- Point location and interpolation are much harder than for Cartesian grids
  - a solution is to decompose each hexahedral cell into tetrahedra
  - point ‘inside’ test then easier...
  - … and interpolation is linear
- Point location
  - draw line to new point
  - calculate intersection with faces to determine adjacent tetrahedron
  - check whether point inside new tetrahedron