CS 519: Scientific Visualization

Vector Field Visualization
Stream Objects

Eric Shaffer

Some slides adapted Alexandru Telea, *Data Visualization Principles and Practice*

Flow Visualization Applications

- Gaseous flow:
  - development of cars, aircraft, spacecraft
  - design of machines - turbines, combustion engines
- Liquid Flow:
  - naval applications - ship design
  - civil engineering - harbor design, coastal protection
- Chemistry - fluid flow in reactor tanks
- Medicine - blood vessels, SPECT, fMRI

Flow Visualization Taxonomy

- Methods can be classified as being
  - Direct
    - Visualizing vector field
  - Integration-based
    - Long-term behavior (eg streamlines)
  - Derived
    - Field topology

Flow Visualization

- Flows have different spatial and temporal characteristics
  - 2D versus 3D
  - Flow on surfaces
  - Steady versus unsteady flows
- Today, we will look at integration-based techniques
- Important issues for these techniques include
  - Seeding
  - Accuracy
  - Large Data

Flow Visualization

- Given (typically):
  - physical **position** (**vector**)
  - pressure (scalar),
  - density (scalar),
  - **velocity** (**vector**),
  - entropy (scalar)
- steady flow - vector field stays constant
- unsteady - vector field changes with time

Quick Introduction to Fluids

- Basic properties of fluids
  - Pressure
  - Density
  - Viscosity
  - Surface tension
- Different types of fluids:
  - Incompressible (divergence-free) fluids:
    Fluids doesn't change volume (very much).
  - Compressible fluids:
    Fluids change their volume significantly.
  - Viscous fluids:
    Fluids tend to resist deformation

Fluids: Quick Terminology Guid

- Inviscid (Ideal) fluids:
  No viscosity
- Turbulent flow:
  Appears to have chaotic and random changes
- Laminar (streamline) flow
  Flow that has smooth behavior
- Newtonian fluids:
  Fluids continue to flow, regardless of force acting on it

Fluids

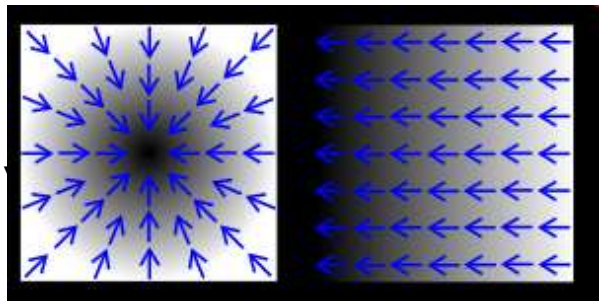
- Non-Newtonian fluids:
  Fluids that have non-constant viscosity
  - Popular EOH demo
  - Mythbusters YouTube….

Vector Calculus Review

- Gradient (∇): A vector pointing in the direction of the greatest rate of increment

$$\nabla u = \left(\frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}, \frac{\partial u}{\partial z} \right)$$

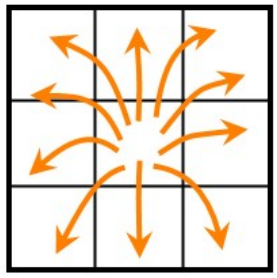
u can be a scalar or a vector-valued function



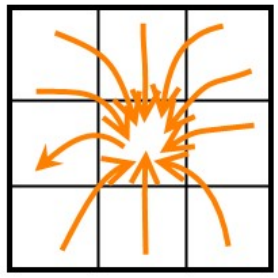
- Divergence ($\nabla \cdot$): Measure how the vectors are converging or diverging at a given location (volume density of the outward flux)

$$\nabla \cdot \boldsymbol{u} = \frac{\partial \boldsymbol{u}}{\partial x} + \frac{\partial \boldsymbol{u}}{\partial y} + \frac{\partial \boldsymbol{u}}{\partial z}$$

u can only be a vector-valued function



Source, Div(**u**) > 0



Sink, Div(**u**) < 0

Vector Calculus Review

- Laplacian (Δ or ∇^2): Divergence of the gradient

$$\nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}$$

u can be a scalar or a vector

Curl (Vorticity) Revisited

- In 3D we have: $\operatorname{rot}\,\mathbf{v} = \left(\frac{\partial v_z}{\partial y} - \frac{\partial v_y}{\partial z}, \frac{\partial v_x}{\partial z} - \frac{\partial v_z}{\partial x}, \frac{\partial v_y}{\partial x} - \frac{\partial v_x}{\partial y} \right)$
- In 2D that degenerates to $\nabla \times v = \left(0, 0, \frac{\partial v_y}{\partial x} - \frac{\partial v_x}{\partial y} \right)$
- And the magnitude would just be the absolute value of the non-zero entry…

Navier Stokes Equations

$$\underbrace{\rho\left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u}\right)}_{1} = \underbrace{-\nabla p}_{2} + \underbrace{\nabla\cdot\left(\mu(\nabla\mathbf{u} + (\nabla\mathbf{u})^T) - \frac{2}{3}\mu(\nabla\cdot\mathbf{u})\mathbf{I}\right)}_{3} + \underbrace{\mathbf{F}}_{4}$$

The different terms correspond to the inertial forces (1), pressure forces (2), viscous forces (3), and the external forces applied to the fluid (4).

- Describe how velocity, pressure, temperature, density of moving fluid are related.
- The equations were derived independently by G.G. Stokes, in England, and M. Navier, in France, in the early 1800's.
- The equations are a set of coupled differential equations and could, in theory, be solved for a given flow problem by using methods from calculus. But, in practice, these equations are too difficult to solve analytically.

Navier-Stokes Equations (Simplified)

- Basically tell you how a vector field will change
  - So…it's sort of describing the acceleration
- If you solve them, you have a closed form formula
  - …that for any time t will you what the field is
- Euler's equations for fluids were simpler
  - ...maybe easier to understand

$$\frac{\partial u_x}{\partial t} + u_x\frac{\partial u_x}{\partial x} + u_y\frac{\partial u_x}{\partial y} + u_z\frac{\partial u_x}{\partial z} = f_x(x, y, z, t) - \frac{\partial p}{\partial x} \quad (2)$$

$$\frac{\partial u_y}{\partial t} + u_x\frac{\partial u_y}{\partial x} + u_y\frac{\partial u_y}{\partial y} + u_z\frac{\partial u_y}{\partial z} = f_y(x, y, z, t) - \frac{\partial p}{\partial y} \quad (3)$$

$$\frac{\partial u_z}{\partial t} + u_x\frac{\partial u_z}{\partial x} + u_y\frac{\partial u_z}{\partial y} + u_z\frac{\partial u_z}{\partial z} = f_z(x, y, z, t) - \frac{\partial p}{\partial z} \quad (4)$$

Navier-Stokes Equations

- "…it has not yet been proven that in three dimensions solutions always exist (existence), or that if they do exist, then they do not contain any singularity (they are smooth). These are called the Navier–Stokes existence and smoothness problems. The Clay Mathematics Institute has called this one of the seven most important open problems in mathematics and has offered a US$1,000,000 prize for a solution or a counter-example." --Wikipedia

Stream Objects (Integration-based Techniques)

Main idea

- think of the vector field $\mathbf{v} : D$ as a flow field
- choose some 'seed' points $s \in D$
- move the seed points s in $\mathbf{v}$
- show the trajectories

Stream lines

- assume that $\mathbf{v}$ is not changing in time (stationary field)
- for each seed $p_o \in D$
  - the streamline S seeded at p_o is given by

$$S = \{p(\tau), \tau \in [0, T]\}, p(\tau) = \int_{t=0}^{\tau} \mathbf{v}(p)dt, \quad \text{where } p(0) = p_0$$

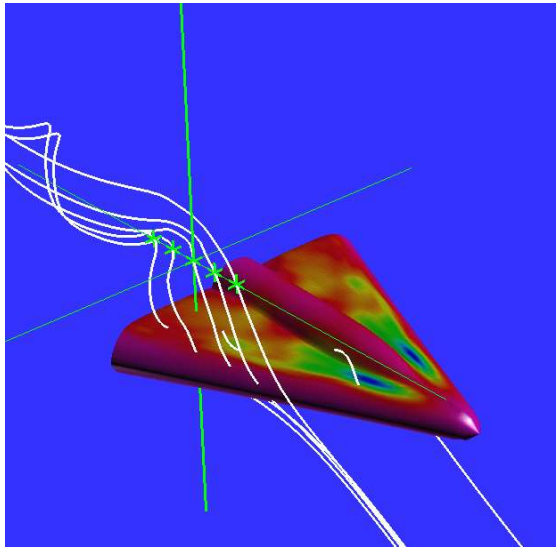
integrate p_o in vector field $\mathbf{v}$ for time T

- if v is time dependent $\mathbf{v}=\mathbf{v}(t)$, streamlines are called particle traces

Stream Objects (Integration-based Techniques)

- Vector glyph plots show the **trajectories over a short time** of trace particles released in the vector fields

- Stream objects show the **trajectories for longer time intervals** for a given vector field



Stream Objects

Practical construction

- numerically integrate

$$S = \{p(\tau), \tau \in [0, T]\}, p(\tau) = \int_{t=0}^{\tau} \mathbf{v}(p)dt, \quad \text{where } p(0) = p_0$$

- discretizing time yields

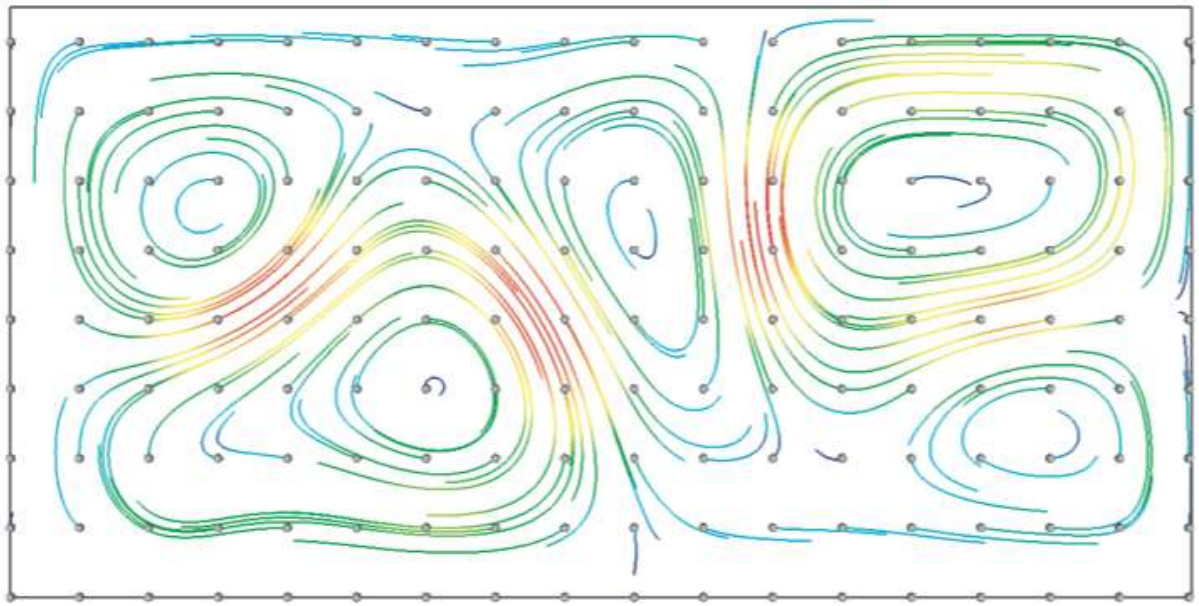
$$\int_{t=0}^{\tau} \mathbf{v}(p)dt = \sum_{i=0}^{\tau/\Delta t} \mathbf{v}(p_i)\Delta t \quad \text{where } p_i = p_{i-1} + \mathbf{v}_{i-1}\Delta t$$

(simple Euler integration)

- Euler integration explained
  - we consider $\mathbf{v}$ constant between two sample points p_i and p_{i+1}
  - variant: use $\mathbf{v}(p)/||\mathbf{v}(p)||$ instead of $\mathbf{v}(p)$ in integral
  - S will be a polyline, $S = \{p_i\}$
- stop when $\tau = T$ or $\mathbf{v}(p) = 0$ or $p \notin D$

Details: See book, Listing 6.1

Stream Objects



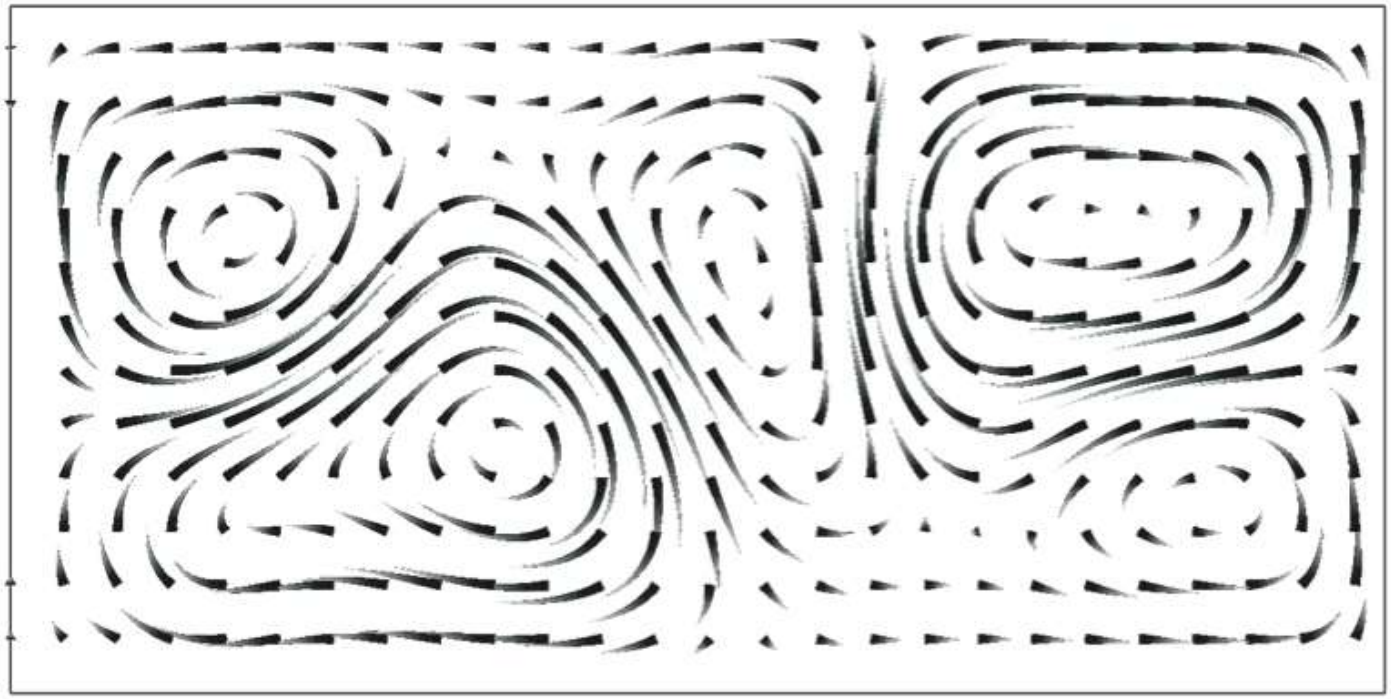
streamlines: seeds from regular grid; use un-normalized $\mathbf{v}$ for integration; color by $\|\mathbf{v}\|$

Why is this better than vector glyphs?

Stream Tubes

Variations

- modulate tube thickness by
  - data (hyperstreamlines)
  - integration time – we obtain nice tapered arrows

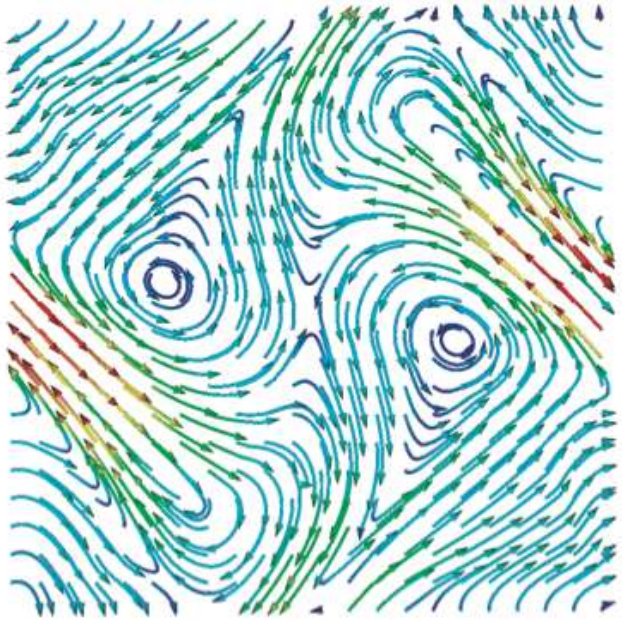


stream tubes – radius *and* opacity decrease with integration time

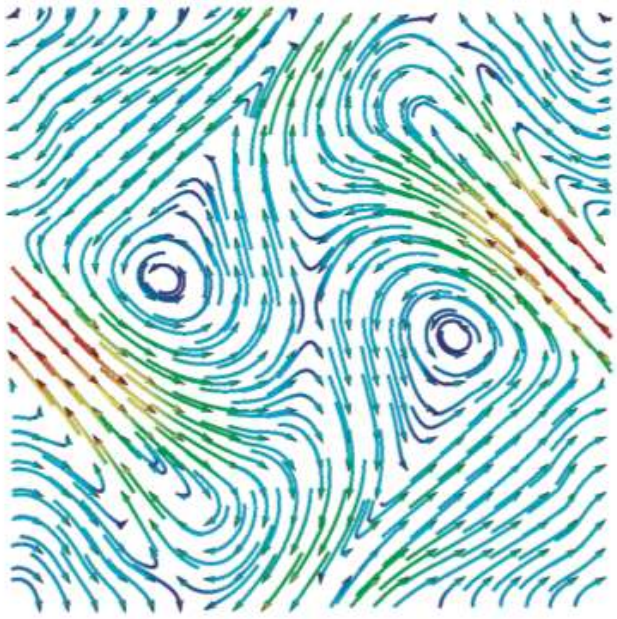
Stream Tubes

Like stream objects, but 3D

- compute 1D stream objects (e.g. streamlines)
- sweep (circular) cross-section along these
- visualize result with shading



stream tubes, forward integration

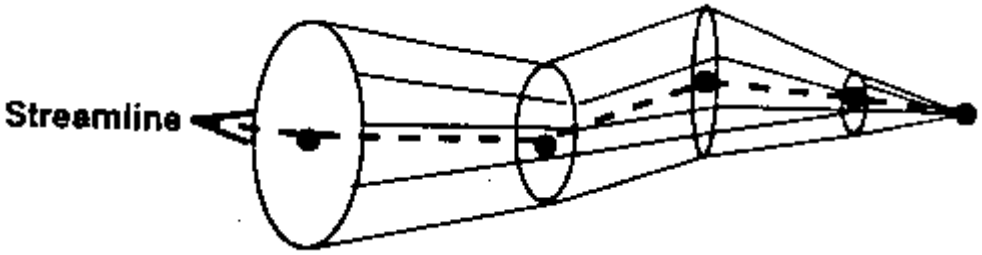


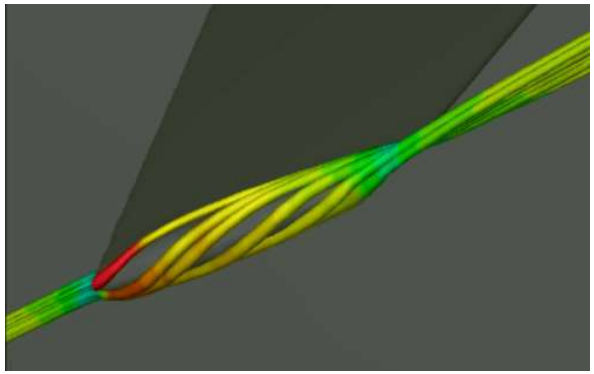
stream tubes, backward integration

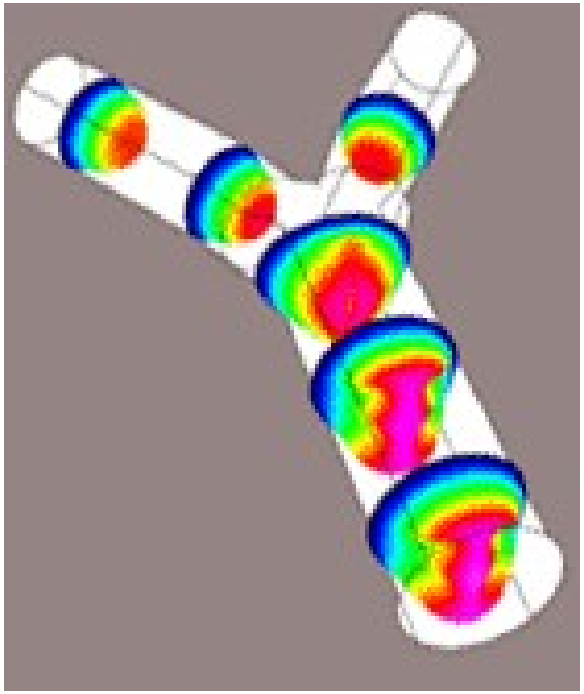
- in 2D they are a nicer option than hedgehog/glyph plots

Stream Tube

- Generate a stream-line and connect circular crossflow sections along the stream-line



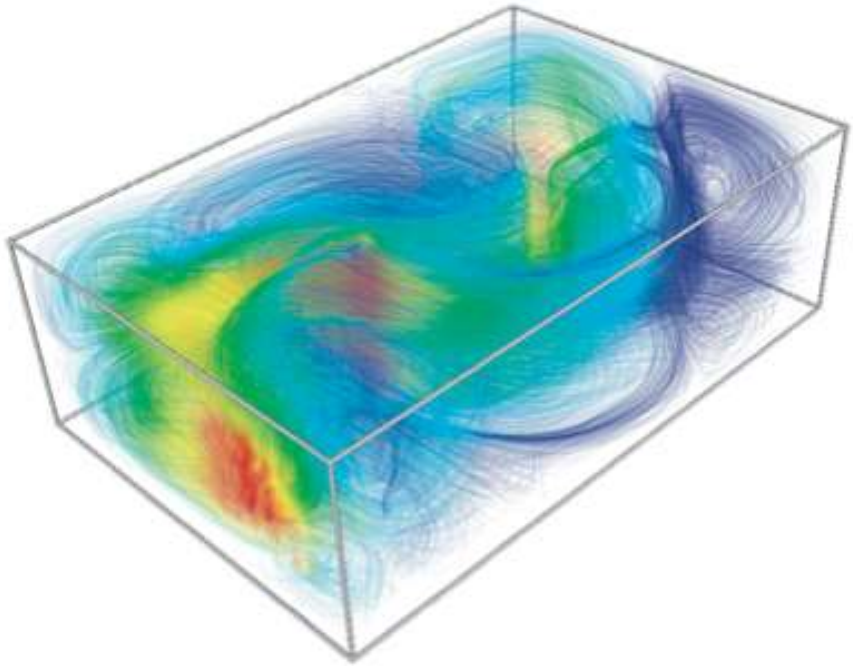




Streamlines in 3D

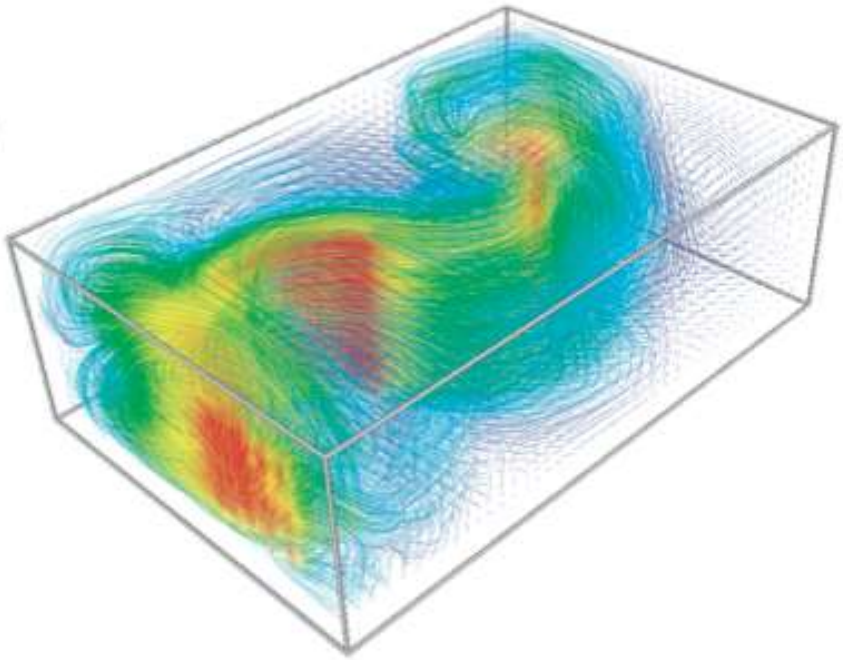
Variations

- play with opacity, seeding density, integration time



undersampling 3x3x3, opacity=0.1

- less occlusion (see through)
- good coverage

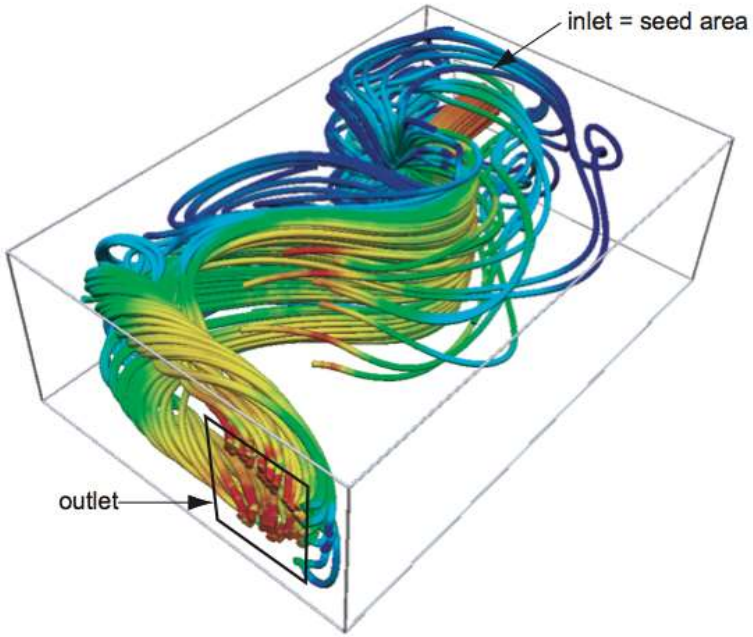


undersampling 3x3x3, shorter time

- more local insight (better coverage)
- even less occlusion
- but less continuity

Stream Tubes in 3D

- even higher occlusion problem than for 2D streamlines
- must reduce number of seeds



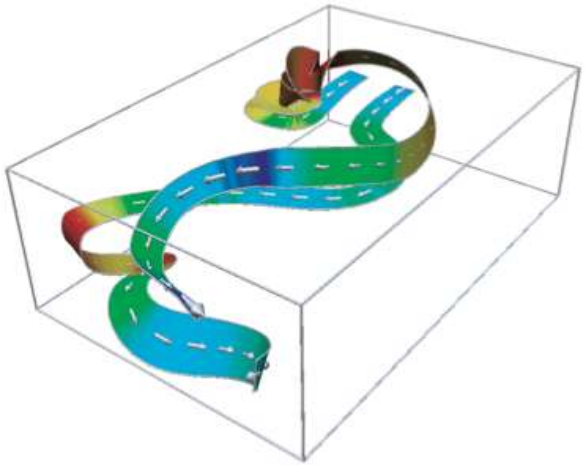
stream tubes traced from inlet to outlet

- show where incoming flow arrives at
- color by flow velocity
- shade for extra occlusion cues

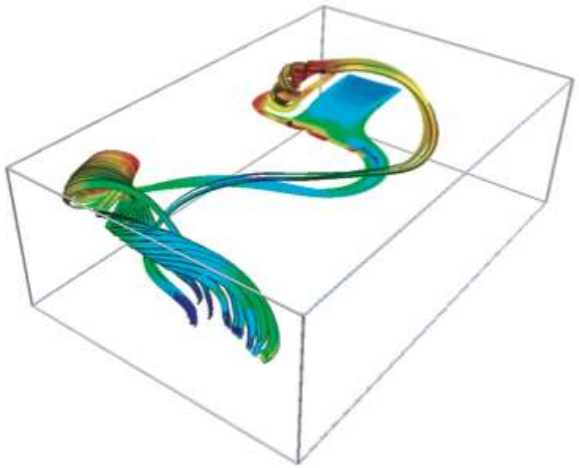
Stream Ribbons

- visualize how the vector field 'twists' around itself as it advances in space
- visualizes the so-called *helicity* of a vector field

$$h = \frac{1}{2}\mathbf{v} \cdot \mathrm{rot}\,\mathbf{v}$$



stream ribbons: two thick ribbons



stream ribbons: 20 thin ribbons

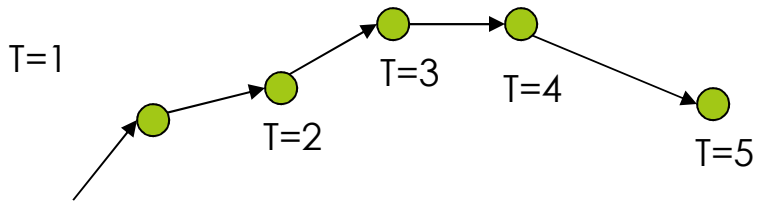
Algorithm

- define pairs of close seeds (p_a, p_b)
- trace streamlines S_a, S_b from (p_a, p_b)
- construct strip surface connecting closest points on S_a, S_b

Pathlines

-Extension of streamlines for time-varying vector fields (unsteady flows)

Pathline:

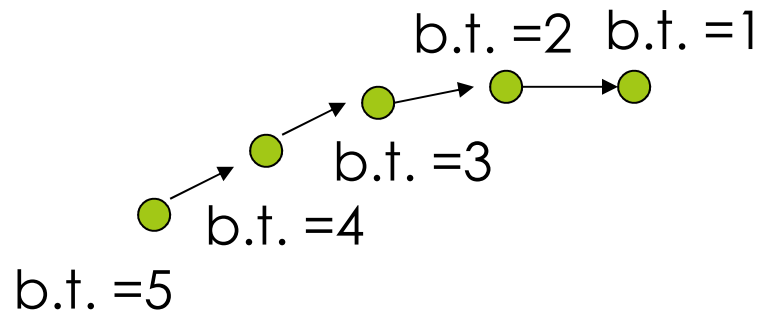


Traces path a particle has taken across the domain over time

Streaklines

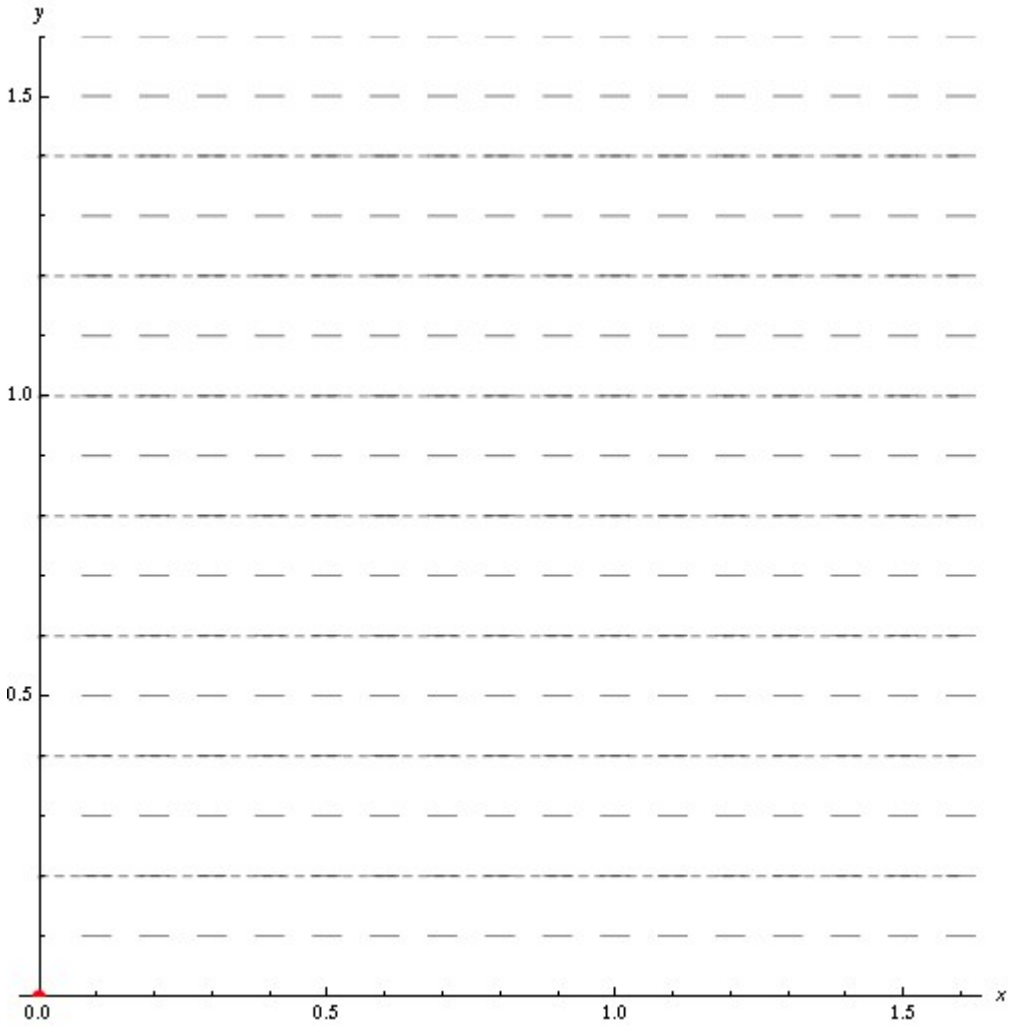
- For unsteady flows also

- Continuously injecting a new particle at each time step, advecting all the existing particles and connect them together into a *streakline*



Does it show you information across time?
Do pathlines and streamlines?

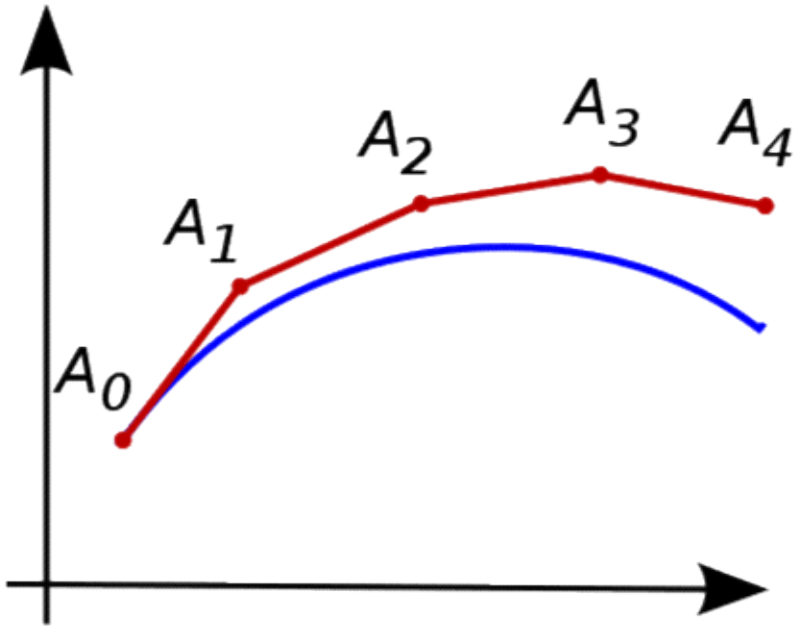
Pathlines and Streaklines

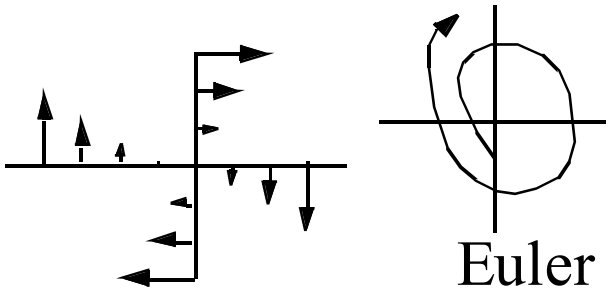


https://en.wikipedia.org/wiki/File:Streaklines_and_pathlines_animation.gif

Numerical Integration

- Euler Integration
  - $y_{n+1} = y_n + \Delta h f(x_n, y_n)$
  - Error is proportional to step-size
    - This is not good
    - Examples:





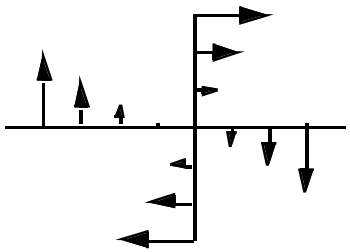
Numerical Integration

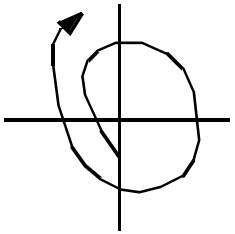
- 2nd order Runge-Kutta

$$y_{n+1} = y_n + ak_1 + bk_2$$

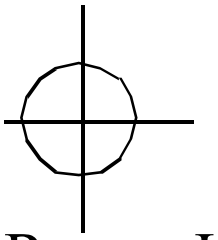
$$k_1 = \Delta h f\left(x_n, y_n\right)$$

$$k_2 = \Delta h f\left(x_n + \alpha\Delta h, y_n + \beta k_1\right)$$





Euler



Runge-Kutta

Runge Kutta Methods

- Used to solve Ordinary Differential Equations
- Good choice if function is easy to evaluate
- 4th-order Runge Kutta (RK4) preferred

Second Order Runge-Kutta

$$y_{n+1} = y_n + ak_1 + bk_2$$
$$k_1 = \Delta h f\left(x_n, y_n\right)$$
$$k_2 = \Delta h f\left(x_n + \alpha\Delta h, y_n + \beta k_1\right)$$

RK2 Example

Consider $\frac{dy}{dx} = -y^2$ Exact Solution $y = \frac{1}{1+x}$

The initial condition is: $y(0) = 1$

The step size is: $\Delta h = 0.1$

Use the coefficients $a = \frac{1}{2}\ b = \frac{1}{2},\ \ \alpha = \beta = 1$

RK2 Example

The values are

$$k_1 = \Delta h f\left(x_i, y_i\right)$$

$$k_2 = \Delta h f\left(x_i + \Delta h, y_i + k_1\right)$$

$$y_{i+1} = y_i + \frac{1}{2}\left[k_1 + k_2\right]$$

RK2 Example

			k_1	Estimate	Solution	k_2	Exact	Error
x_n	y_n	y'_n	hy'_n	y^*_{n+1}	$y^{*\prime}_{n+1}$	$h(y^{*\prime}_{n+1})$		
0	1.00000	-1.00000	-0.10000	0.90000	-0.81000	-0.08100	1.000000	0.000000
0.1	0.90950	-0.82719	-0.08272	0.82678	-0.68357	-0.06836	0.909091	-0.000409
0.2	0.83396	-0.69549	-0.06955	0.76441	-0.58433	-0.05843	0.833333	-0.000629
0.3	0.76997	-0.59286	-0.05929	0.71069	-0.50507	-0.05051	0.769231	-0.000740
0.4	0.71507	-0.51133	-0.05113	0.66394	-0.44082	-0.04408	0.714286	-0.000789
0.5	0.66747	-0.44551	-0.04455	0.62292	-0.38802	-0.03880	0.666667	-0.000801
0.6	0.62579	-0.39161	-0.03916	0.58663	-0.34413	-0.03441	0.625000	-0.000790
0.7	0.58900	-0.34692	-0.03469	0.55431	-0.30726	-0.03073	0.588235	-0.000768
0.8	0.55629	-0.30946	-0.03095	0.52535	-0.27599	-0.02760	0.555556	-0.000738
0.9	0.52702	-0.27775	-0.02778	0.49925	-0.24925	-0.02492	0.526316	-0.000705
1	0.50067	-0.25067	-0.02507	0.47560	-0.22620	-0.02262	0.500000	-0.000671

RK4

The general form of the equations:

$$\Delta y = \frac{1}{6}\left[k_1 + 2k_2 + 2k_3 + k_4\right]$$

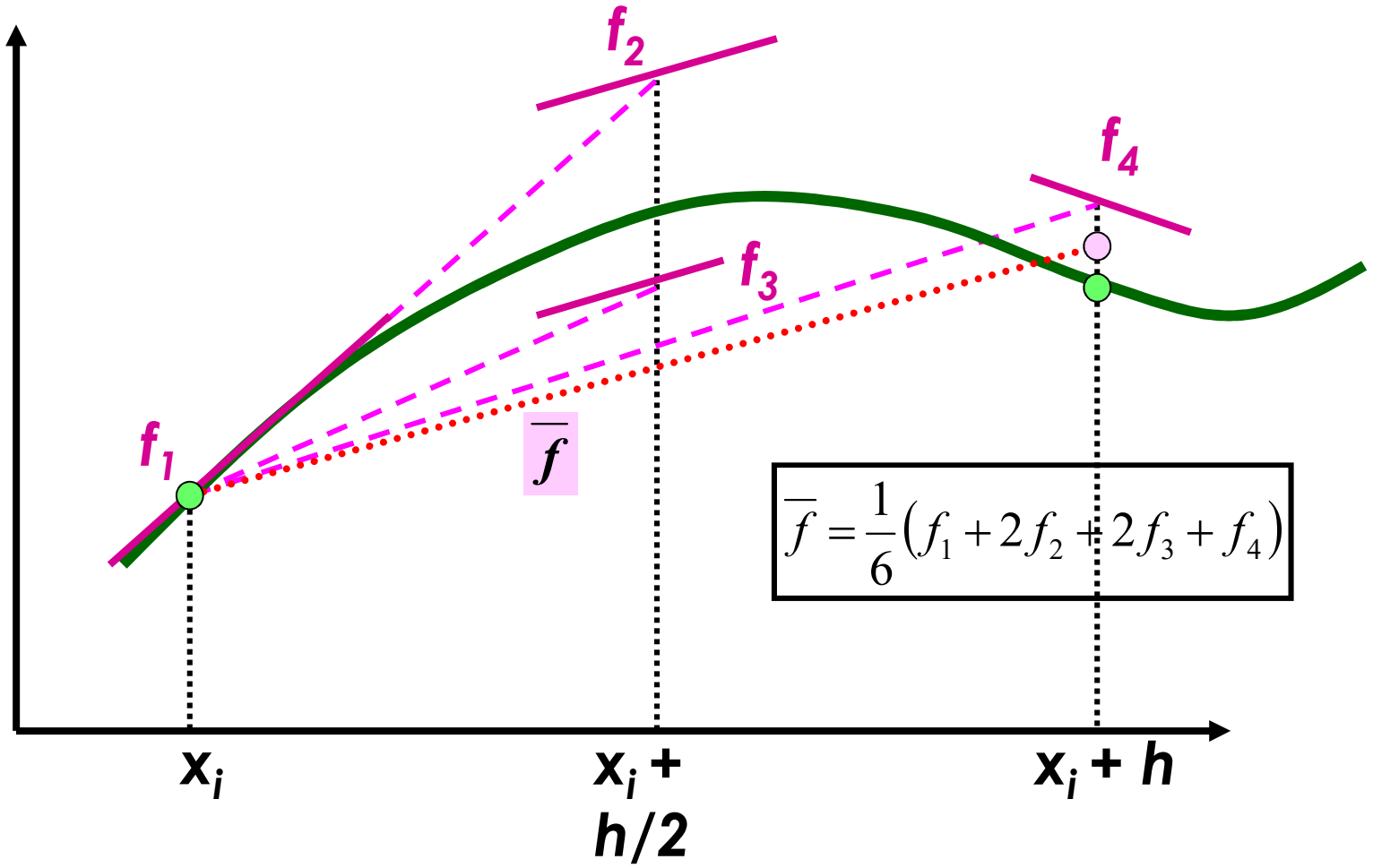
$$k_1 = \Delta h\left[f\left(x, y\right)\right]$$

$$k_2 = \Delta h\left[f\left(x + \frac{1}{2}\Delta h, y + \frac{1}{2}k_1\right)\right]$$

$$k_3 = \Delta h\left[f\left(x + \frac{1}{2}\Delta h, y + \frac{1}{2}k_2\right)\right]$$

$$k_4 = \Delta h\left[f\left(x + \Delta h, y + k_3\right)\right]$$

RK4


f_2
f_4
f_3
f_1
$\overline{f}$
$\overline{f} = \frac{1}{6}(f_1 + 2f_2 + 2f_3 + f_4)$
x_i
$x_i + h/2$
$x_i + h$

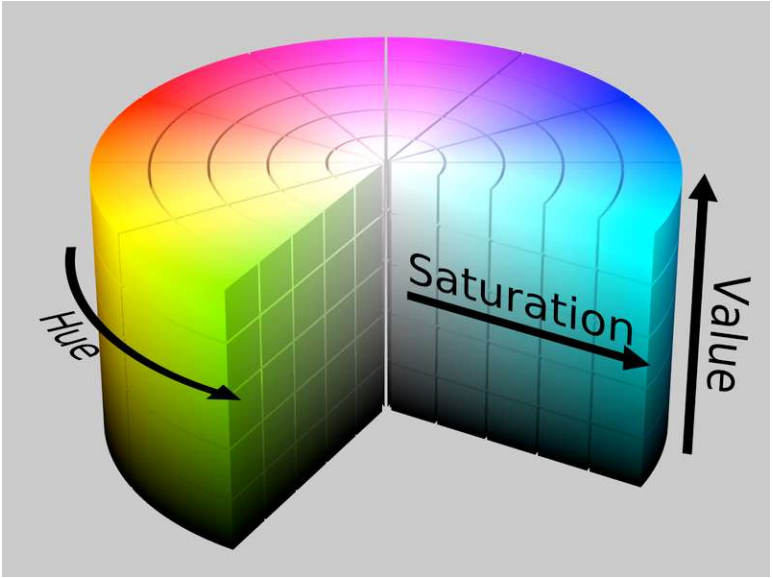
Vector Color Coding: Mapping to HSV

- Similar to scalar color mapping, vector color coding is to associate a color with every point in the data domain

- Typically, use HSV system (color wheel)
  - Hue is used to encode the direction of the vector (angle in the color wheel)

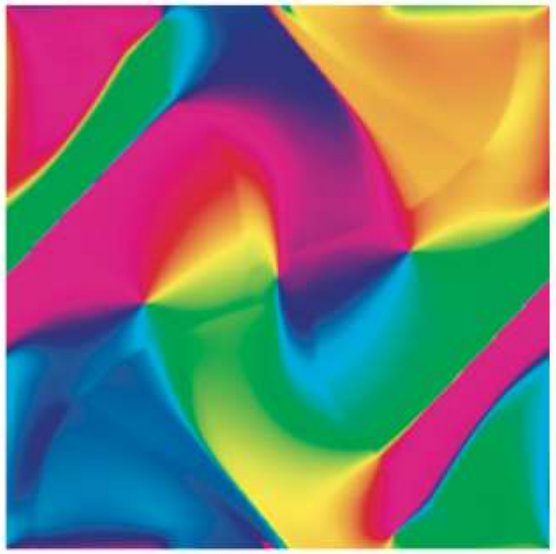
  - Value is magnitude of the vector

  - Saturation is set to one



Vector Color Coding

magnitude=luminance



constant luminance (direction coding only)

direction
color wheel

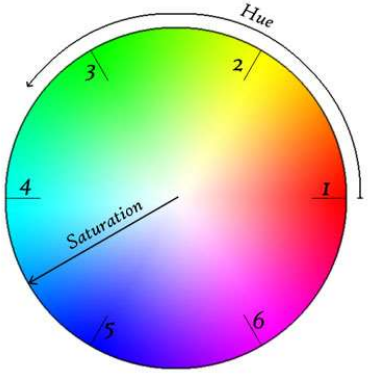
Reduce vector data to scalar data (using HSV color model)

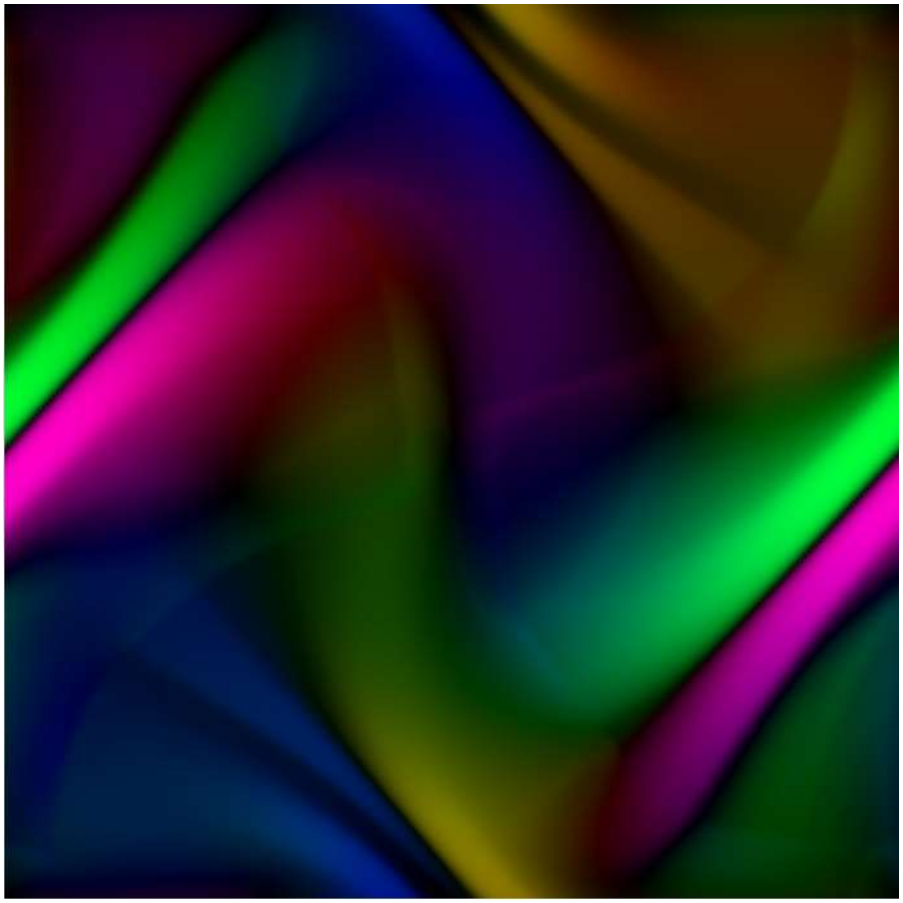
- direction = hue
- magnitude = luminance (optional)
- no occlusion/interpolation problems…
- …but images are highly abstract (recall: we don't naturally *see* directions)
- …doesn't extend to 3D without occlusion

Vector Color Coding

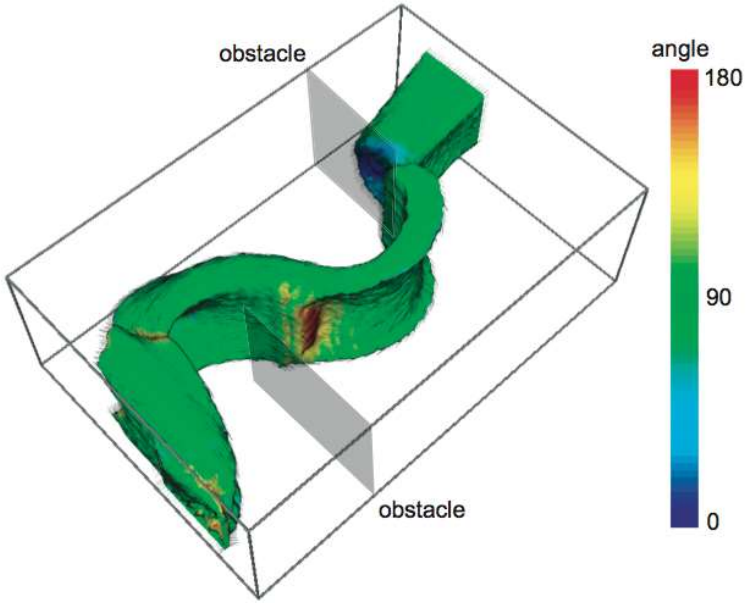
2-D Velocity Field of the MHD simulation:

Orientation, Magnitude





Vector Color Coding



color = angle between vector field and normal of some given surface

See if vectors are tangent to some given surface

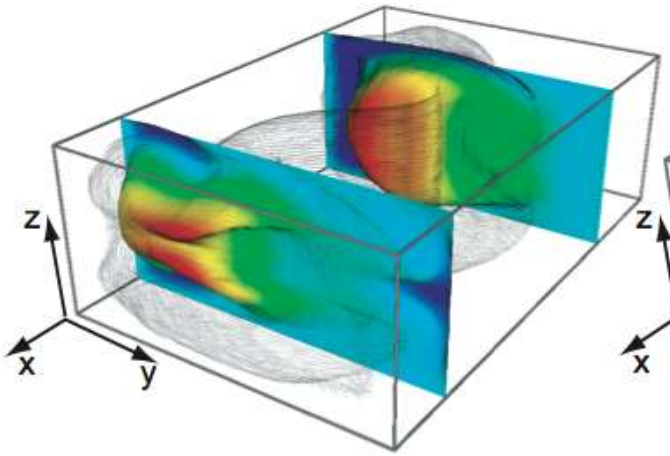
- color-code angle between vector and surface normal
- easily spot
  - tangent regions (flow stays on surface, green)
  - inflow regions (flow enters surface, red)
  - outflow regions (flow exits surface, blue)

Displacement Plots

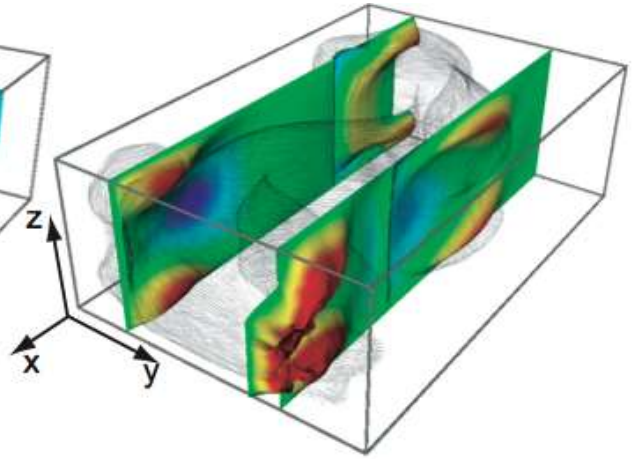
Show motion of a 'probe' surface in the field

- define probe surface $S \subseteq D$
- create displaced surface

$$S_{displ} = \{x + \mathbf{v}(x)\Delta t, \ \forall x \in S\}$$



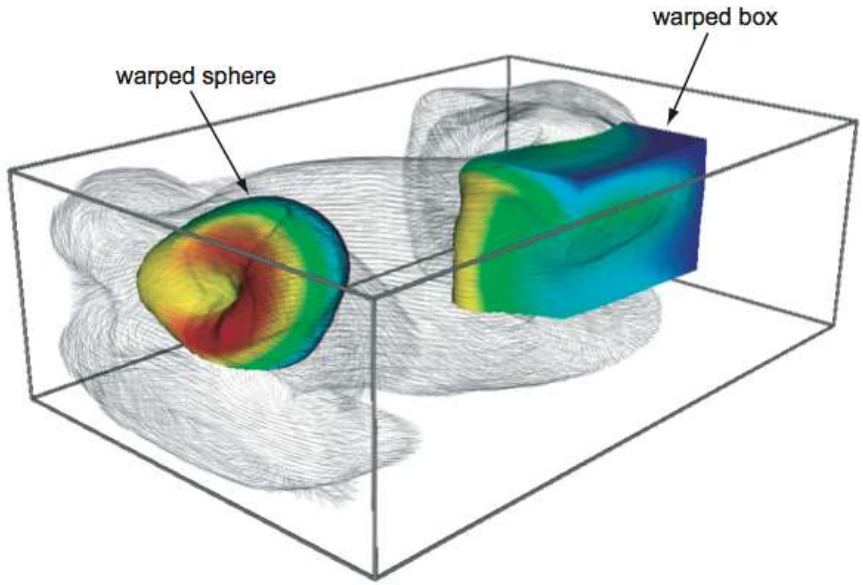
two displacement surfaces orthogonal to x axis



two displacement surfaces orthogonal to y axis

- analogy: think of a flexible sheet bent into the wind
- color can map additional scalar
- robust extension: $S_{displ} = \{x + (\mathbf{v}(x)\mathbf{n}(x))\mathbf{n}(x)\Delta t, \ \forall x \in S\}$
  - removes tangential displacements

Displacement Plots



we can displace any kind of surface

Added value

- see what a *specific* shape becomes like when warped in the vector field

Limitations

- cannot use too high displacement factors Δt
- self-intersections can occur
- we must choose an initial surface to warp ('seeding problem')