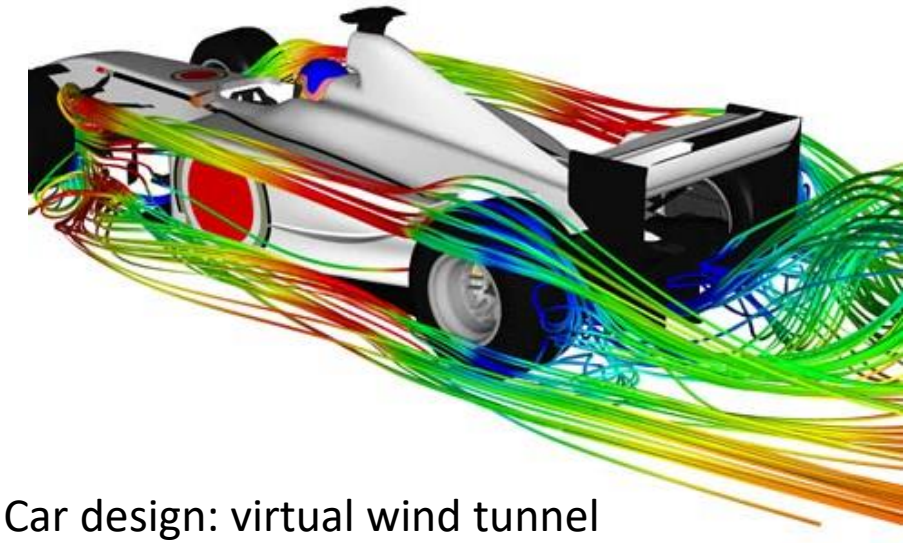
Visualization
– Flow Visualization

J.-Prof. Dr. habil. Kai Lawonn

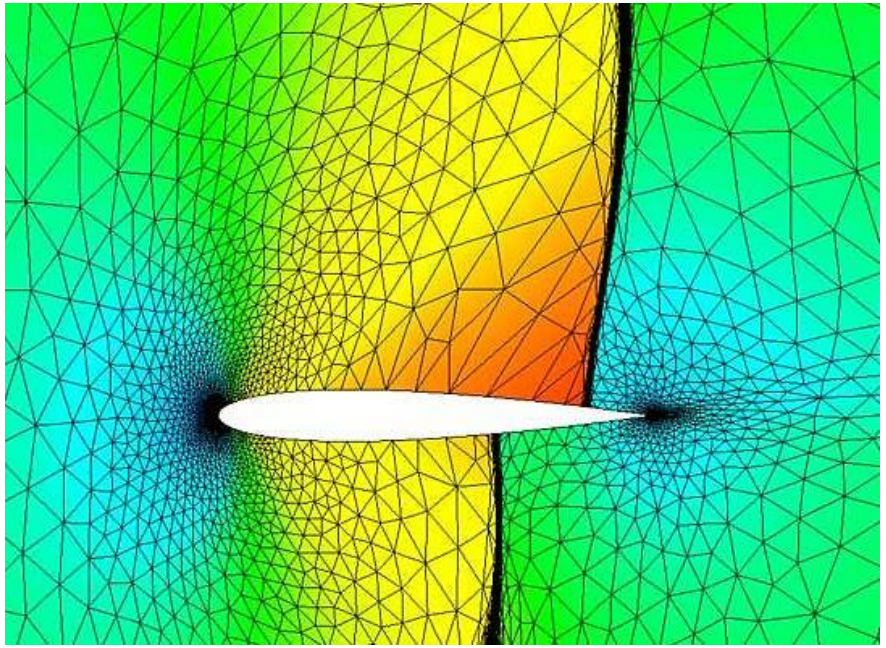
Real wind tunnel

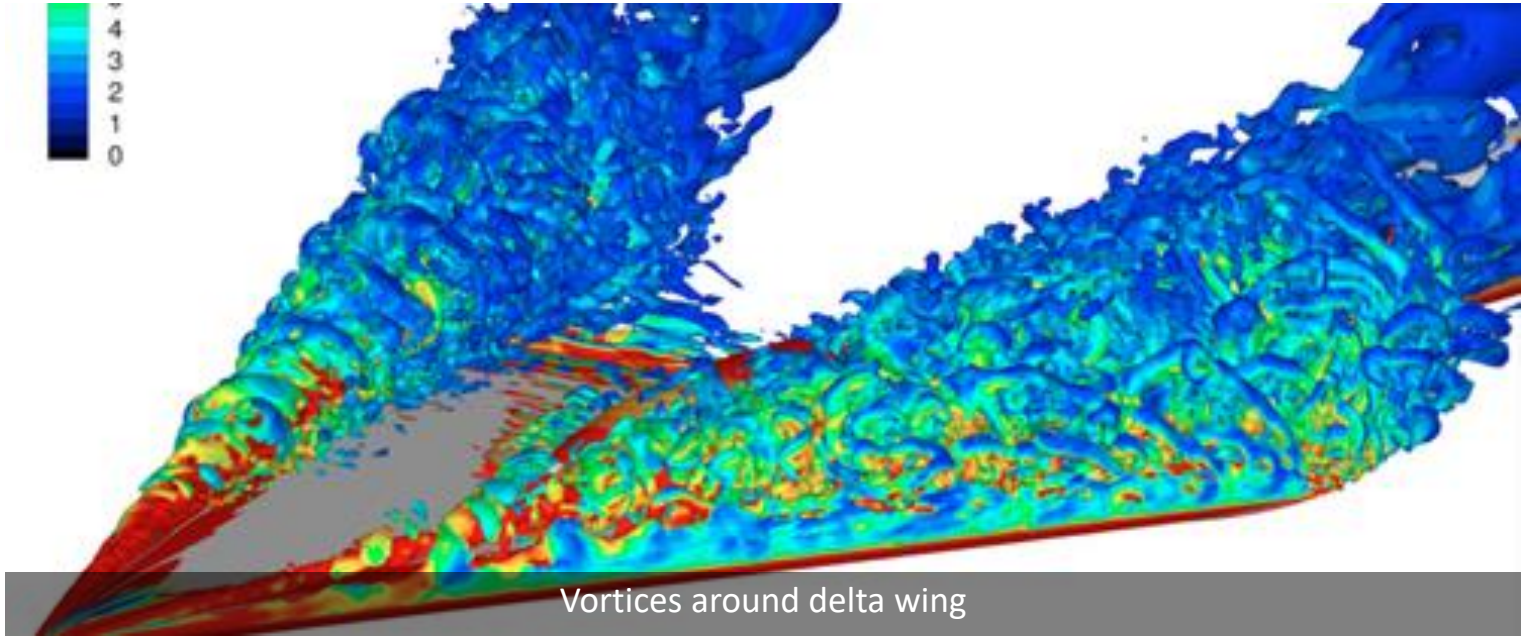


Car design: virtual wind tunnel

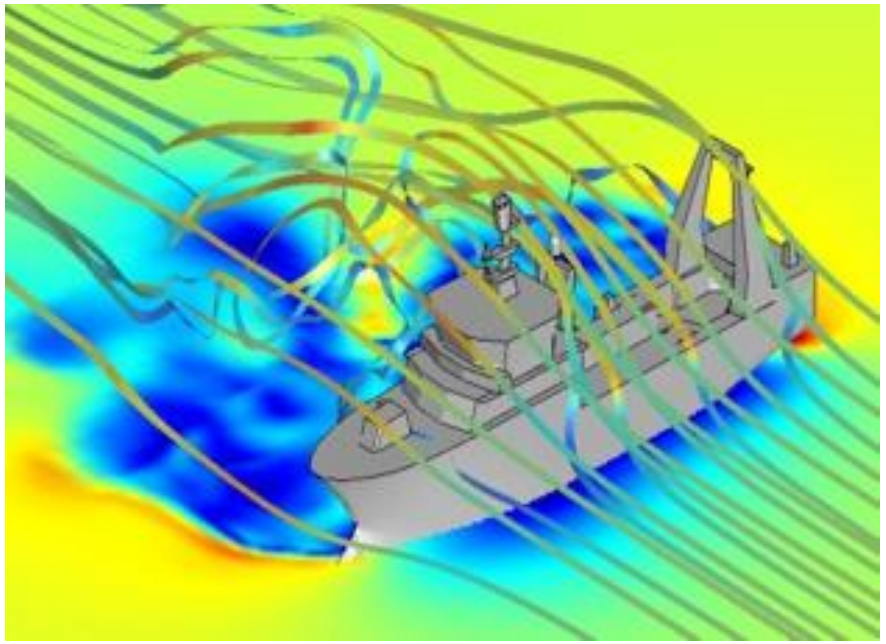
Flow visualization

Visualize vector-valued quantities like wind fields





Vortices around delta wing

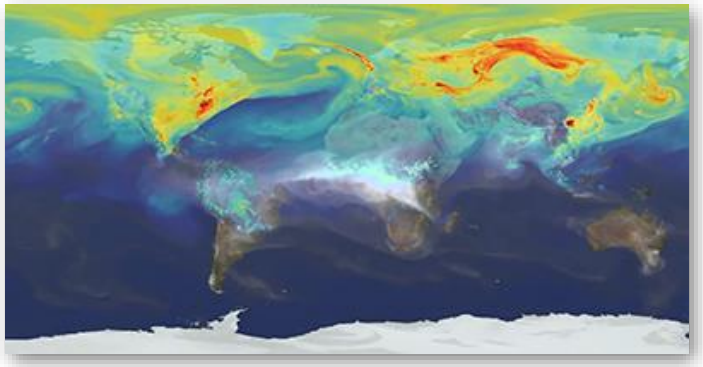


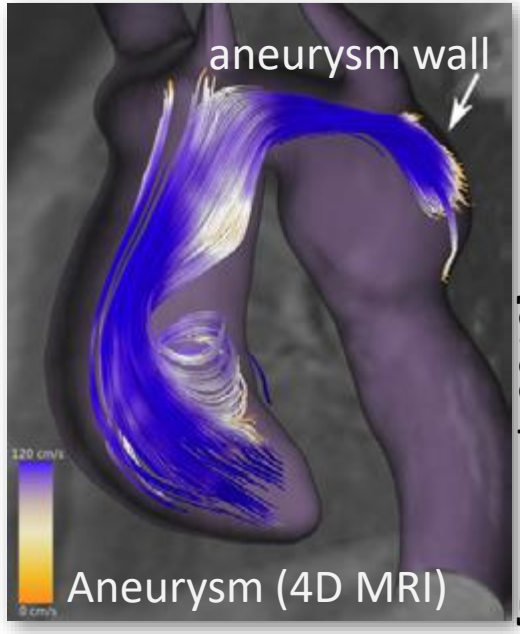
Flow visualization

Data from numerical simulations

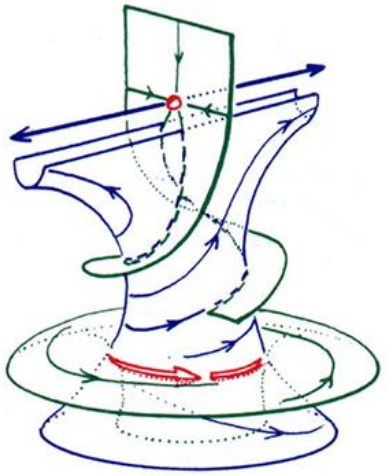
Flow visualization – Data

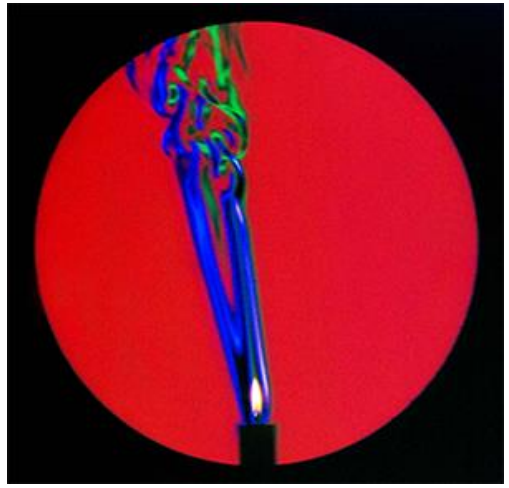
- Flow simulation
  - Design of ships, cars, airplanes, …
  - Weather simulations (e.g., atmospheric flow)
  - Medical blood flows
- Measurements
  - Wind tunnel
  - Schlieren imaging
- Modeling
  - Differential equations systems (dynamical systems)





[Born et al. 2012]





Color schlieren of burning candle

Vector field visualization

- Main application of flow visualization
  - Motion of fluids (gases, liquids)
  - Geometric boundary conditions
  - Velocity/flow field $\boldsymbol{v}(\boldsymbol{x}, t)$
  - Navier-Stokes equations
  - Computational fluid dynamics (CFD)

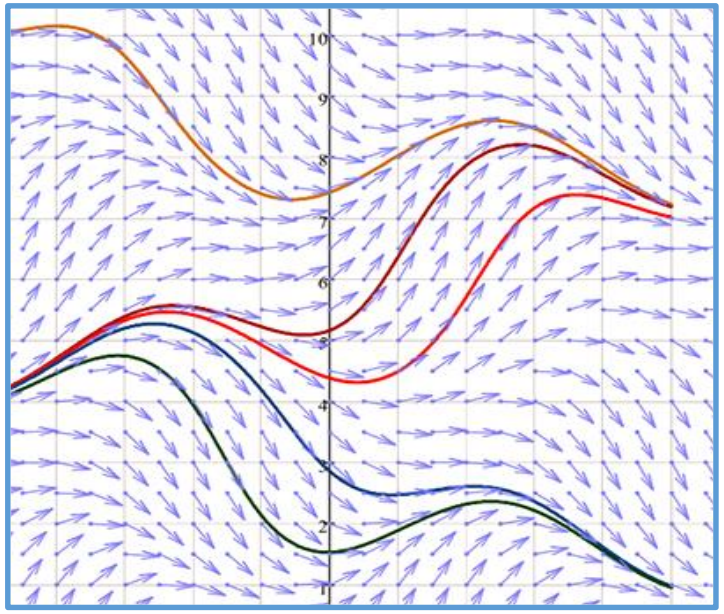
Vector field visualization

- Flow visualization – classification
  - Dimension (2D or 3D)
  - Time-dependency: steady vs. time-varying flows
  - Direct vs. indirect flow visualization

- In most cases, numerical methods required for flow visualization

Vector field visualization

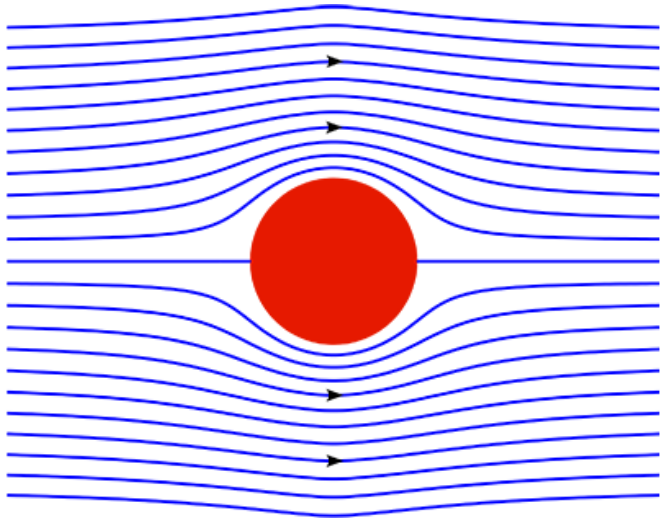
- Vector data representing direction & magnitude
  - Given by an n-tupel $(f_1, \dots, f_n)$ with $f_k = f_k(x_1, \dots, x_n)$, $n \geq 2$ and $1 \leq k \leq n$
  - Typically 2D $(n = k = 2)$ or 3D $(n = k = 3)$



- Example
  - 2D vector field where every sample represents a 2D vector (u, v) with $u = f(x, y)$ and $v = g(x, y)$

Steady vs. time-dependent flows

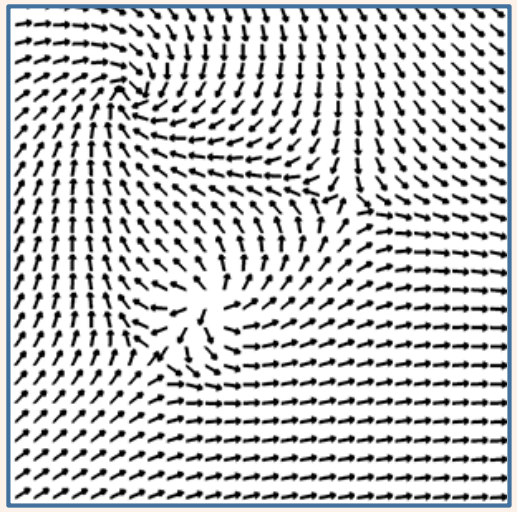
- Steady (time-independent) flow
  - Flow static over time
  - $\boldsymbol{v}(\boldsymbol{x}) : \mathbb{R}^n \rightarrow \mathbb{R}^n$, e.g., laminar flows
  - Simpler interrelationships



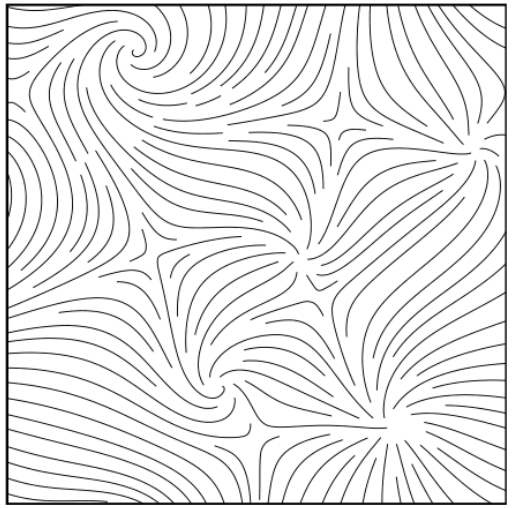
- Time-varying (unsteady) flow
  - Flow changes over time
  - $\boldsymbol{v}(\boldsymbol{x}, t) : \mathbb{R}^n \times \mathbb{R}^1 \rightarrow \mathbb{R}^n$, e.g., turbulent flows
  - More complex interrelationships

Turbulence from an airplane wing

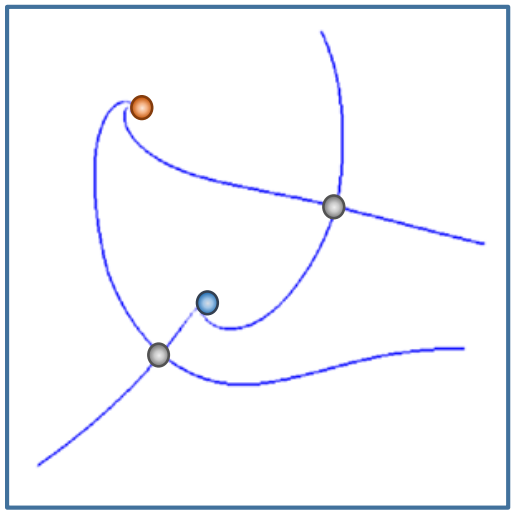
Flow visualization – Approaches



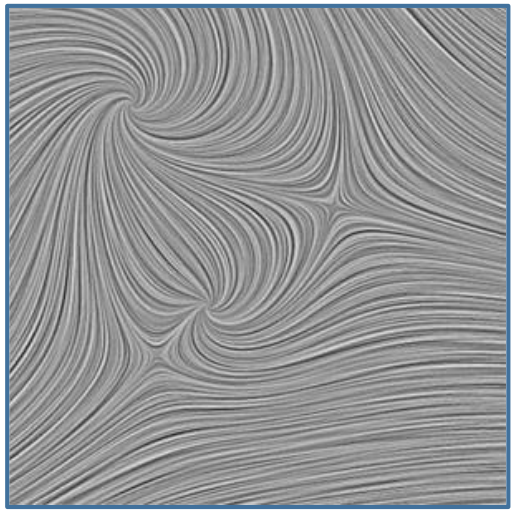
Direct flow visualization
(arrows, color coding, …)



Geometric flow visualization
(stream lines/surfaces, …)



Sparse (feature-based) vis.

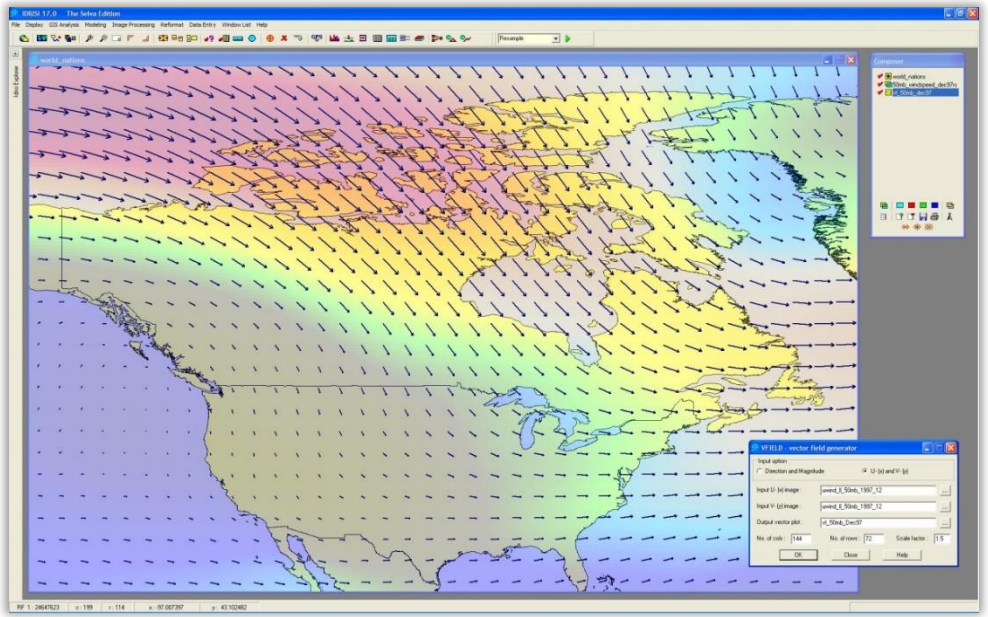


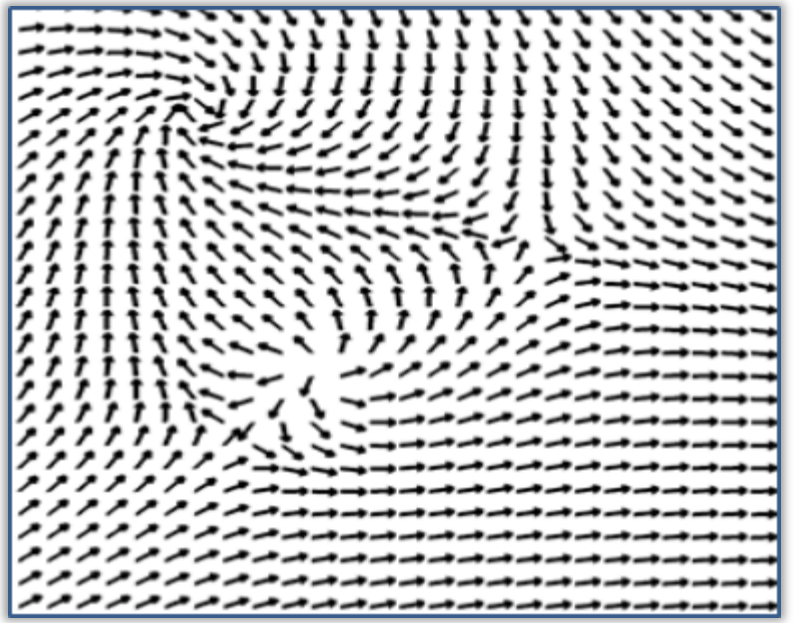
Dense (texture-based) vis.

Direct Flow Visualization

Flow visualization – Approaches

- Direct flow visualization
  - Color coding, arrow plots, glyphs
  - Gives overview on current flow state
  - Visualization of vectors





Vector and scalar functions

- Scalar function $\rho(\boldsymbol{x}, t)$
  - Gradient

$$\nabla\rho(\boldsymbol{x}, t) = \begin{pmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} \end{pmatrix} \rho(\boldsymbol{x}, t) = \begin{pmatrix} \frac{\partial}{\partial x}\rho(\boldsymbol{x}, t) \\ \frac{\partial}{\partial y}\rho(\boldsymbol{x}, t) \\ \frac{\partial}{\partial z}\rho(\boldsymbol{x}, t) \end{pmatrix}$$

$\boldsymbol{x}$ … position
t … time
∇… vector operator

Vector of partial derivatives

- Gradient takes a scalar field and gives us a vector field
- Gradient points into direction of max. change of $\rho(\boldsymbol{x}, t)$
- Gradient magnitude gives the slope along this direction

Vector and scalar functions

- Vector function $\boldsymbol{v}(\boldsymbol{x}, t)$
  - Jacobian matrix ("Gradient tensor")

$$\mathbf{J} = \nabla \boldsymbol{v}(\boldsymbol{x}, t) = \begin{pmatrix} \frac{\partial}{\partial x}\boldsymbol{v}_x & \frac{\partial}{\partial y}\boldsymbol{v}_x & \frac{\partial}{\partial z}\boldsymbol{v}_x \\ \frac{\partial}{\partial x}\boldsymbol{v}_y & \frac{\partial}{\partial y}\boldsymbol{v}_y & \frac{\partial}{\partial z}\boldsymbol{v}_y \\ \frac{\partial}{\partial x}\boldsymbol{v}_z & \frac{\partial}{\partial y}\boldsymbol{v}_z & \frac{\partial}{\partial z}\boldsymbol{v}_z \end{pmatrix}$$

Matrix of first-order partial derivatives of vector field $\boldsymbol{v}(\boldsymbol{x}, t)$

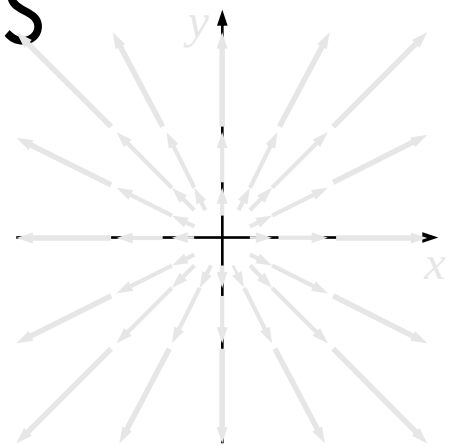
  - Divergence (dot product of vector operator)

$$div\ \boldsymbol{v}(\boldsymbol{x}, t) = \nabla \cdot \boldsymbol{v}(\boldsymbol{x}, t) = \frac{\partial}{\partial x}\boldsymbol{v}_x(\boldsymbol{x}, t) + \frac{\partial}{\partial y}\boldsymbol{v}_y(\boldsymbol{x}, t) + \frac{\partial}{\partial z}\boldsymbol{v}_z(\boldsymbol{x}, t)$$

Outflow per unit volume (rate at which density exits a region)

Vector and scalar functions

- Properties of divergence
  - Describes flow into/out of a region
  - $div\ \boldsymbol{v}$ is a scalar field
  - $div\ \boldsymbol{v}(\boldsymbol{x}_0) > 0$: $\boldsymbol{v}$ has a source in $\boldsymbol{x}_0$
  - $div\ \boldsymbol{v}(\boldsymbol{x}_0) < 0$: $\boldsymbol{v}$ has a sink in $\boldsymbol{x}_0$
  - $div\ \boldsymbol{v}(\boldsymbol{x}_0) = 0$: $\boldsymbol{v}$ is source-free in $\boldsymbol{x}_0$

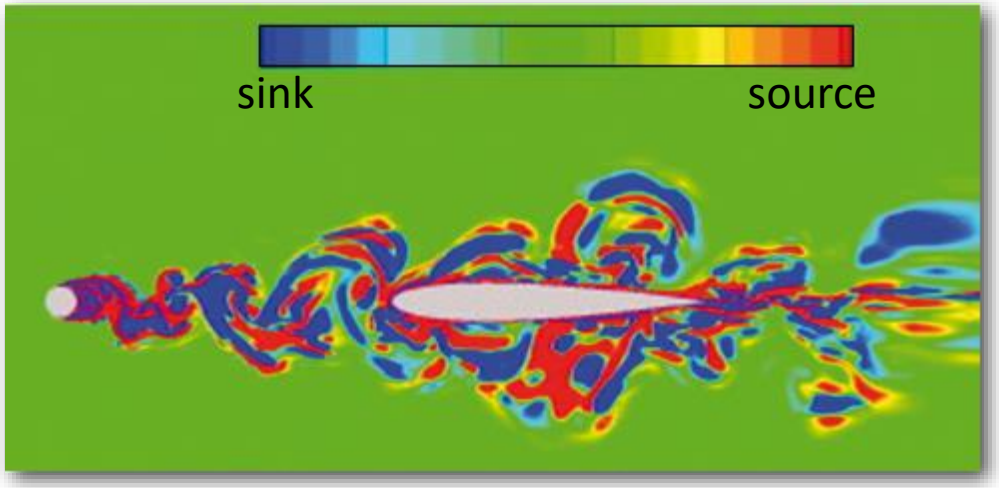


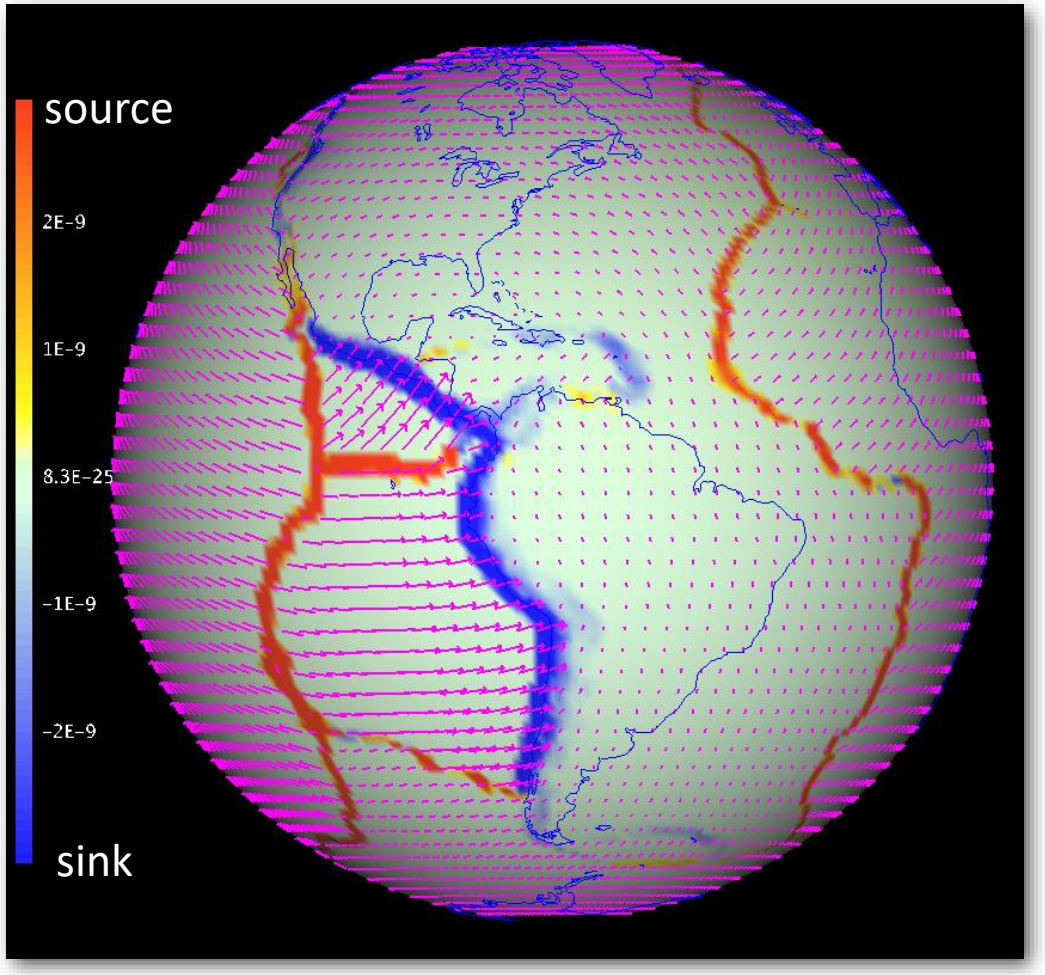
source

sink

Vector and scalar functions

- Color coding of divergence





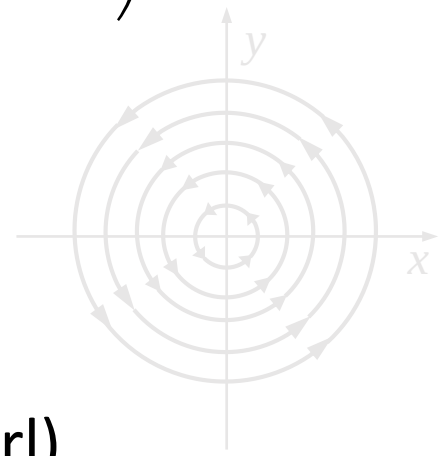
Tectonic plate motion

Vector and scalar functions

- Curl / vorticity

$$curl\ \boldsymbol{v}(\boldsymbol{x},t) = \nabla \times \boldsymbol{v}(\boldsymbol{x},t) = \begin{pmatrix} \frac{\partial}{\partial y}\boldsymbol{v}_z(\boldsymbol{x},t) - \frac{\partial}{\partial z}\boldsymbol{v}_y(\boldsymbol{x},t) \\ \frac{\partial}{\partial z}\boldsymbol{v}_x(\boldsymbol{x},t) - \frac{\partial}{\partial x}\boldsymbol{v}_z(\boldsymbol{x},t) \\ \frac{\partial}{\partial x}\boldsymbol{v}_y(\boldsymbol{x},t) - \frac{\partial}{\partial y}\boldsymbol{v}_x(\boldsymbol{x},t) \end{pmatrix}$$

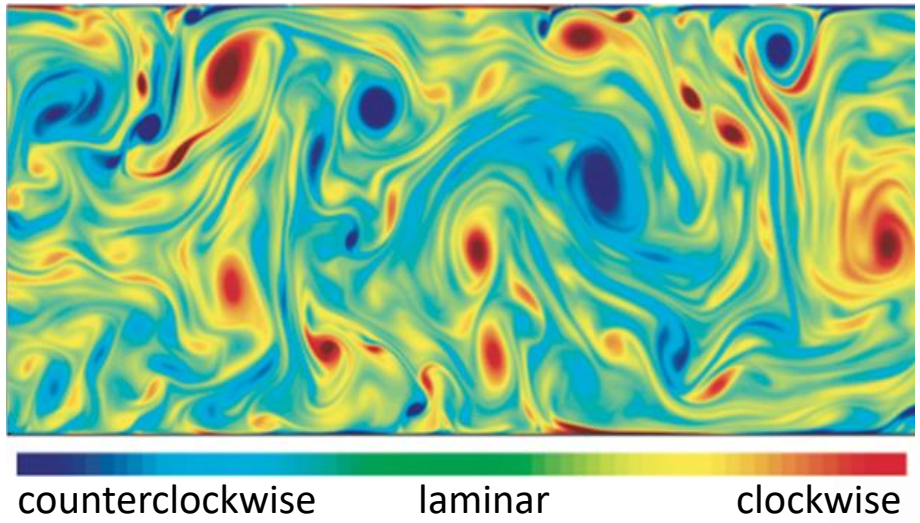
  - Curl is a vector-valued quantity
  - Describes vortex characteristics in flow

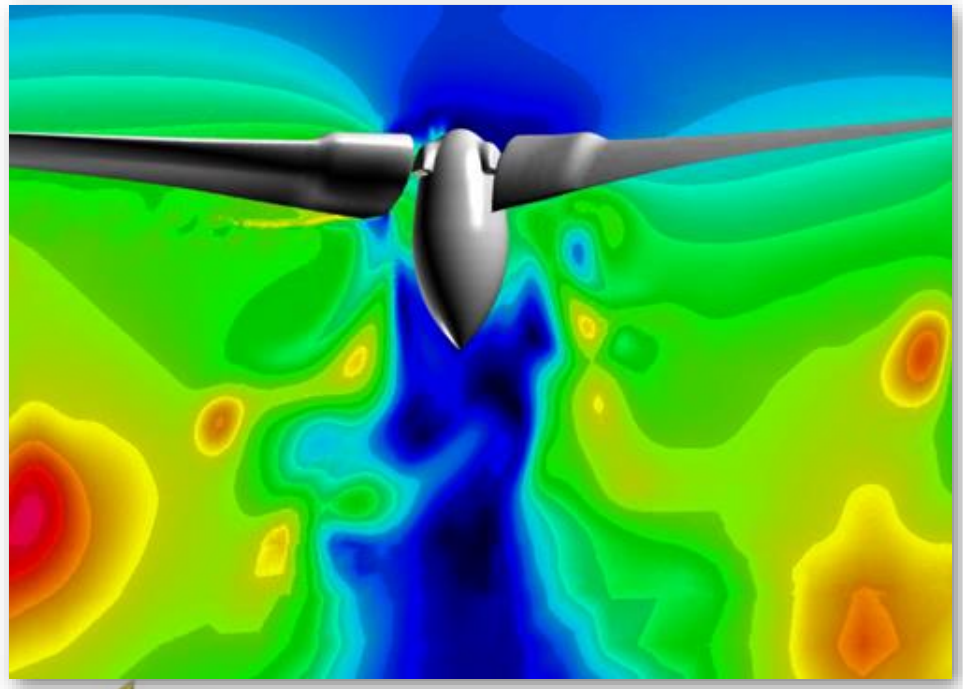


  - A measure of how fast the flow rotates (magnitude of the curl), and
  - … around which axis it rotates (direction of the curl)

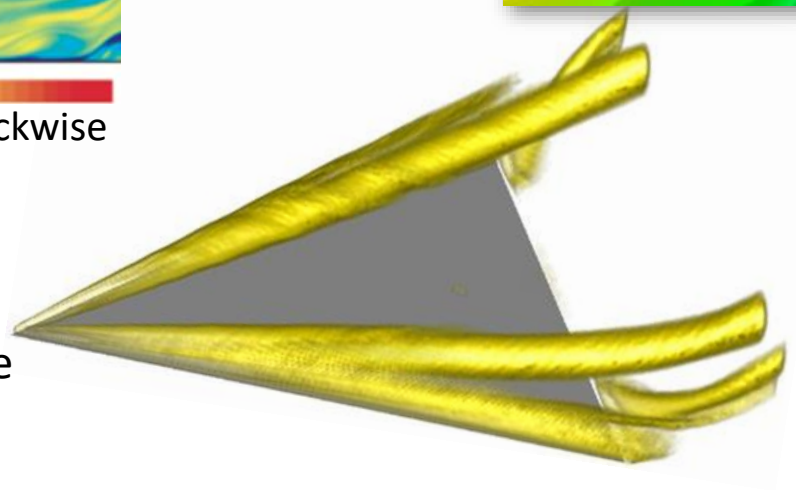
Vector and scalar functions

- Color coding of curl direction/magnitude





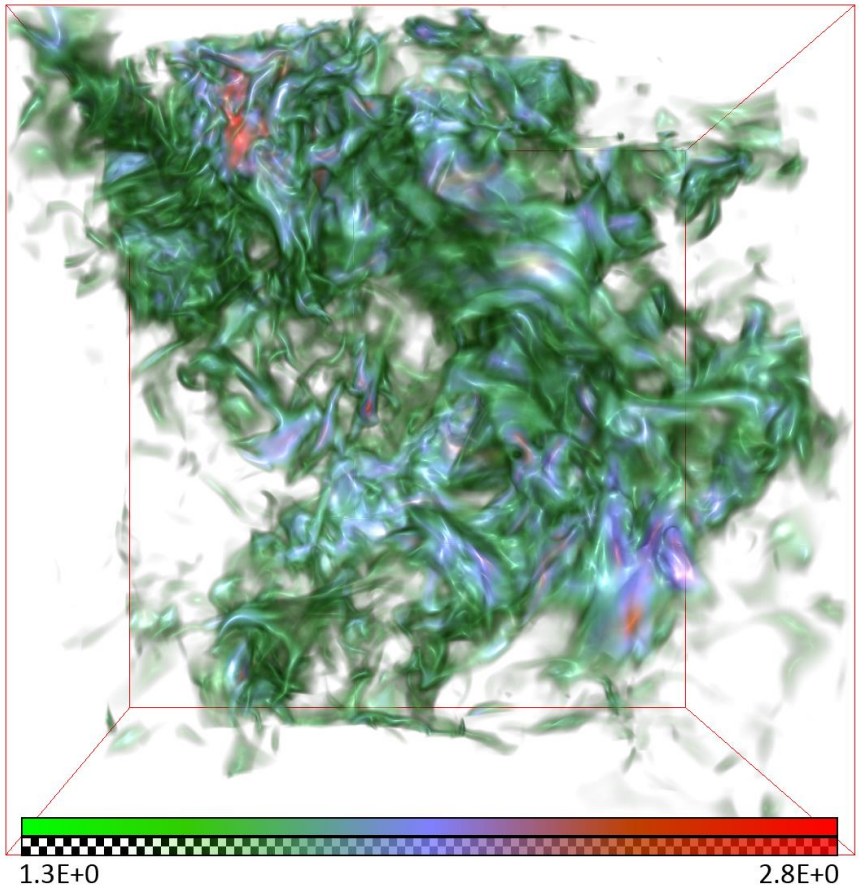
Curl magnitude



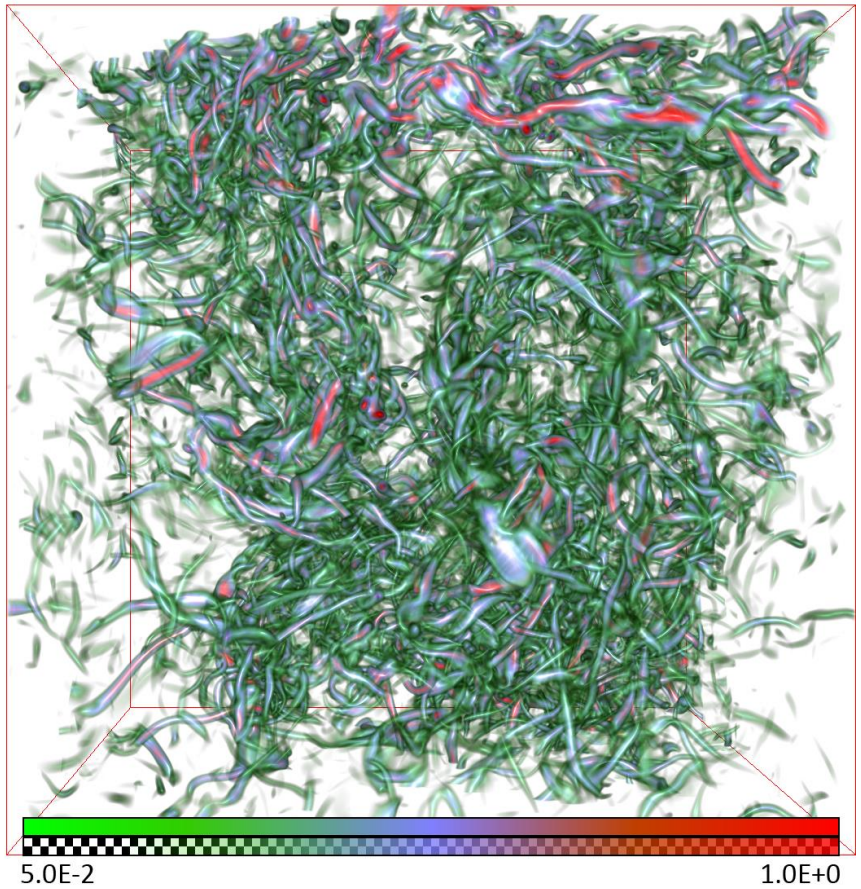
Threshold on
curl magnitude

Vector and scalar functions

- Direct volume rendering (DVR) of turbulent flow



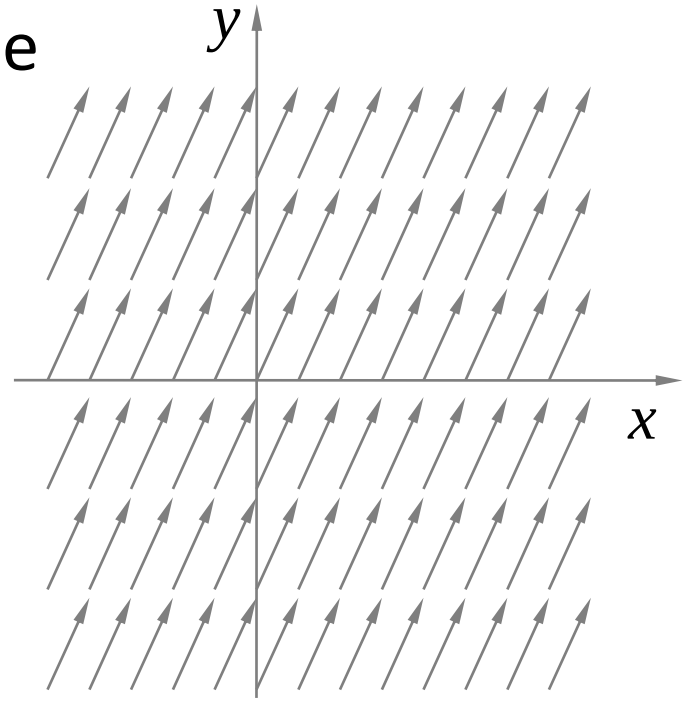
DVR of velocity magnitude



DVR of curl magnitude

Vector and scalar functions

- Example



A constant vector function

$$H_1(x,y,z) = \begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix}$$

$$J_{H_1}(x,y,z) = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

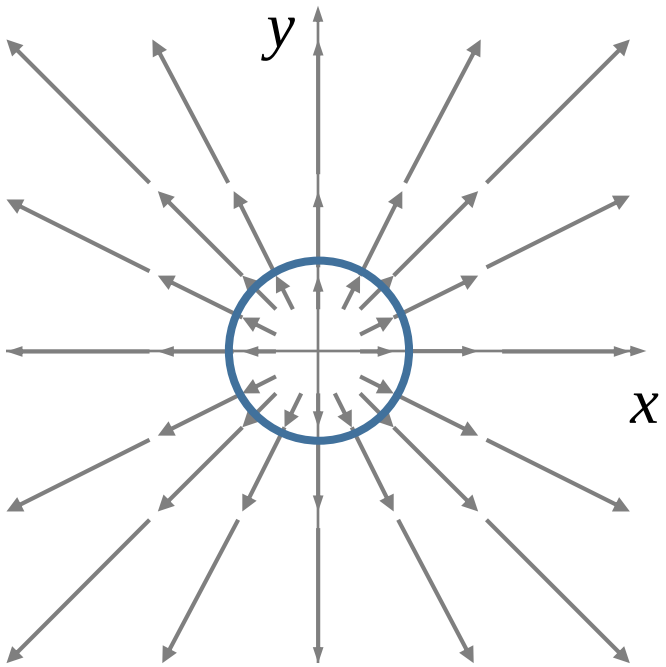
No velocity change in any direction

$$div(H_1)(x,y,z) = 0$$

$$curl(H_1)(x,y,z) = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

Vector and scalar functions

- Example



The 2D identity vector function

$$\text{Identity}: H_2(x,y,z) = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

Velocity increases with distance

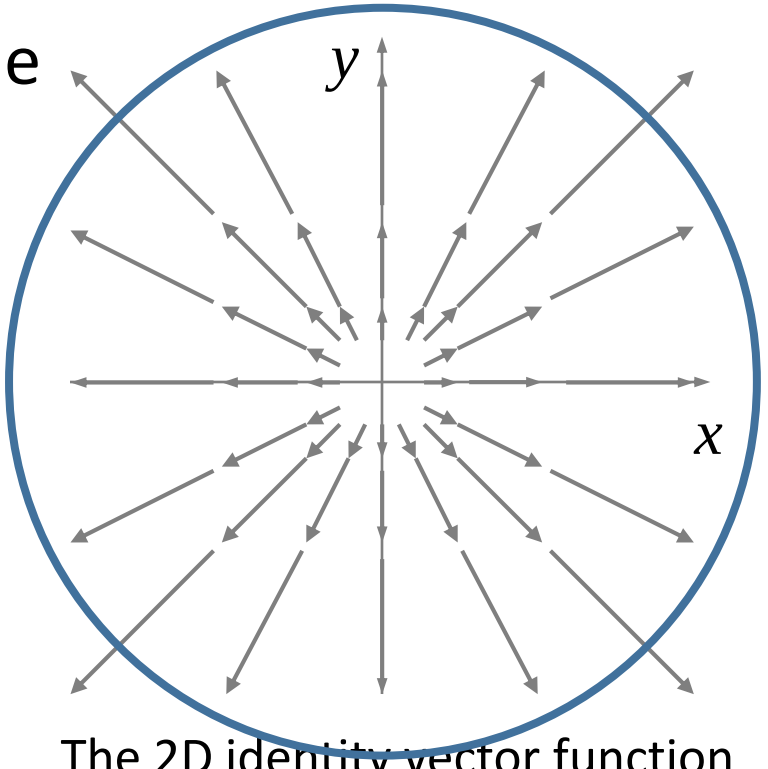
$$J_{H_2}(x,y,z) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$div(H_2)(x,y,z) = 3$$

$$curl(H_2)(x,y,z) = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

Vector and scalar functions

- Example



The 2D identity vector function

$$\text{Identity}: H_2(x,y,z) = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

Velocity increases with distance

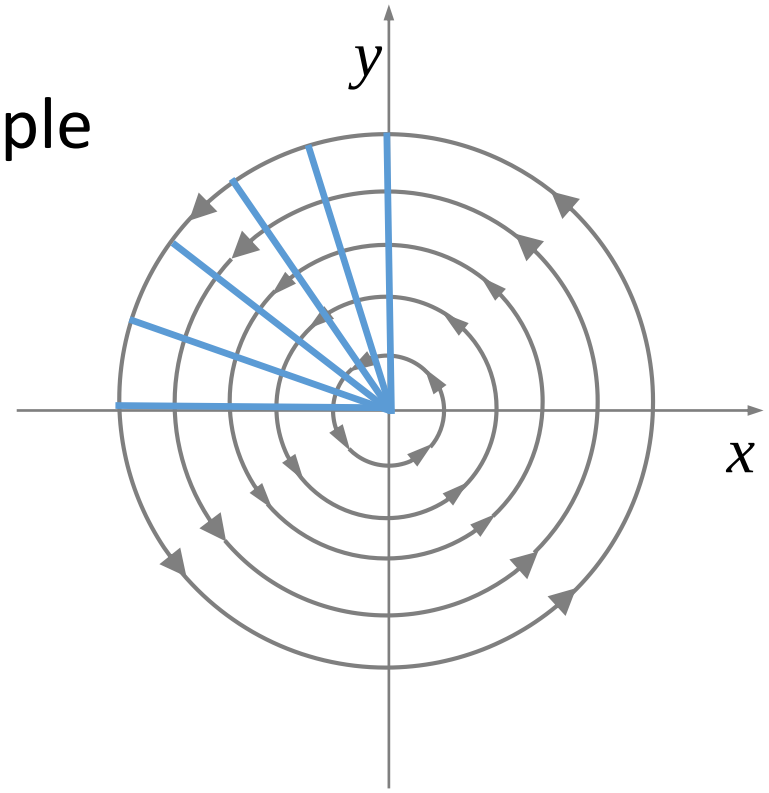
$$J_{H_2}(x,y,z) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$div(H_2)(x,y,z) = 3$$

$$curl(H_2)(x,y,z) = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

Vector and scalar functions

- Example



A 2D curl field

$$\text{Curl field}: H_3(x,y,z) = \begin{pmatrix} -y \\ x \\ 0 \end{pmatrix}$$

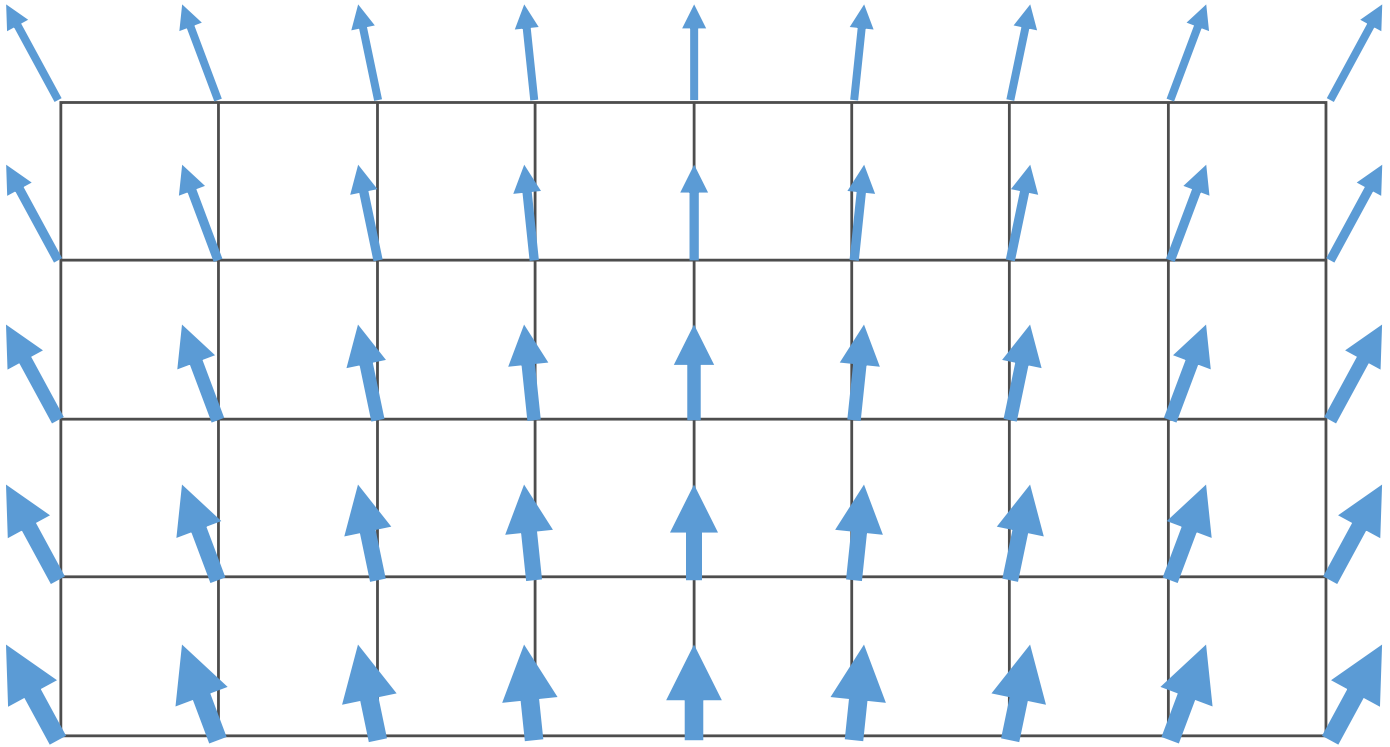
$$J_{H_3}(x,y,z) = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$div(H_3)(x,y,z) = 0$$

$$curl(H_3)(x,y,z) = \begin{pmatrix} 0 \\ 0 \\ 2 \end{pmatrix}$$

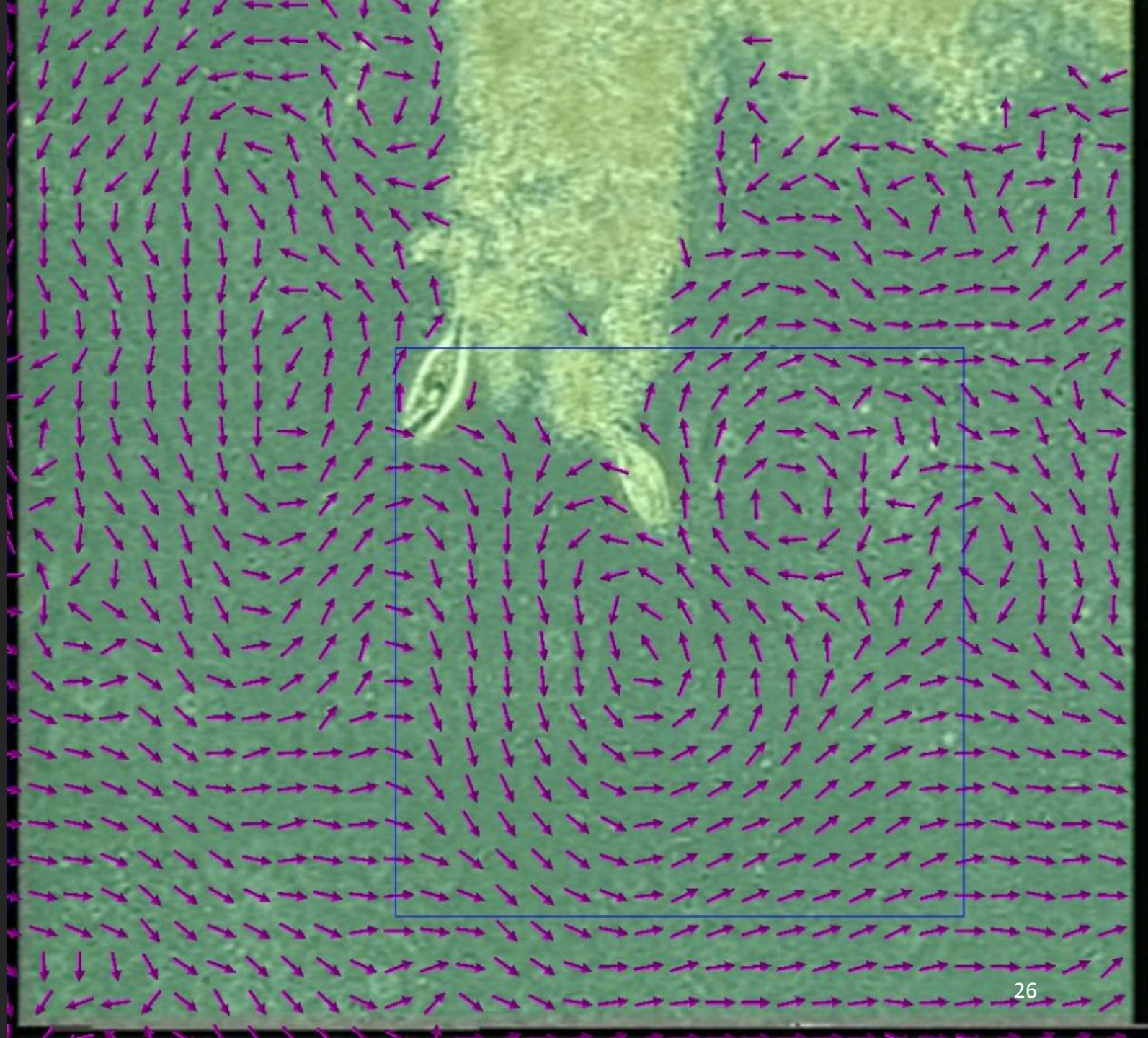
Flow visualization

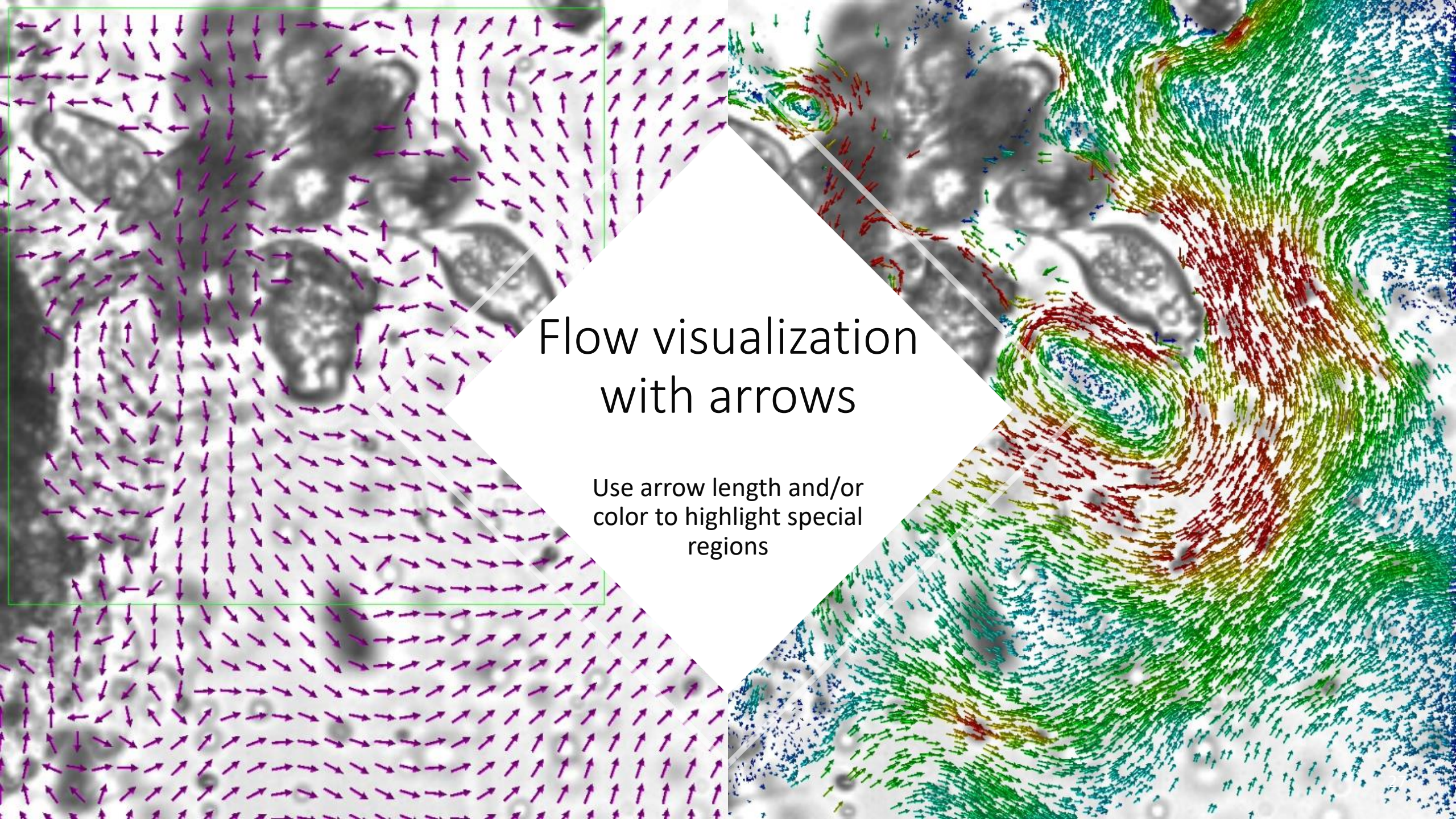
- Glyphs
  - Visualize local features of the vector field
  - Map vector or curl to arrow glyphs



Flow visualization with arrows

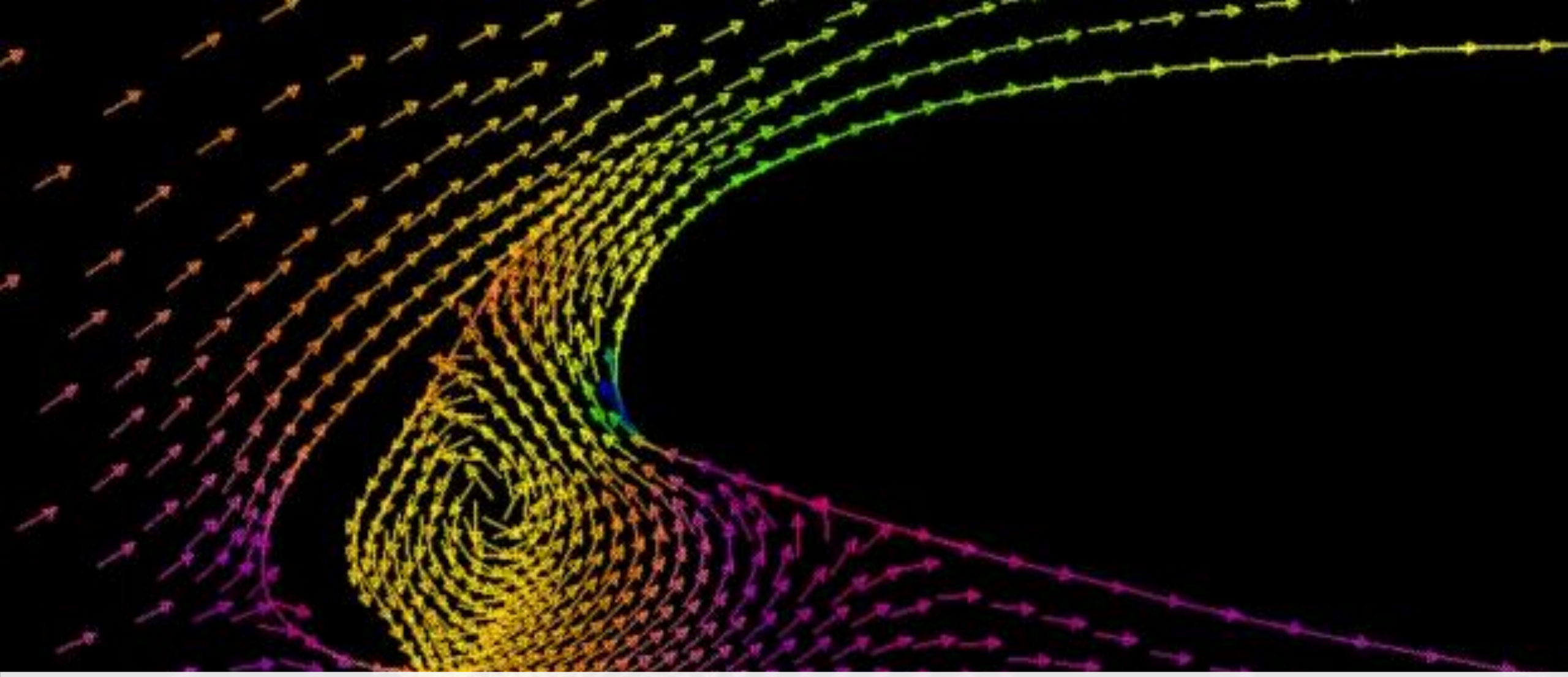
Vector per grid point pointing into the flow direction





Flow visualization with arrows

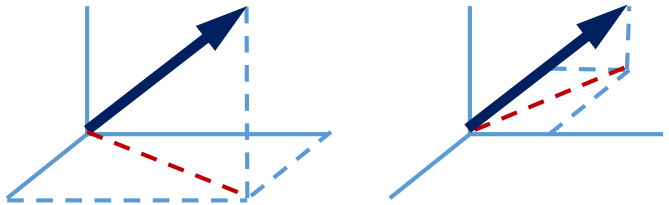
Use arrow length and/or color to highlight special regions



Flow visualization with arrows

Arrows in 3D

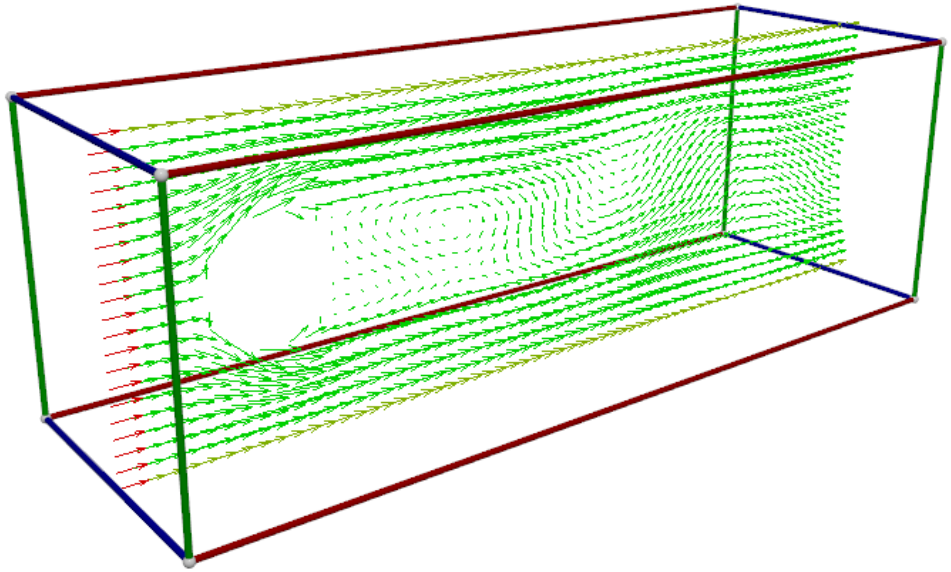
- Advantages
  - Simple
  - 3D effects
- Disadvantages
  - Ambiguity
  - Difficult spatial perception
  - Inherent occlusion effects
  - Poor results if magnitude of velocity varies significantly and changes rapidly

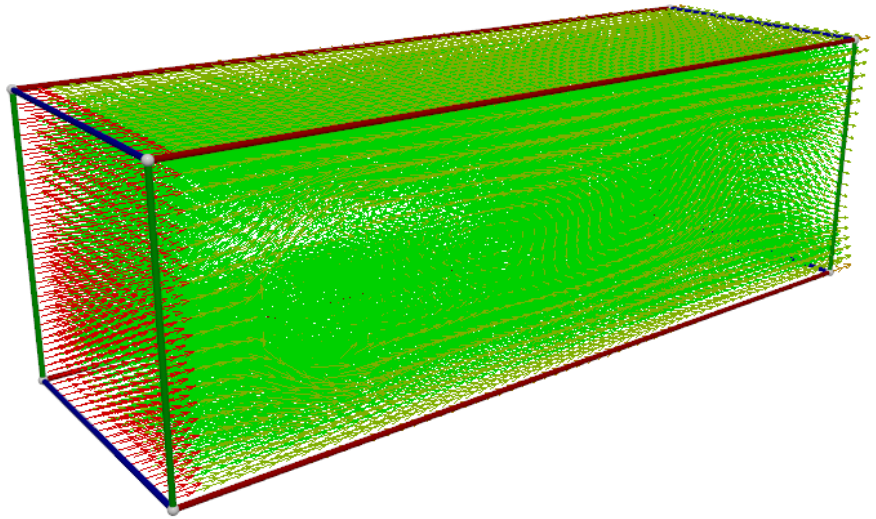


Use 3D arrows of constant length and color code magnitude

Arrows in 3D

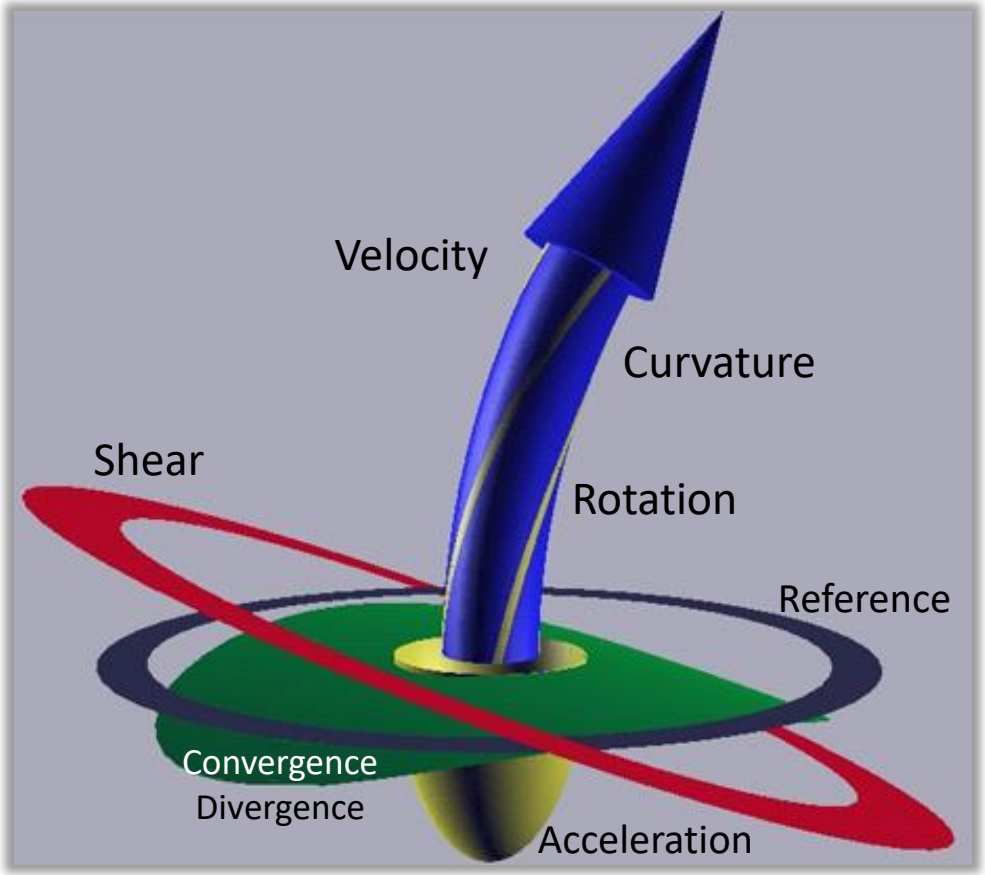
- Compromise
  - Arrows only in slices





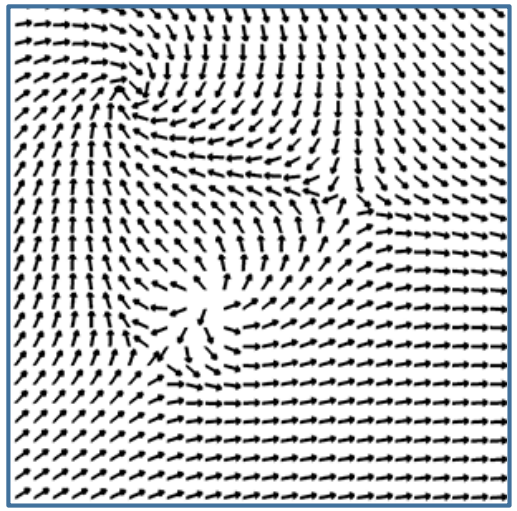
Flow visualization with glyphs

- Glyphs
  - Can visualize more features of vector field
  - Displays local information

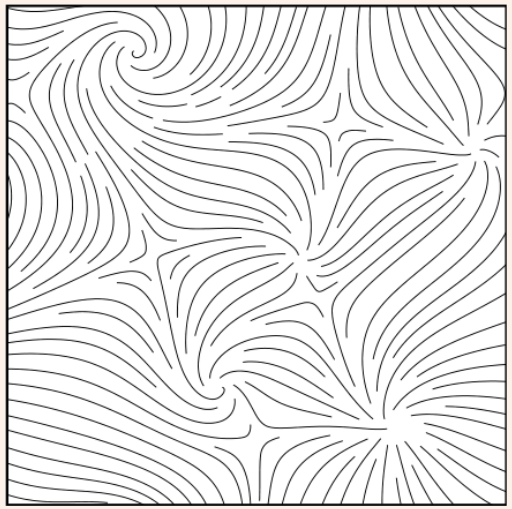


[de Leeuw and van Wijk 93]

Flow visualization – Approaches

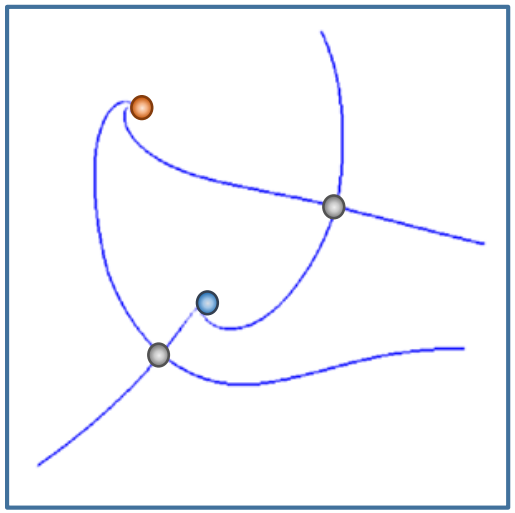


Direct flow visualization
(arrows, color coding, …)

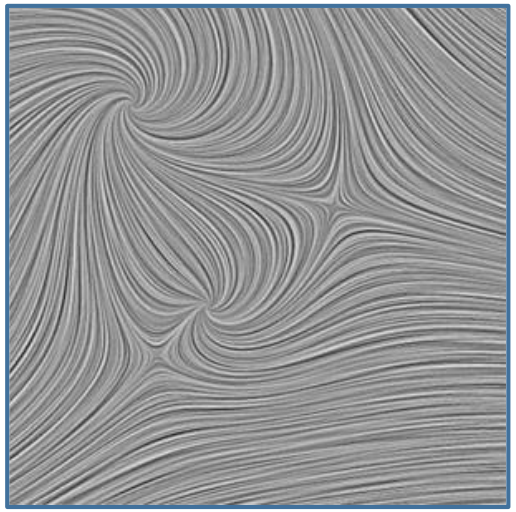


Geometric flow visualization
(stream lines/surfaces, …)

- Use intermediate representation (vector-field integration over time)
- Visualization of temporal evolution
- Stream lines, path lines, streak lines



Sparse (feature-based) vis.

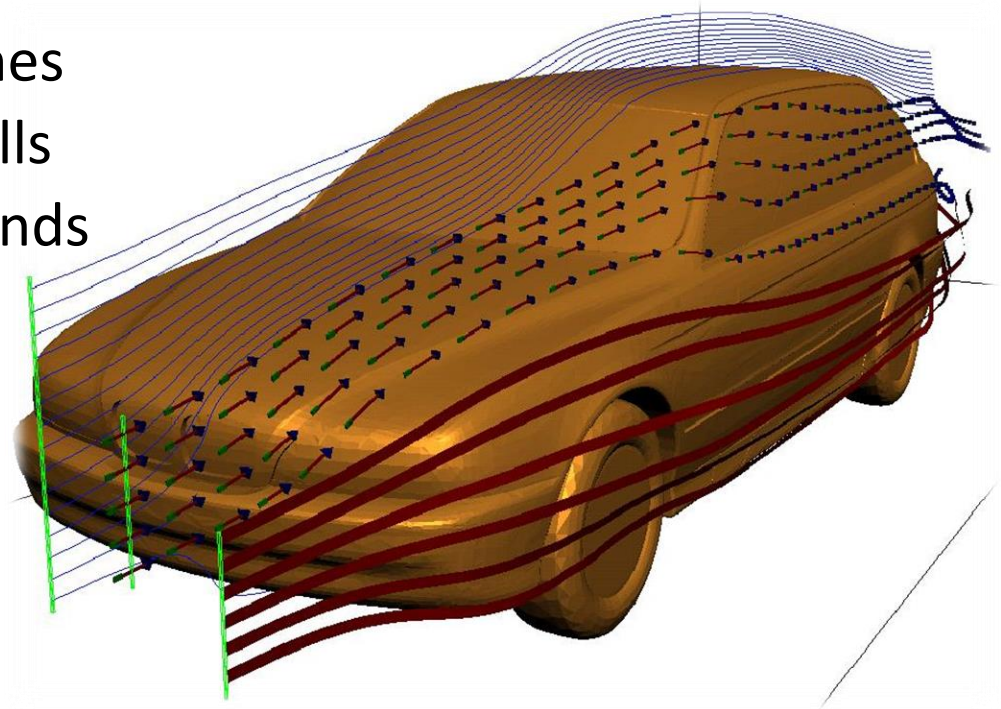


Dense (texture-based) vis.

Geometric Flow Visualization

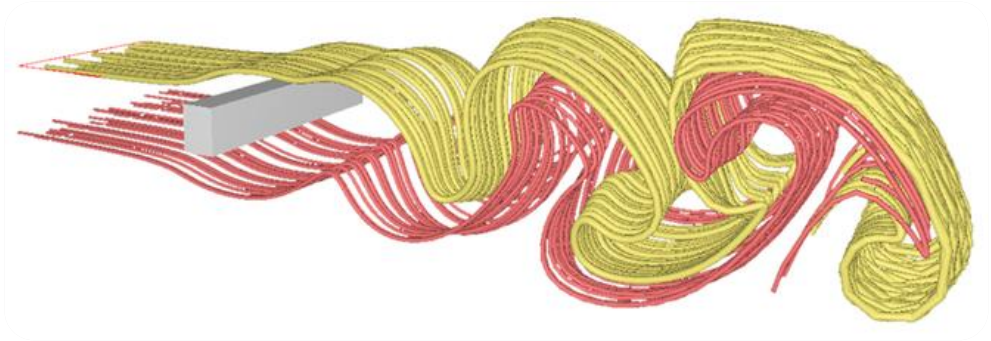
Characteristic Lines

- Basic idea: trace particles along characteristic trajectories
- Map trajectories to
  - Particles
  - Lines
  - Balls
  - Bands



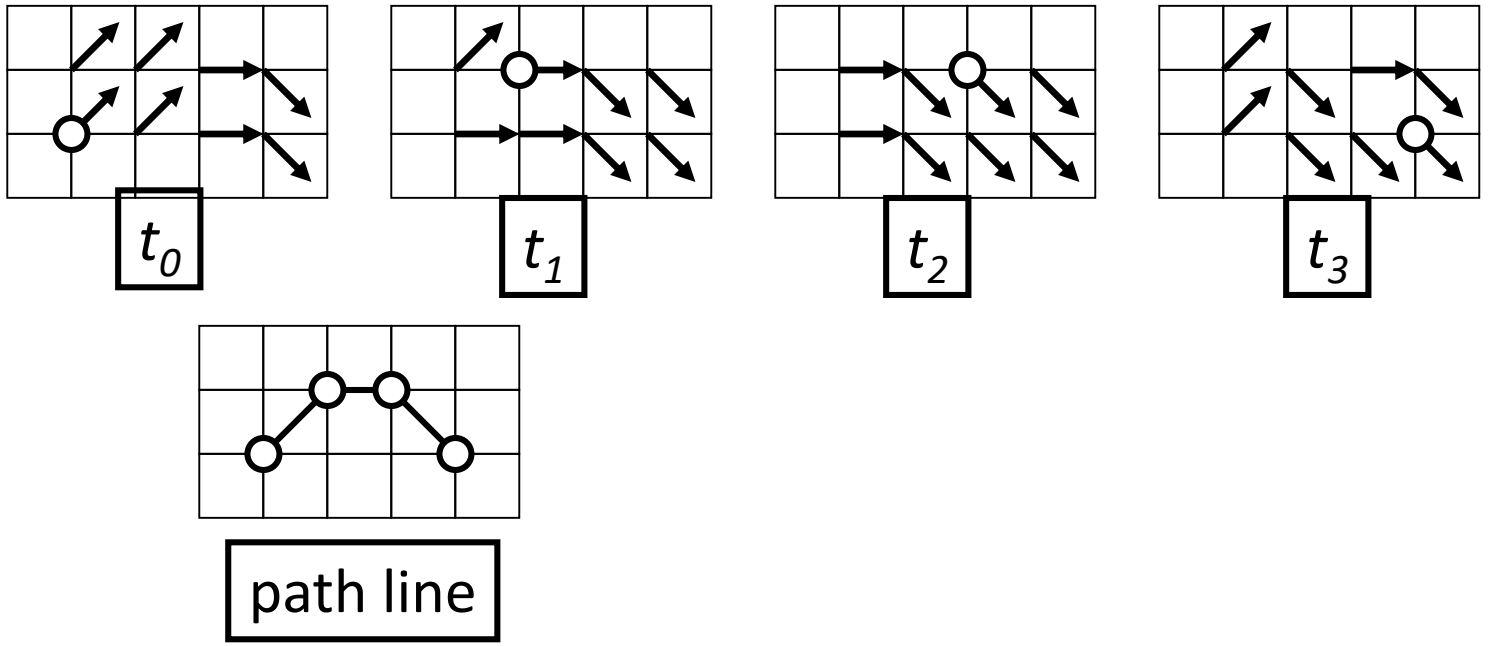
Characteristic Lines

- Types of characteristic lines
  - Stream lines: trajectories of massless particles in a “frozen” (steady) vector field
  - Path lines: trajectories of massless particles in (unsteady/time-varying) flow
  - Streak lines: trace of dye that is continuously released into (unsteady/time-varying) flow at a fixed position



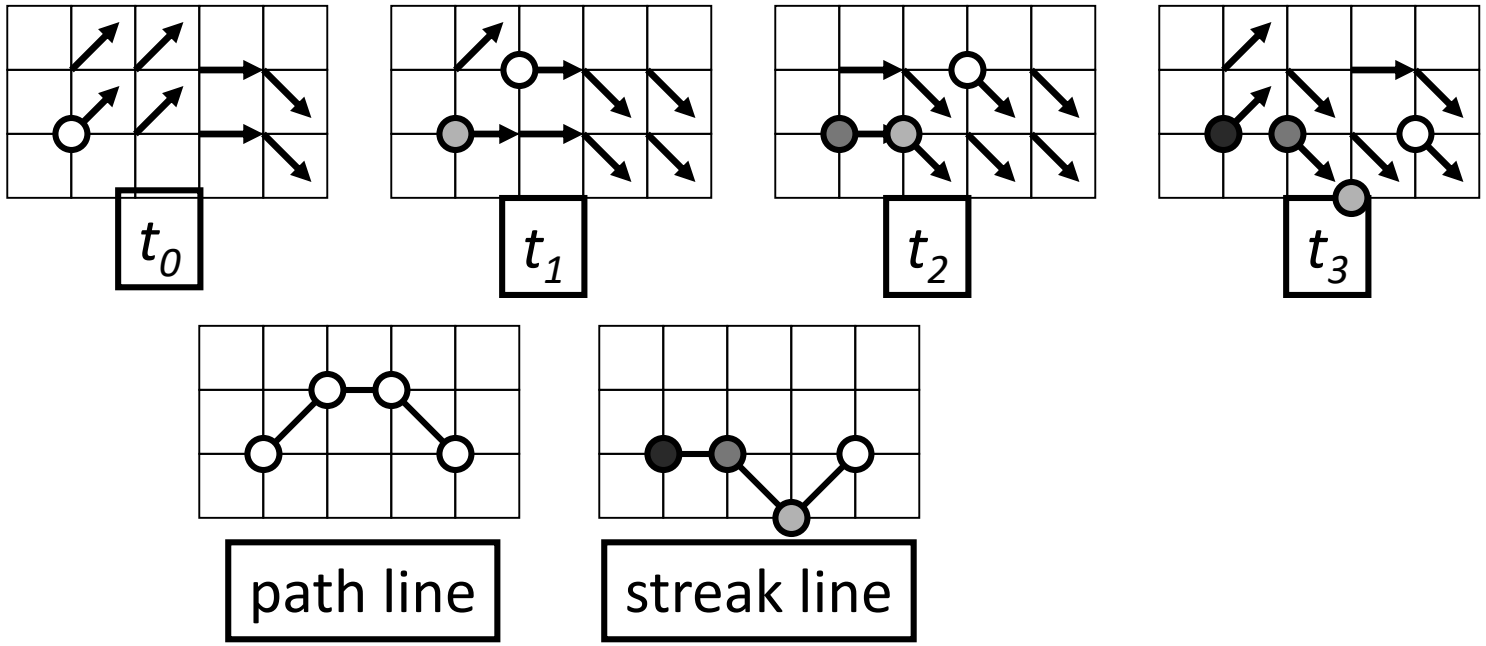
Characteristic Lines

- Path lines
  - Follow one particle through time and space



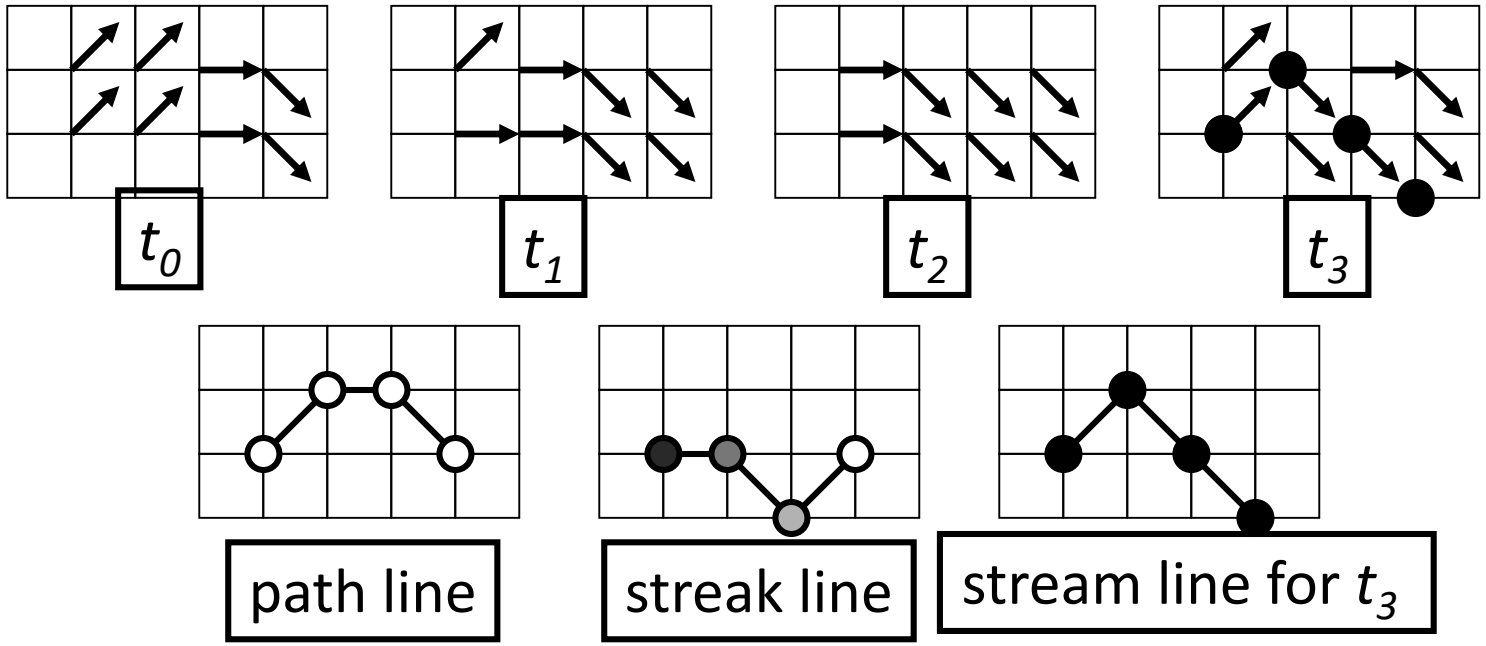
Characteristic Lines

- Streak lines
  - Connect all particles that started at the same seed point
    - A new particle is continuously injected at the same seed point
    - All existing particles are advected & connected (from youngest to oldest)



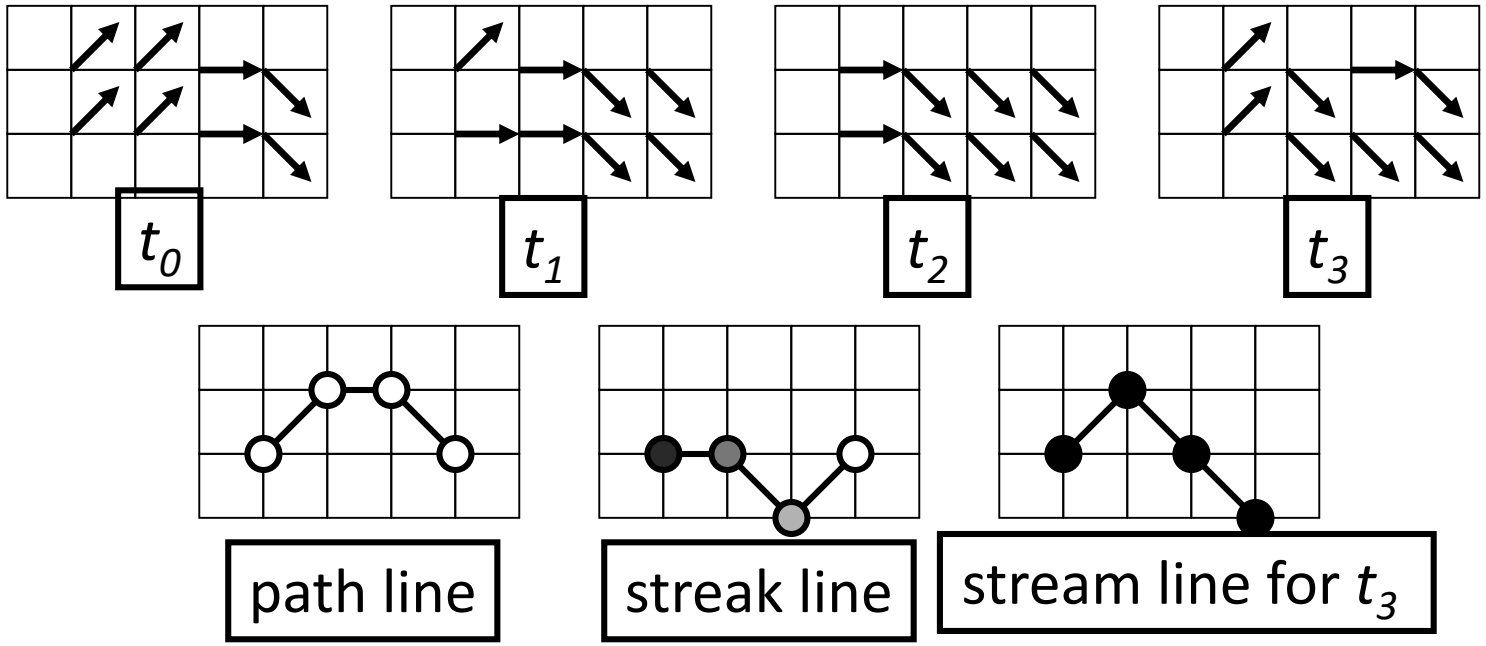
Characteristic Lines

- Stream lines
  - Trajectories of massless particles at one time step



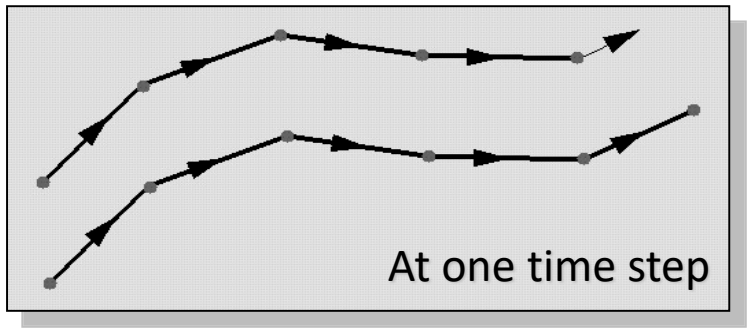
Characteristic Lines

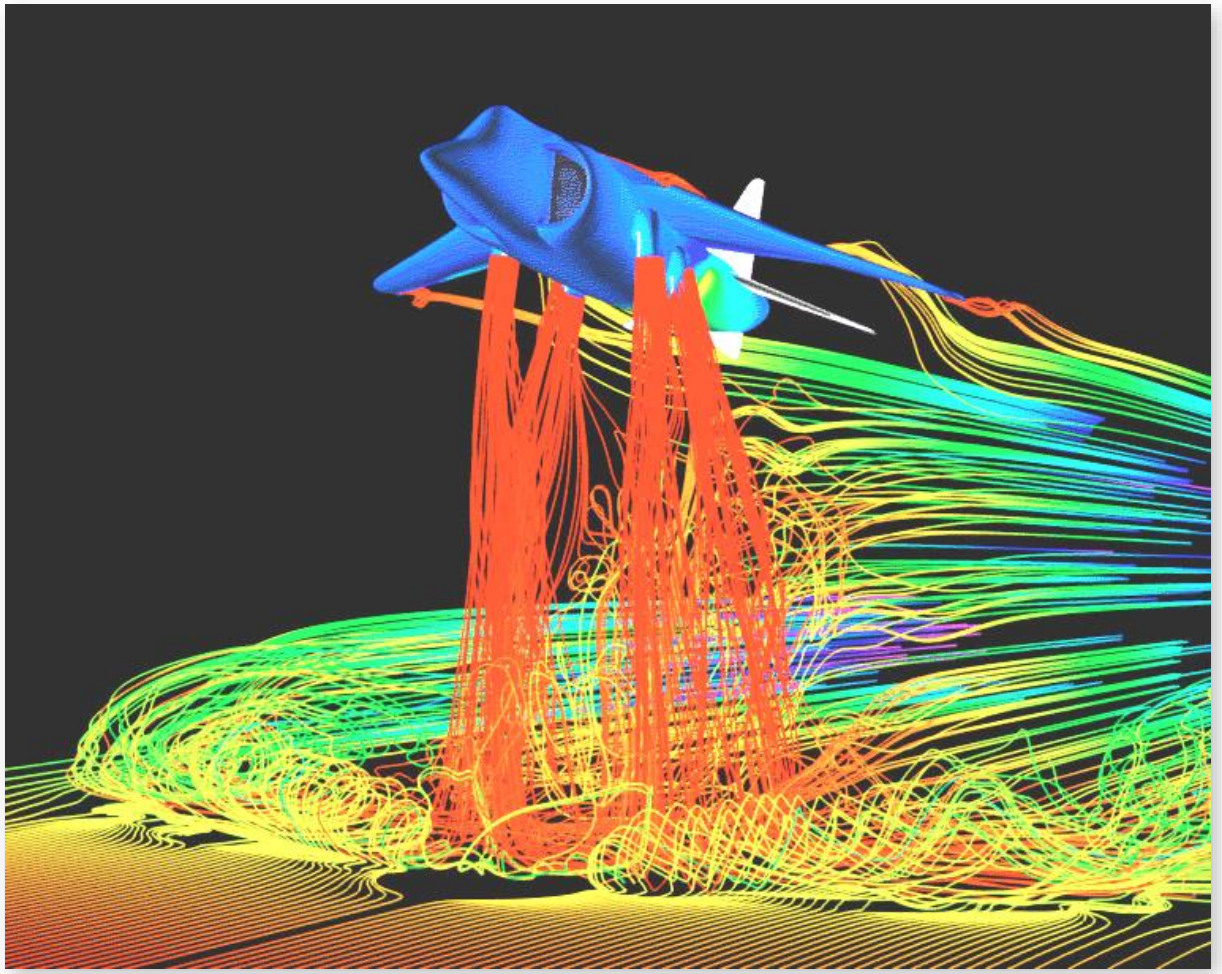
- Comparison of path lines, streak lines, and stream lines
  - Identical for steady flows



Mapping Based on Particle Tracing

- Stream lines

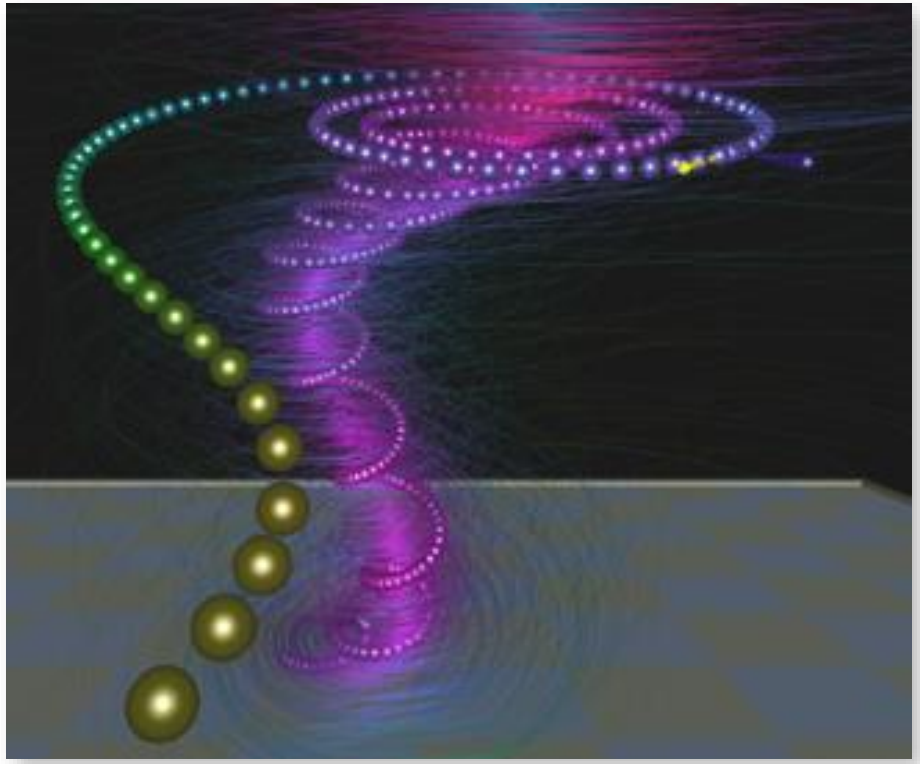




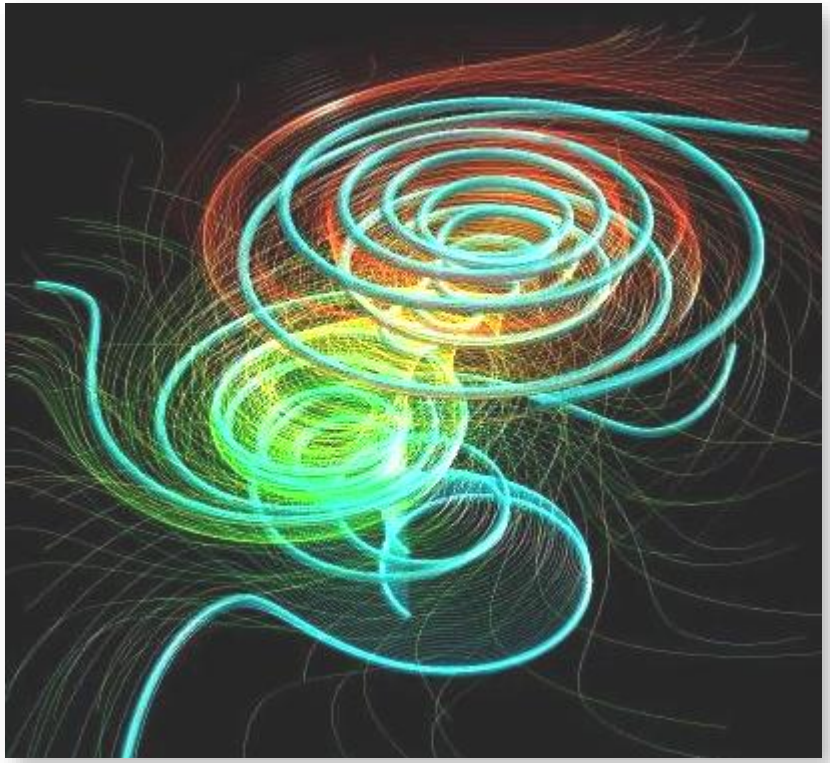
Simple colored stream lines

Mapping Based on Particle Tracing

- Stream lines



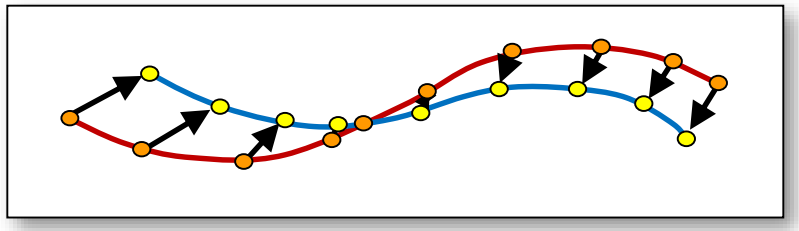
Stream balls & illuminated stream lines

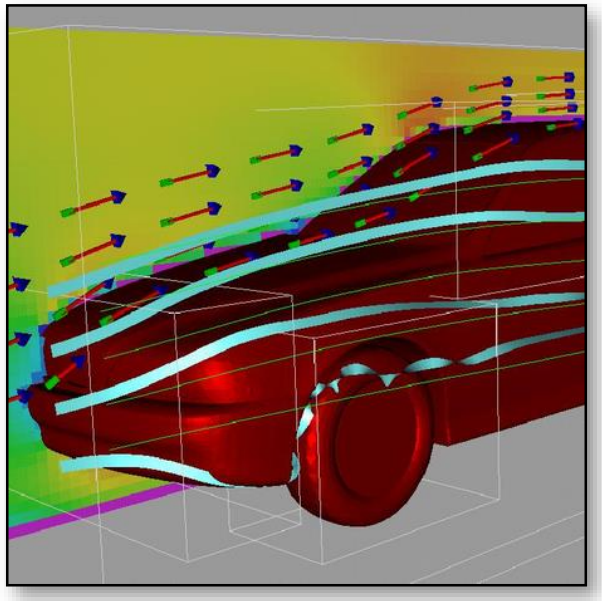


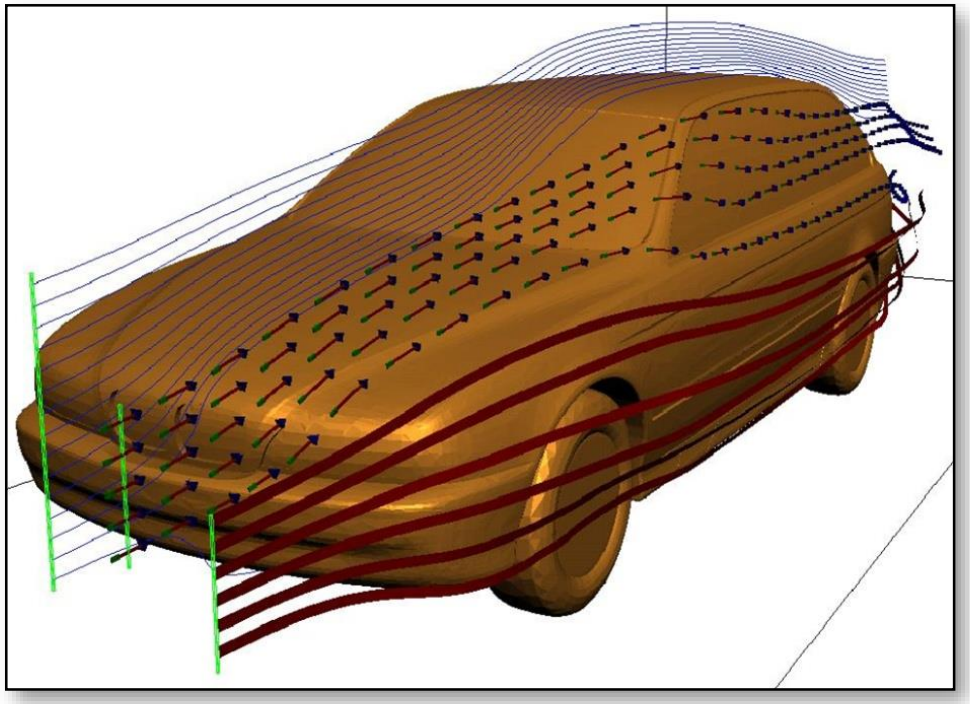
Stream balls & stream tubes

Mapping Based on Particle Tracing

- Stream ribbons (flow oriented)
  - We liked to see places where the flow twists (vortices)
  - Trace two close-by particles
    (keep distance constant)
  - Or rotate band according to curl

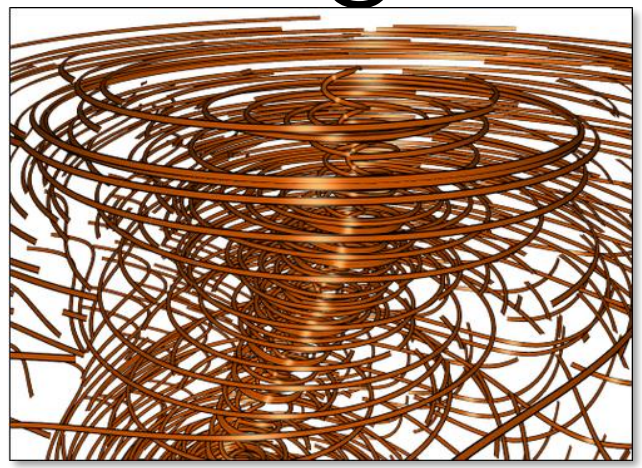


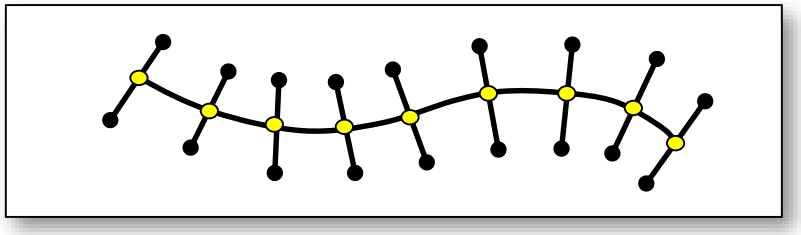


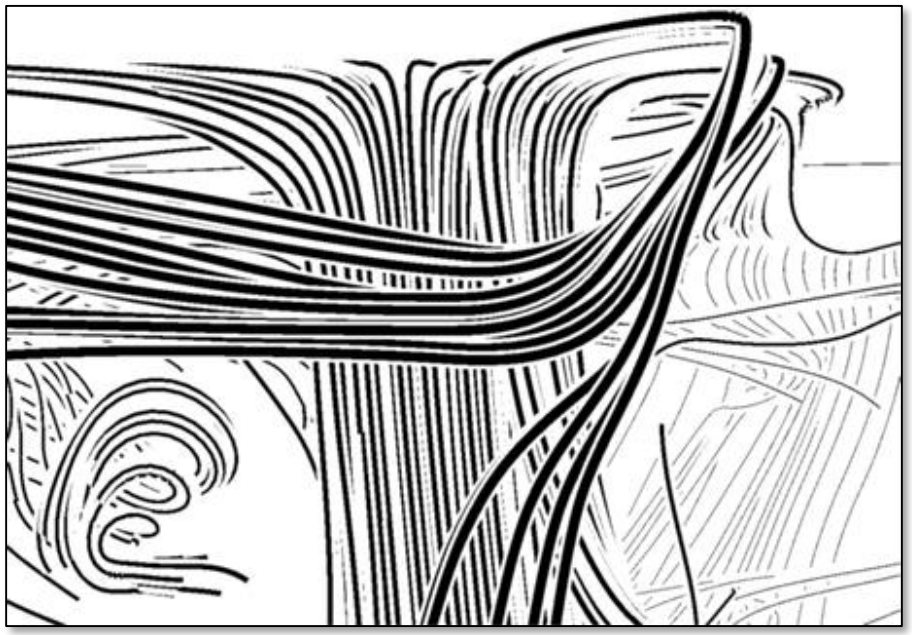


Mapping Based on Particle Tracing

- Stream ribbons (view oriented)
  - Trace one particle, span ribbon…
    - normal parallel to view direction
  - Well-suited for rendering with halos
    - Improves depth perception

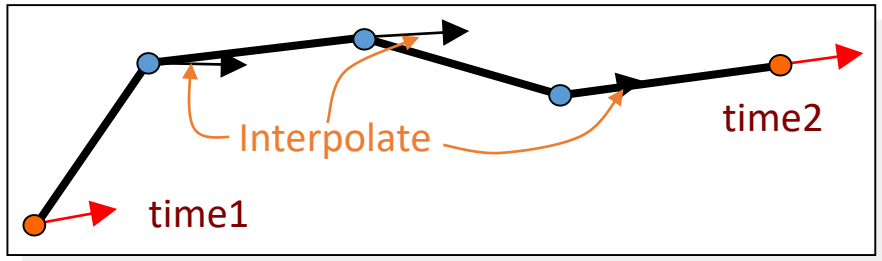


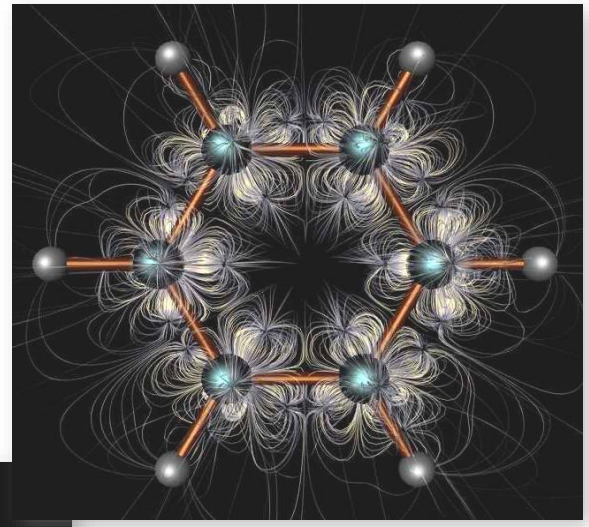


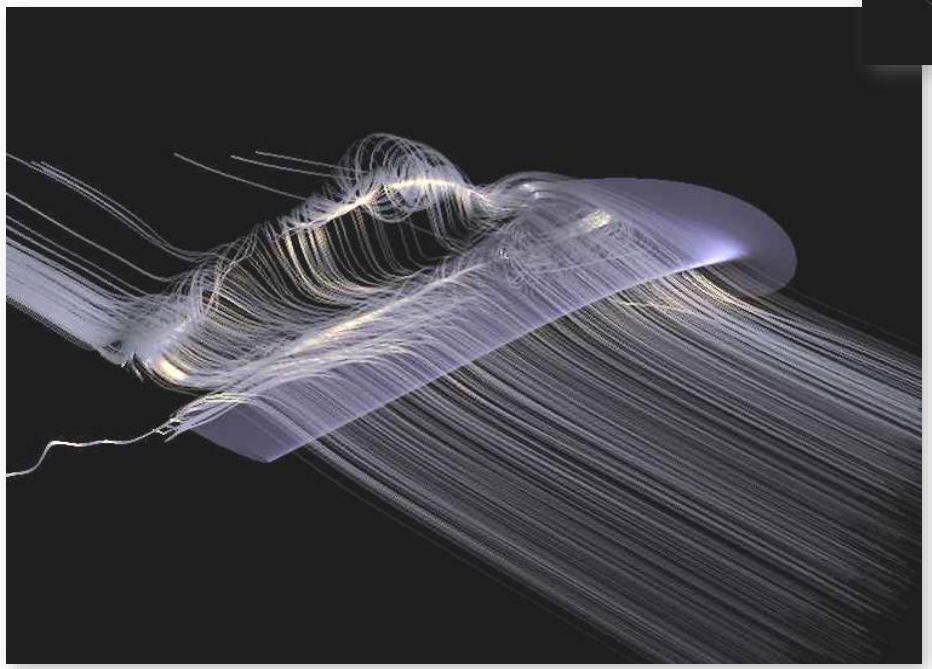


Mapping Based on Particle Tracing

- Path lines
  - Follow one particle through time and space
  - Illuminated 3D curves improves 3D perception

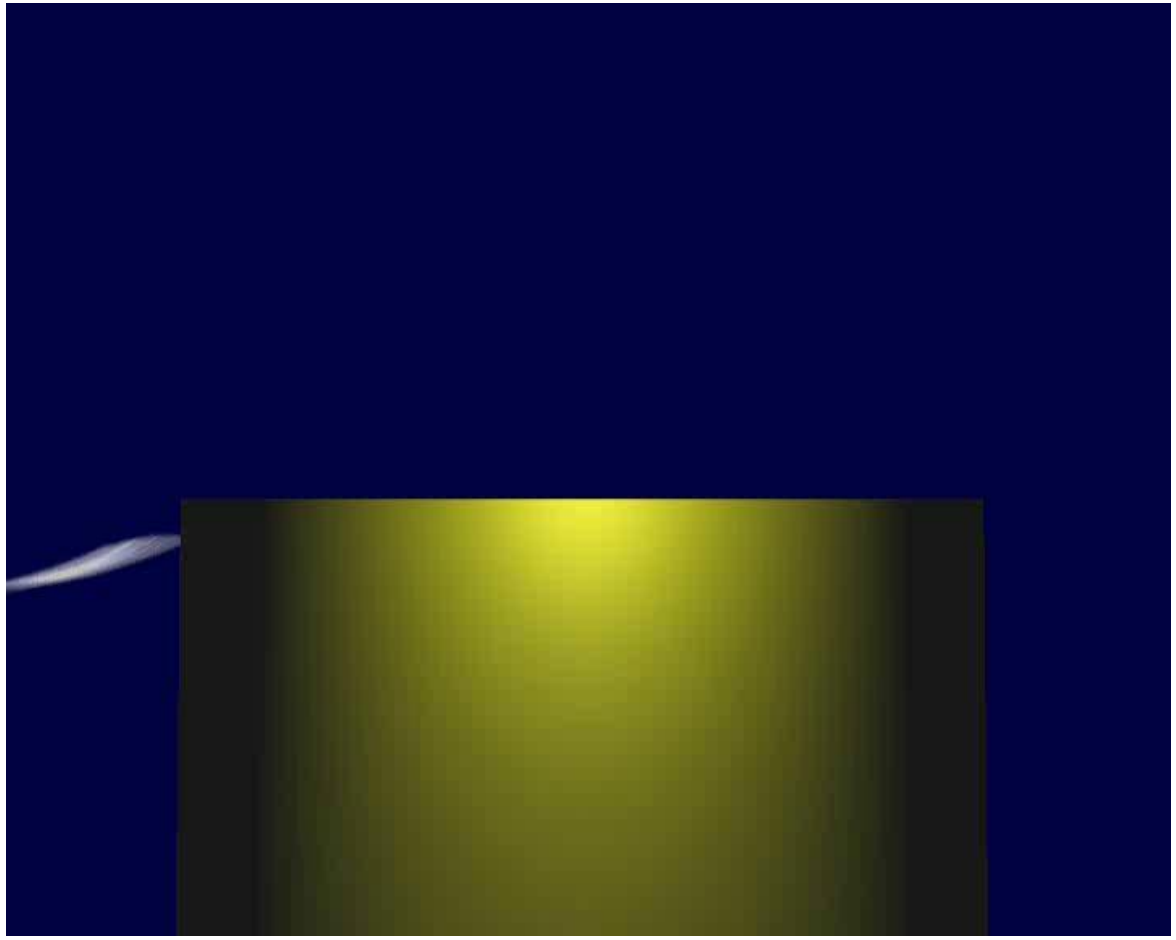






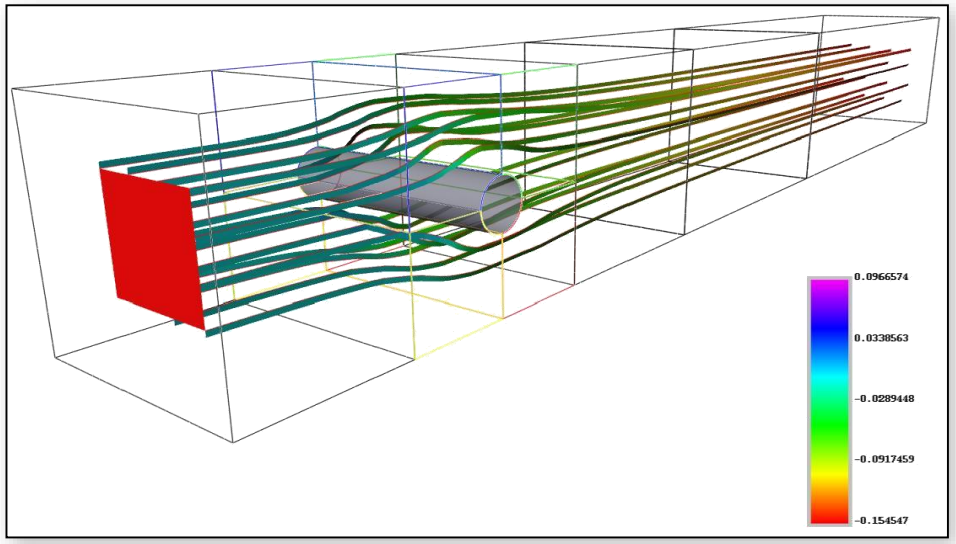
Mapping Based on Particle Tracing

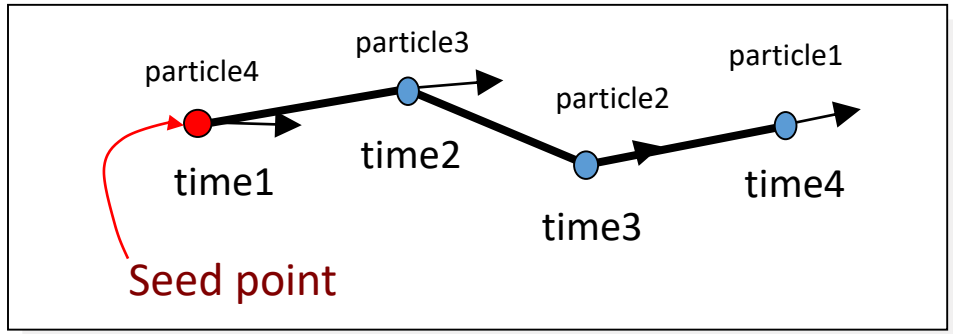
- Particle visualization in unsteady flow

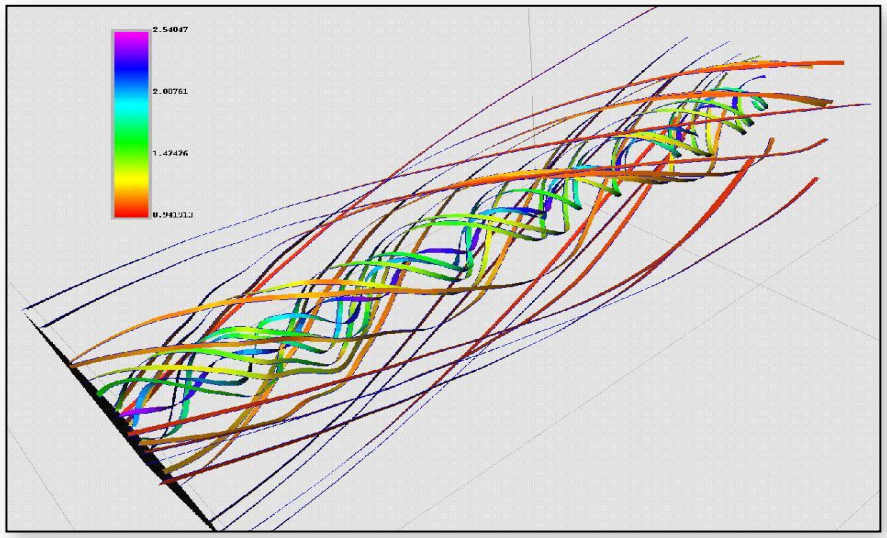


Mapping Based on Particle Tracing

- Streak lines
  - A new particle is continuously injected at the same seed point
  - Existing particles are advected & connected (from youngest to oldest)

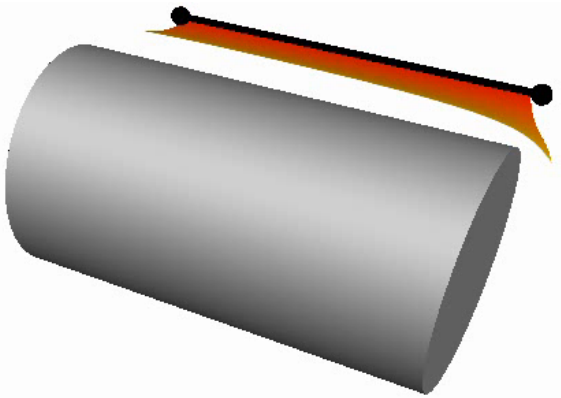






Mapping Based on Particle Tracing

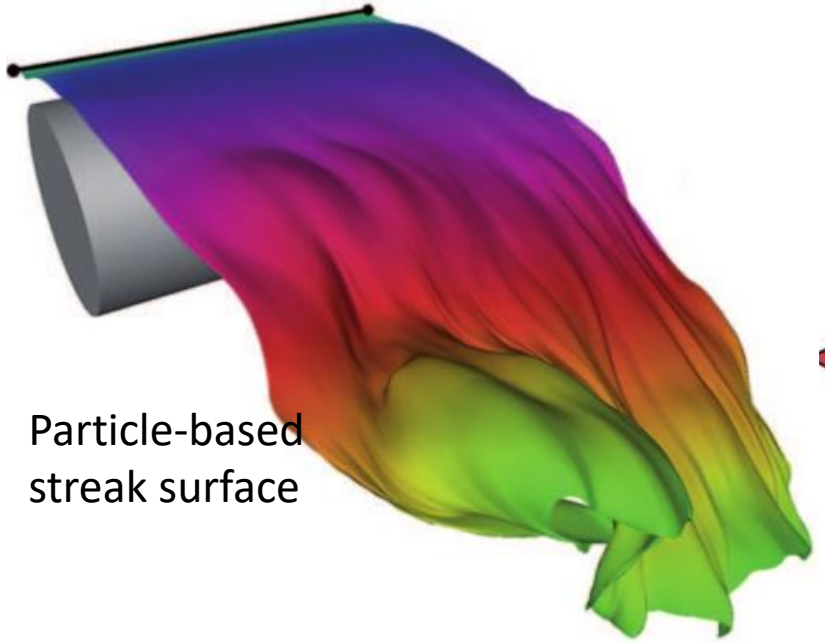
- Streak surface
  - Simultaneously release particles along a seeding structure (line) and connect all them to form a surface



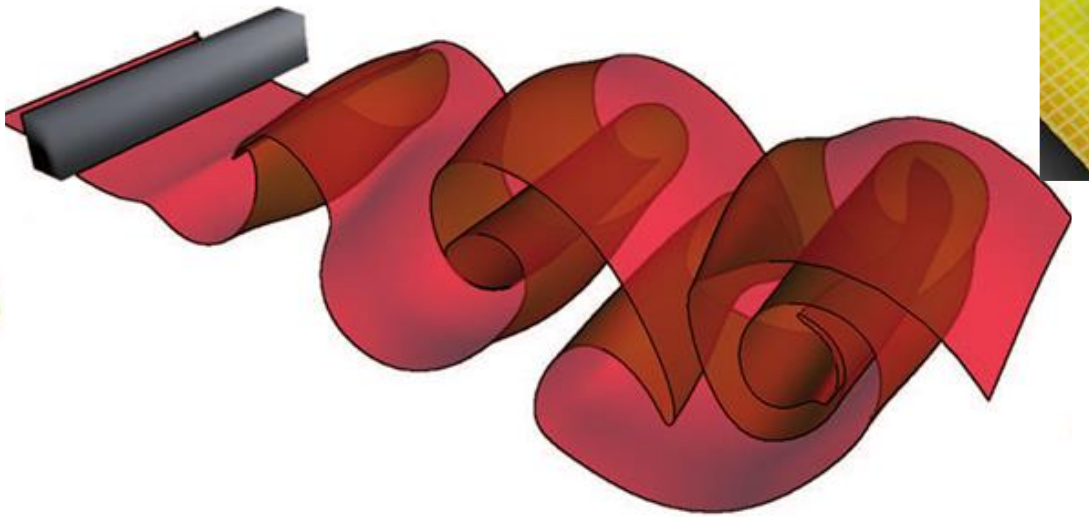
[Bürger et al. 09]

Mapping Based on Particle Tracing

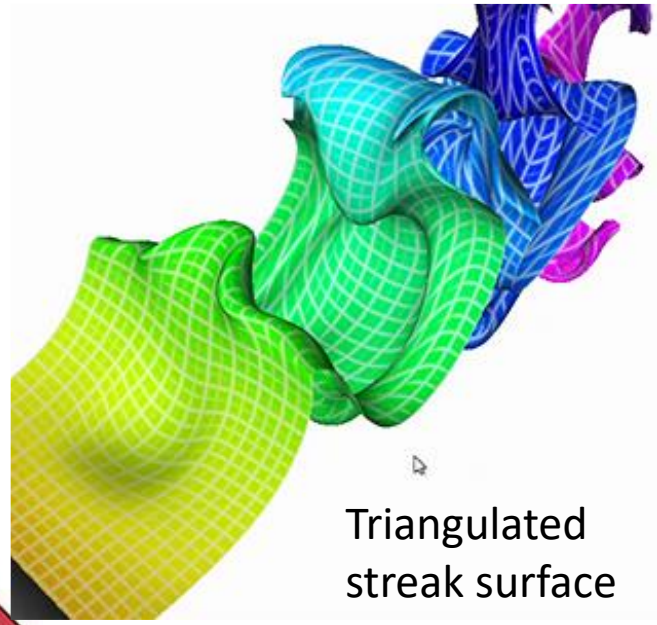
- Streak surface



Particle-based streak surface



Semi-transparent streak surface

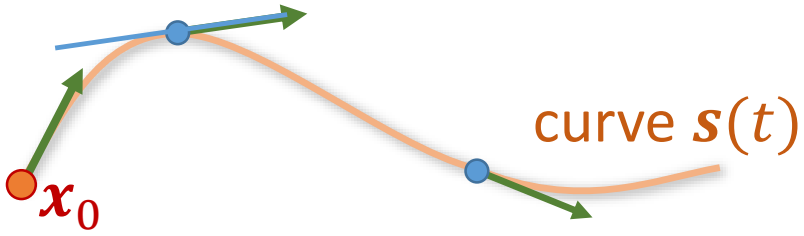


Triangulated streak surface

Characteristic Lines

- Characteristic lines are tangential to the flow
  - Means that line tangent (1st derivative) = vector field direction

$$\frac{d\boldsymbol{s}(t)}{dt} = \boldsymbol{v}(\boldsymbol{s}(t), t)$$



  - They are solutions to the initial value problem of an ordinary differential equation (ODE)

$$\frac{d\boldsymbol{s}(t)}{dt} = \boldsymbol{v}(\boldsymbol{s}(t), t)\,, \qquad \boldsymbol{s}(0) = \boldsymbol{x}_0$$

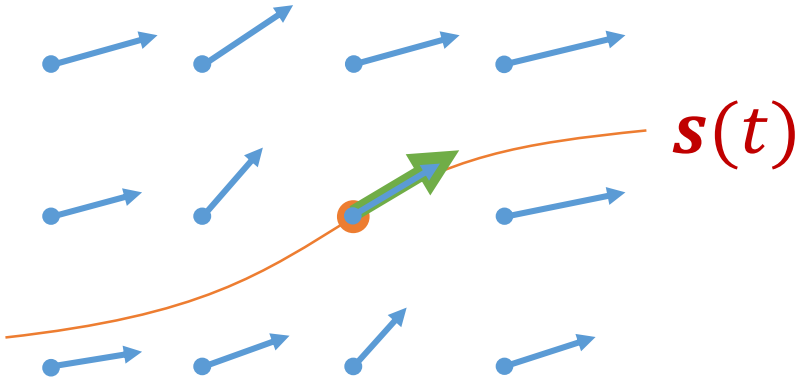
Ordinary differential equation (ODE)

Initial value (seed point $\boldsymbol{x}_0$)

Numerical Integration of ODEs

- Particle tracing problem – how to solve the differential equation:

$$\boldsymbol{s}(0) = \boldsymbol{x}_0\,, \qquad \dot{\boldsymbol{s}}(t) = \frac{d\boldsymbol{s}(t)}{dt} = \boldsymbol{v}(\boldsymbol{s}(t), t)$$



- What kind of numerical solver?

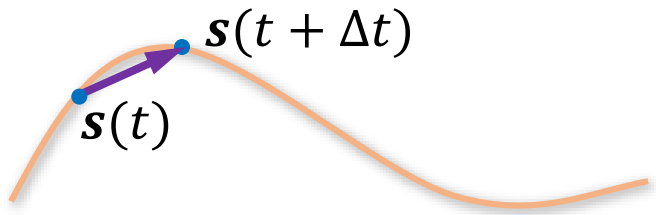
Numerical Integration of ODEs

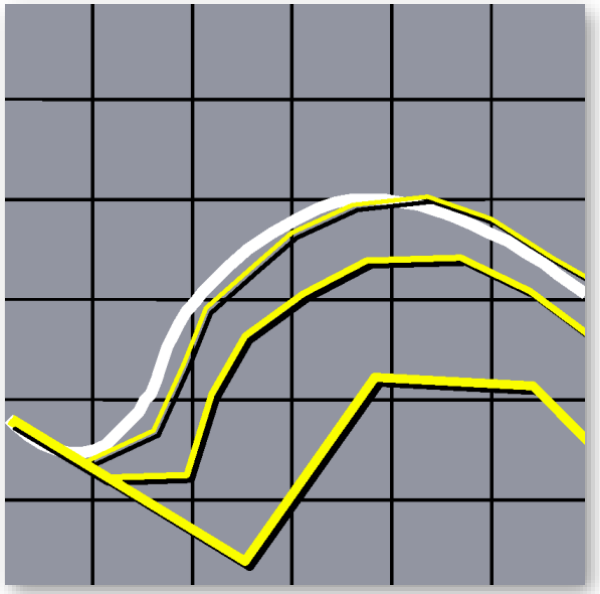
- Initial value problem for: $\dot{\boldsymbol{s}}(t) = \boldsymbol{v}(\boldsymbol{s}(t), t)$
- Simple approach: Euler method

$$\frac{\boldsymbol{s}(t + \Delta t) - \boldsymbol{s}(t)}{\Delta t} \approx \boldsymbol{v}(\boldsymbol{s}(t), t)$$

$$\Rightarrow \boldsymbol{s}(t + \Delta t) \approx \boldsymbol{s}(t) + \Delta t \cdot \boldsymbol{v}(\boldsymbol{s}(t), t)$$

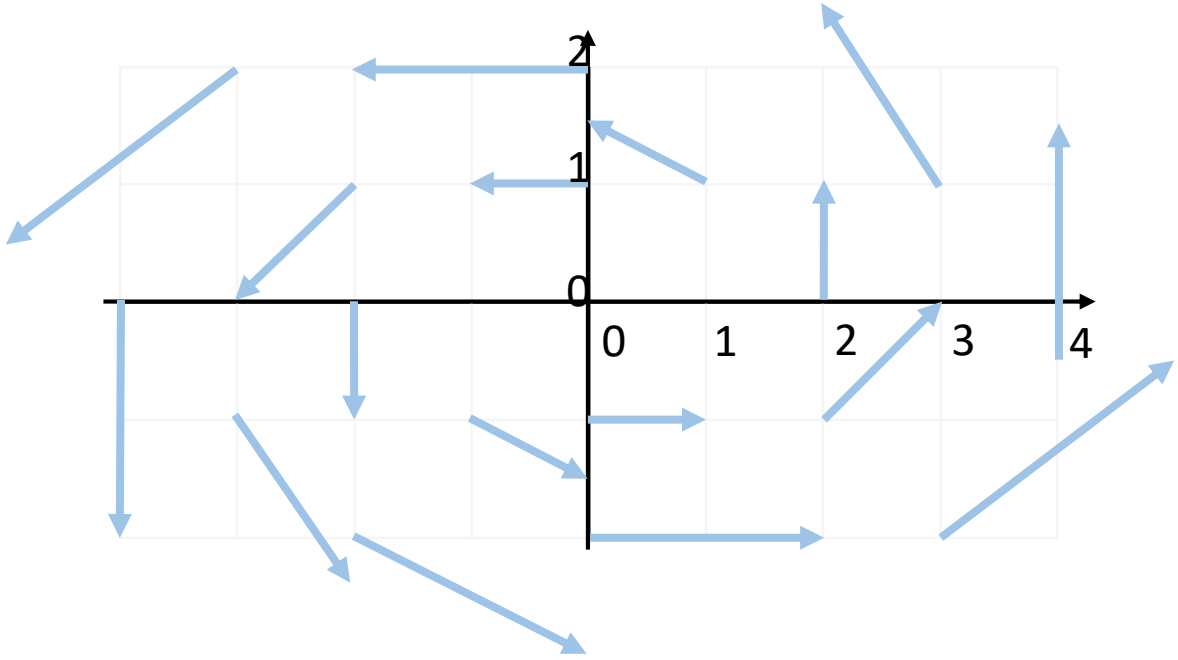
  - Based on Taylor expansion
  - First order method
  - Higher accuracy with smaller step size





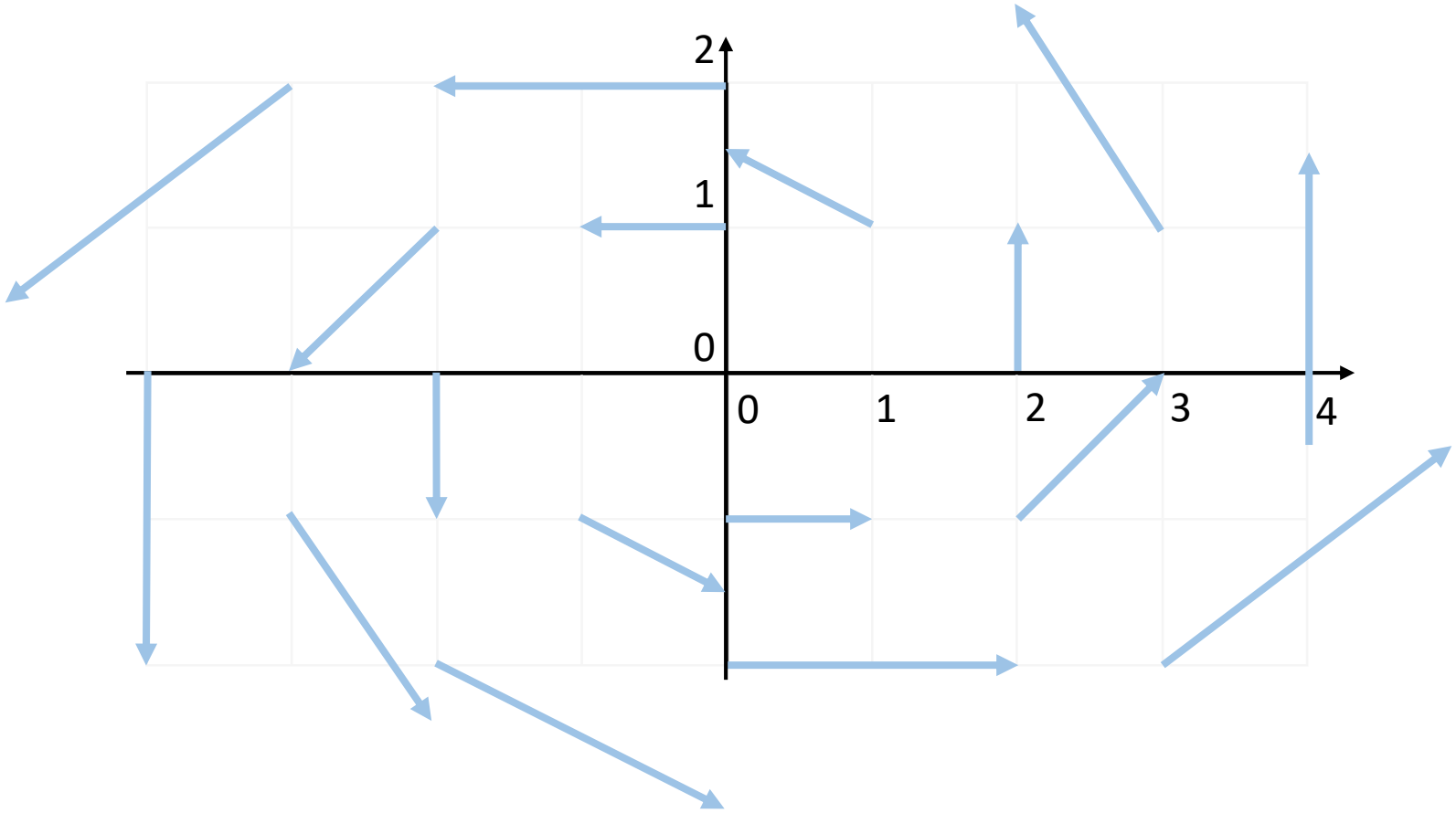
Numerical Integration of ODEs

- Example: 2D vector field $\boldsymbol{v}(x, y, t) = (-y, {}^{x}/_{2})^{T}$
  - Sample arrows shown
  - True solution are ellipses



Numerical Integration of ODEs

$$\boldsymbol{v}(x, y, t) = \left(-y, {}^{x}/_{2}\right)$$

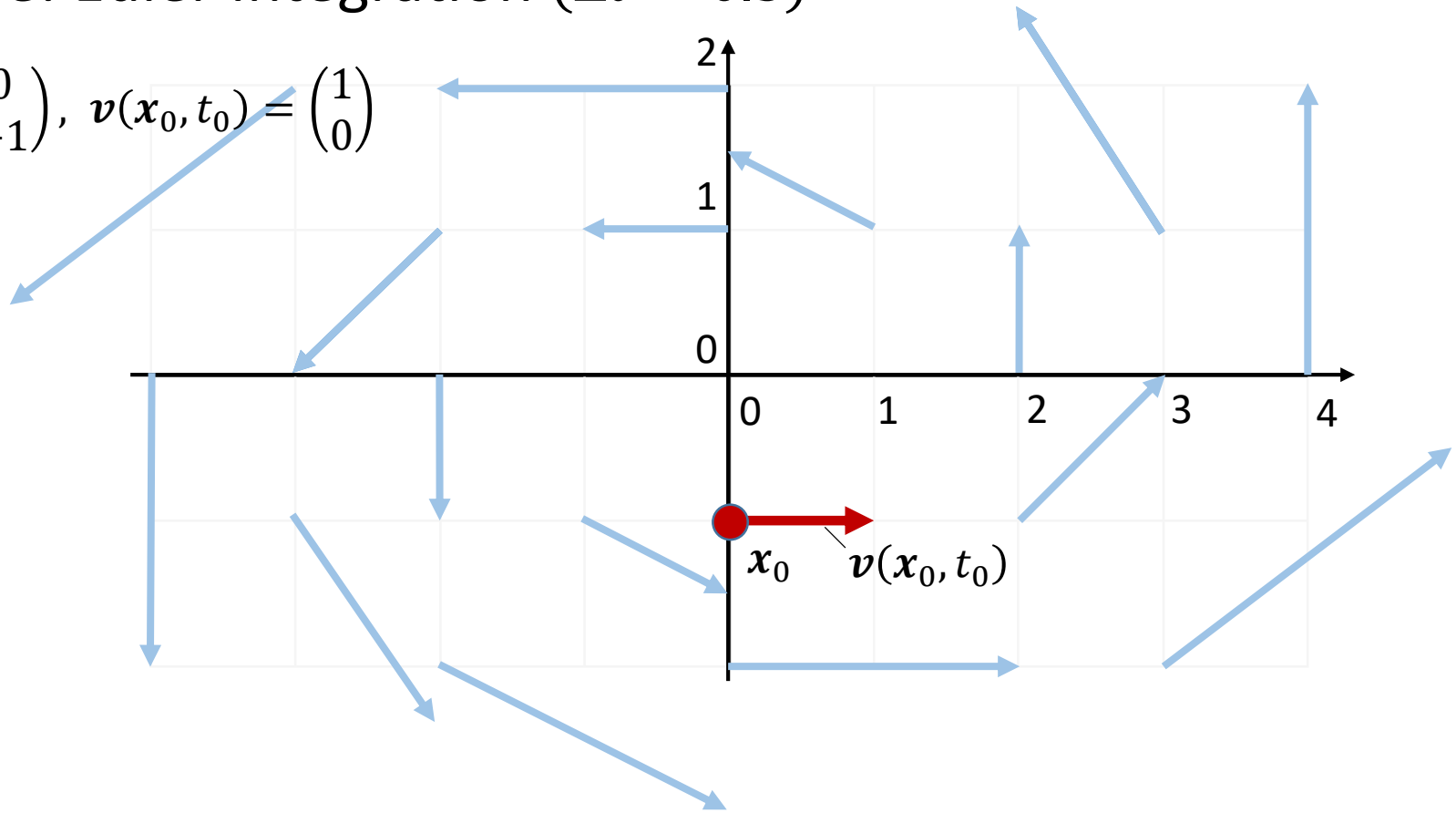


Numerical Integration of ODEs

$$\boldsymbol{v}(x, y, t) = \left(-y,\ {}^{x}/_{2}\right)$$

- Example: Euler Integration ($\Delta t = 0.5$)

Seed point: $\boldsymbol{x}_0 = \begin{pmatrix} 0 \\ -1 \end{pmatrix}$, $\boldsymbol{v}(\boldsymbol{x}_0, t_0) = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$



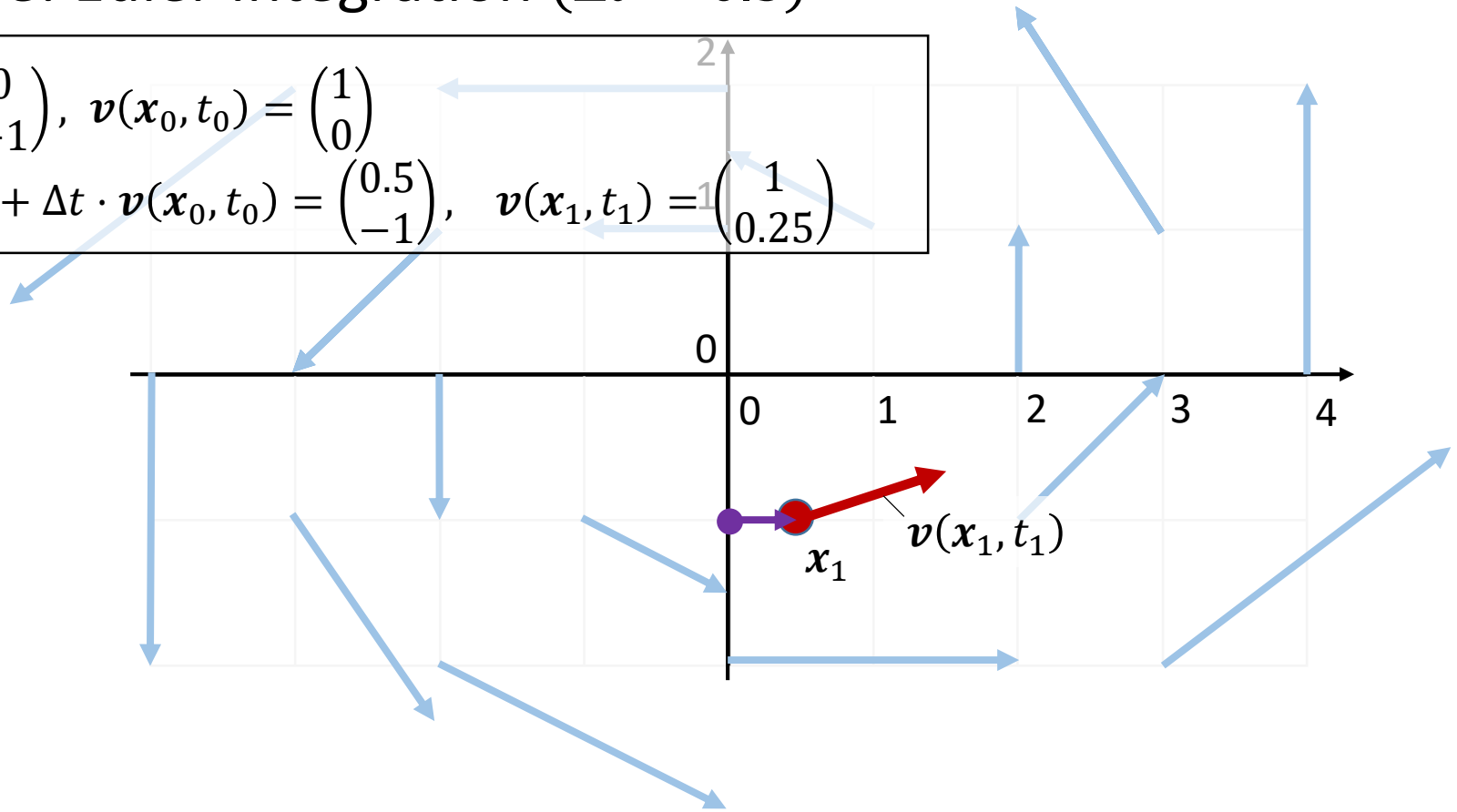
Numerical Integration of ODEs

$$\boldsymbol{v}(x, y, t) = \left(-y, \, {}^{x}/_{2}\right)$$

- Example: Euler Integration ($\Delta t = 0.5$)

Seed point: $\boldsymbol{x}_0 = \begin{pmatrix} 0 \\ -1 \end{pmatrix}$, $\boldsymbol{v}(\boldsymbol{x}_0, t_0) = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$

New point: $\boldsymbol{x}_1 = \boldsymbol{x}_0 + \Delta t \cdot \boldsymbol{v}(\boldsymbol{x}_0, t_0) = \begin{pmatrix} 0.5 \\ -1 \end{pmatrix}$, $\boldsymbol{v}(\boldsymbol{x}_1, t_1) = \begin{pmatrix} 1 \\ 0.25 \end{pmatrix}$



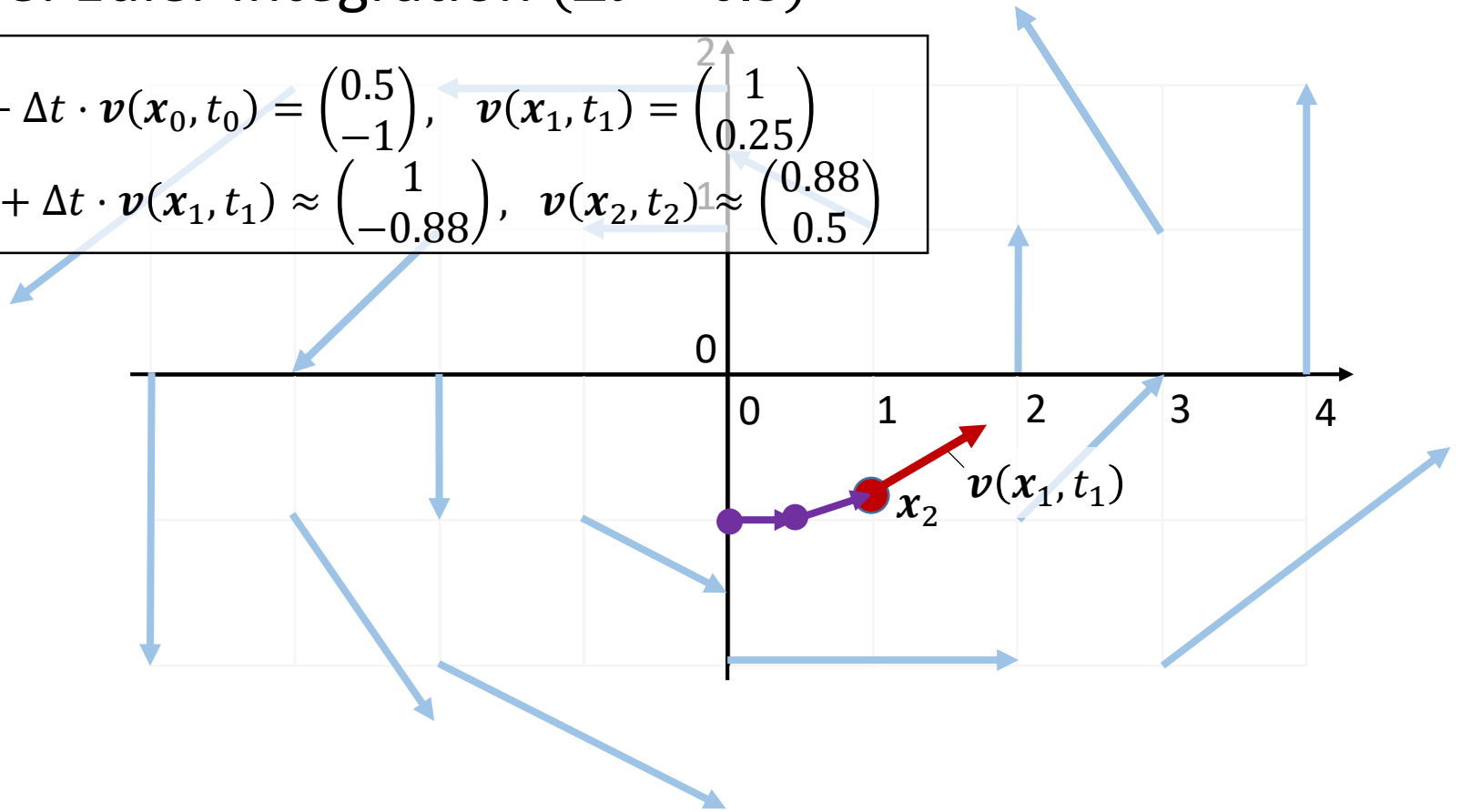
Numerical Integration of ODEs

$$\boldsymbol{v}(x, y, t) = \left(-y, \, {}^{x}/_{2}\right)$$

- Example: Euler Integration ($\Delta t = 0.5$)

Old point: $\boldsymbol{x}_1 = \boldsymbol{x}_0 + \Delta t \cdot \boldsymbol{v}(\boldsymbol{x}_0, t_0) = \begin{pmatrix} 0.5 \\ -1 \end{pmatrix}$, $\boldsymbol{v}(\boldsymbol{x}_1, t_1) = \begin{pmatrix} 1 \\ 0.25 \end{pmatrix}$

New point: $\boldsymbol{x}_2 = \boldsymbol{x}_1 + \Delta t \cdot \boldsymbol{v}(\boldsymbol{x}_1, t_1) \approx \begin{pmatrix} 1 \\ -0.88 \end{pmatrix}$, $\boldsymbol{v}(\boldsymbol{x}_2, t_2) \approx \begin{pmatrix} 0.88 \\ 0.5 \end{pmatrix}$



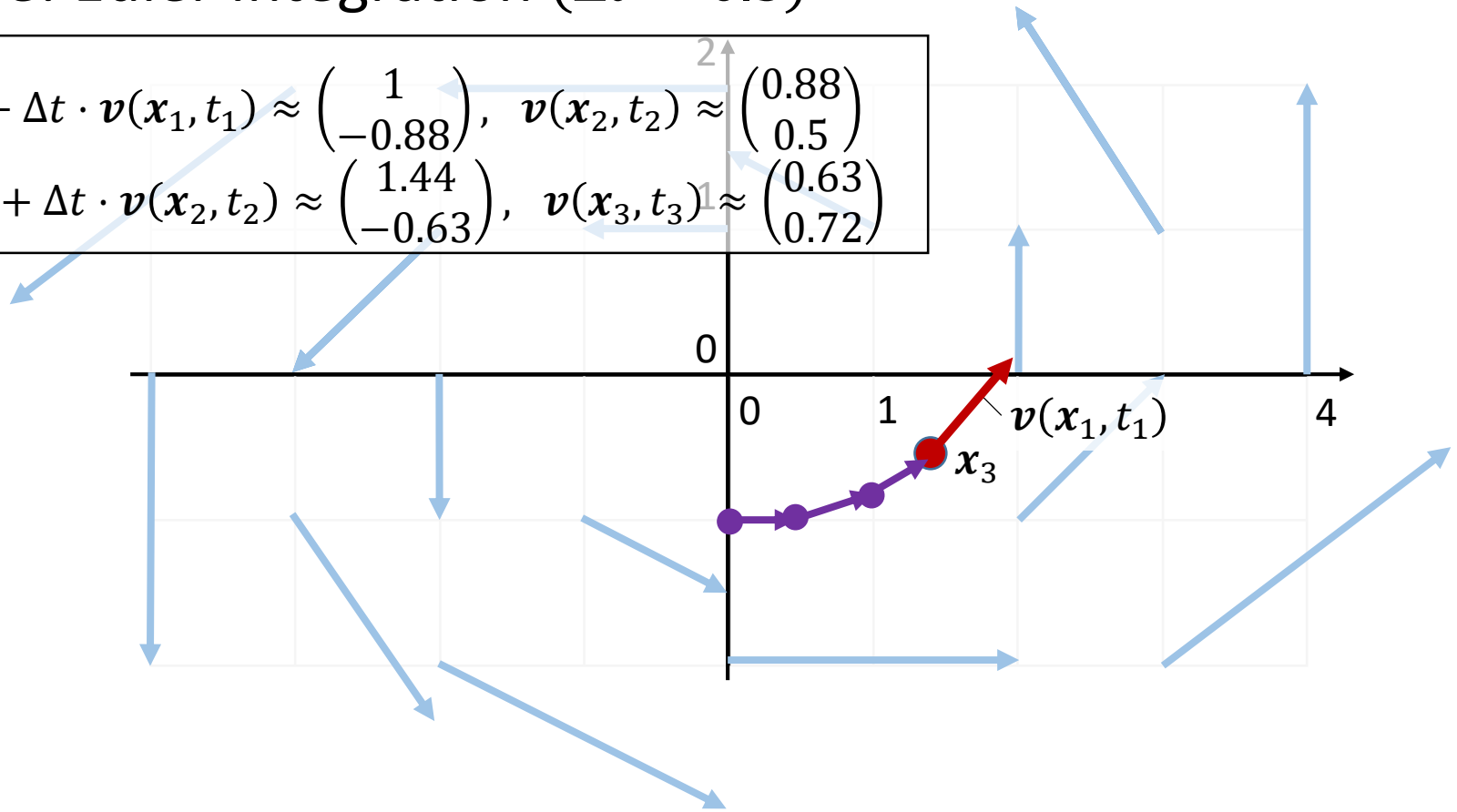
Numerical Integration of ODEs

$$\boldsymbol{v}(x, y, t) = \left(-y, {}^{x}/_{2}\right)$$

- Example: Euler Integration ($\Delta t = 0.5$)

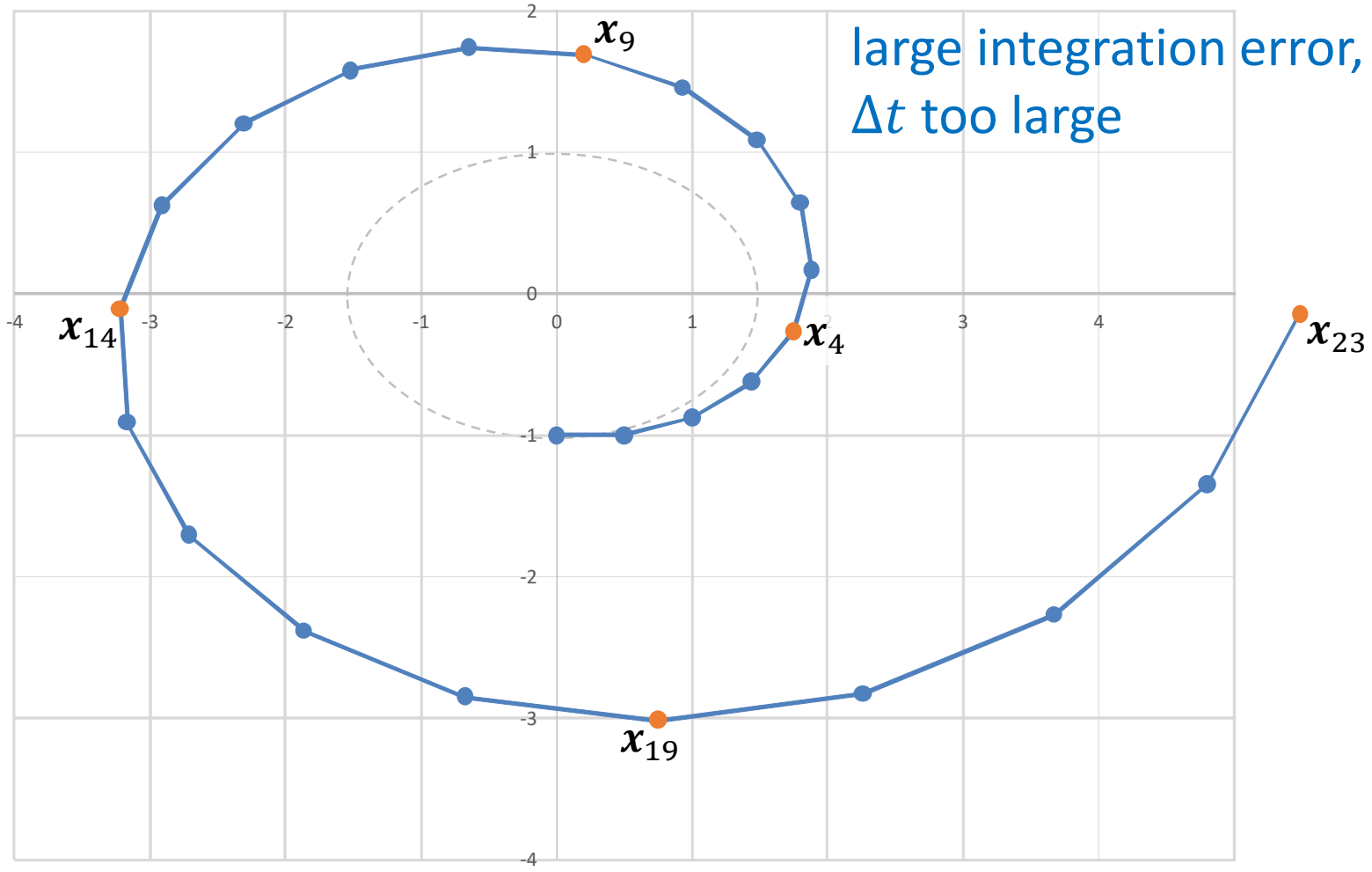
Old point: $\boldsymbol{x}_2 = \boldsymbol{x}_1 + \Delta t \cdot \boldsymbol{v}(\boldsymbol{x}_1, t_1) \approx \begin{pmatrix} 1 \\ -0.88 \end{pmatrix}$, $\boldsymbol{v}(\boldsymbol{x}_2, t_2) \approx \begin{pmatrix} 0.88 \\ 0.5 \end{pmatrix}$

New point: $\boldsymbol{x}_3 = \boldsymbol{x}_2 + \Delta t \cdot \boldsymbol{v}(\boldsymbol{x}_2, t_2) \approx \begin{pmatrix} 1.44 \\ -0.63 \end{pmatrix}$, $\boldsymbol{v}(\boldsymbol{x}_3, t_3) \approx \begin{pmatrix} 0.63 \\ 0.72 \end{pmatrix}$



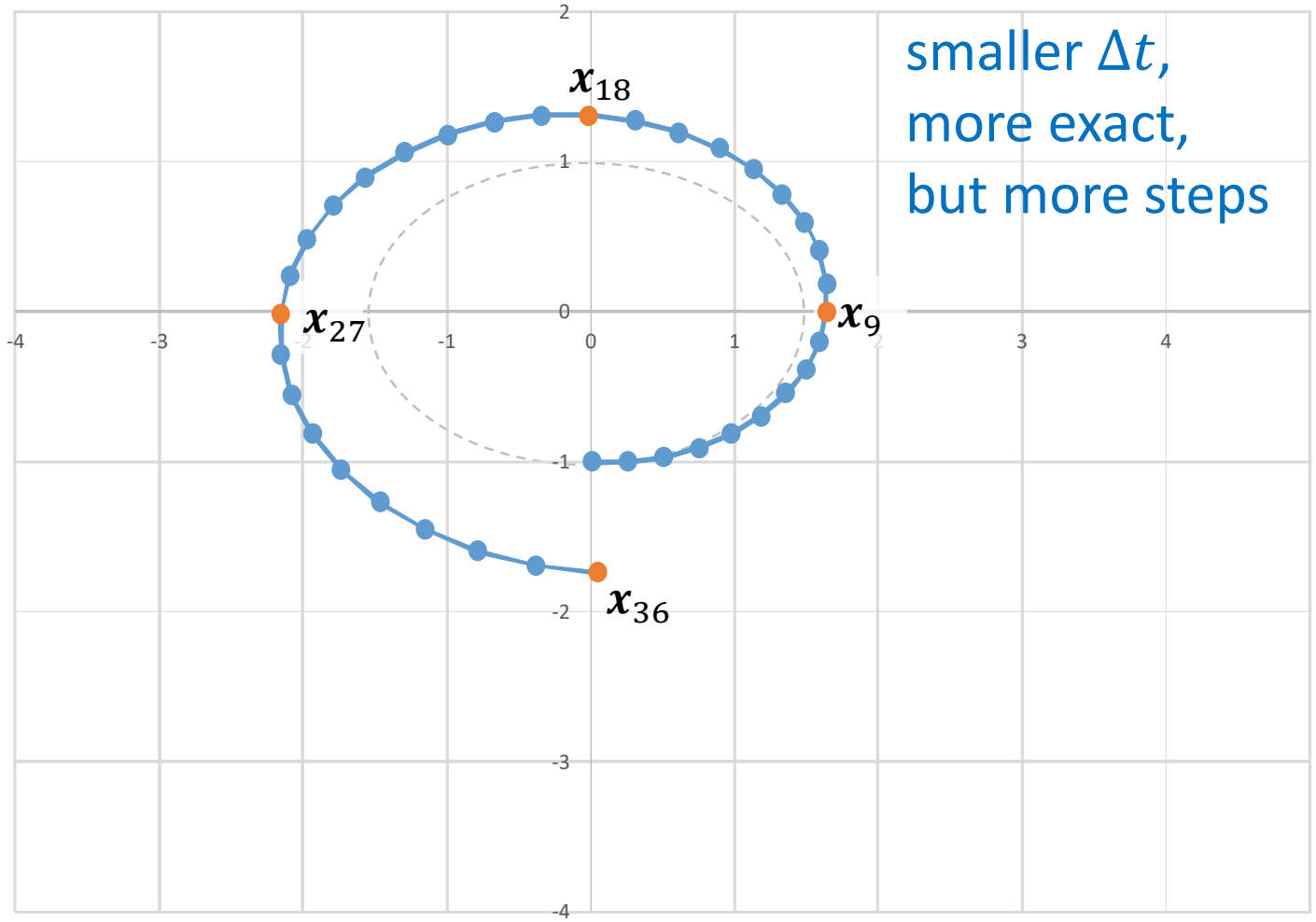
Numerical Integration of ODEs

Euler Integration after 23 steps ($\Delta t = 0.5$)



Numerical Integration of ODEs

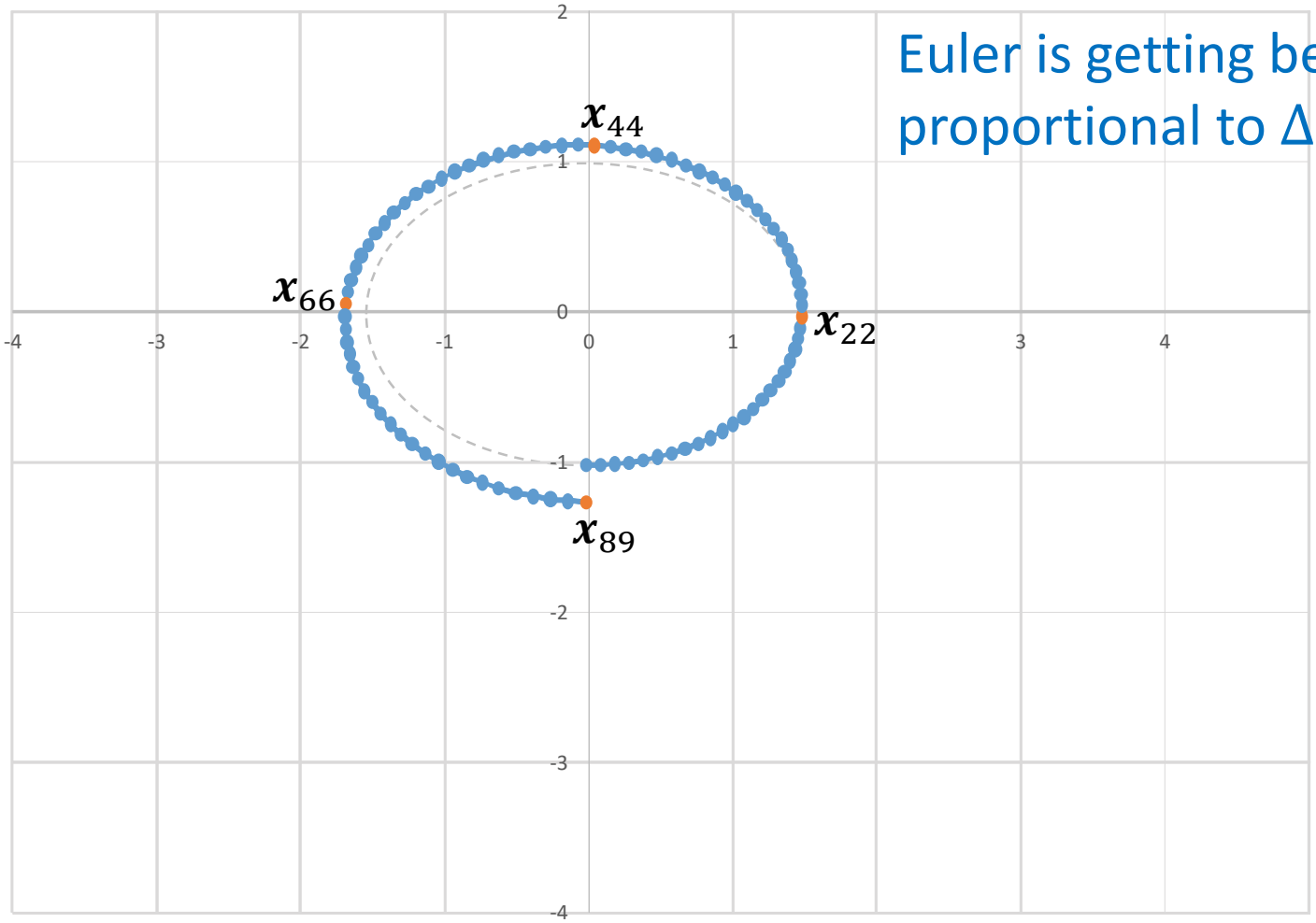
Euler Integration after 36 steps ($\Delta t = 0.25$)



smaller Δt, more exact, but more steps

Numerical Integration of ODEs

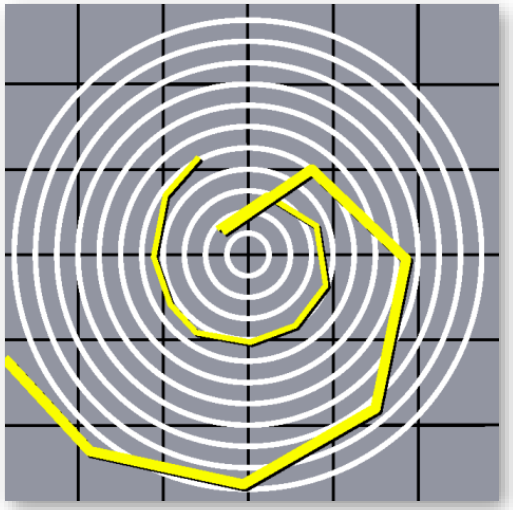
Euler Integration after 89 steps ($\Delta t = 0.1$)



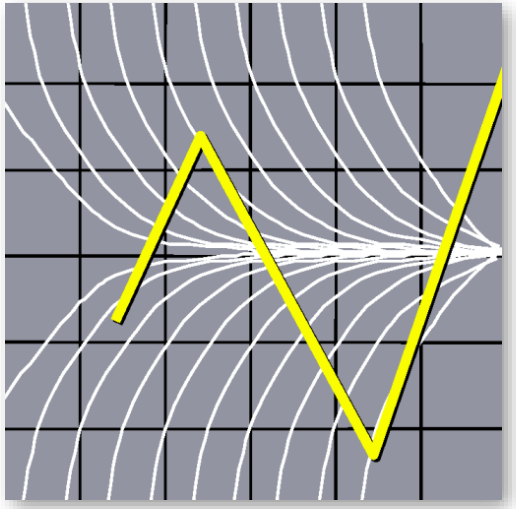
Euler is getting better proportional to Δt

Numerical Integration of ODEs

- Problem of Euler method
  - Inaccurate



  - Unstable



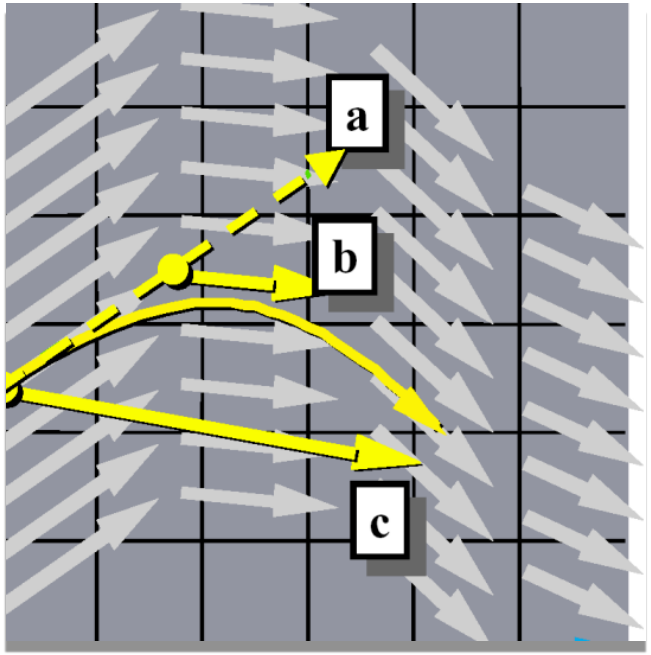
→ Higher order integration possible

Numerical Integration of ODEs

- Midpoint method (Runge-Kutta 2nd order)
  a. Euler step
     $\Delta \boldsymbol{v} = \Delta t \cdot \boldsymbol{v}(\boldsymbol{s}(t), t)$

  b. Evaluation of $\boldsymbol{v}$ at midpoint
     $\boldsymbol{v}_{\text{mid}} = \boldsymbol{v}(\boldsymbol{s}(t) + \frac{\Delta \boldsymbol{v}}{2}, t + \frac{\Delta t}{2})$

  c. Complete step with vector at midpoint
     $\boldsymbol{s}(t + \Delta t) = \boldsymbol{s}(t) + \Delta t \cdot \boldsymbol{v}_{\text{mid}}$



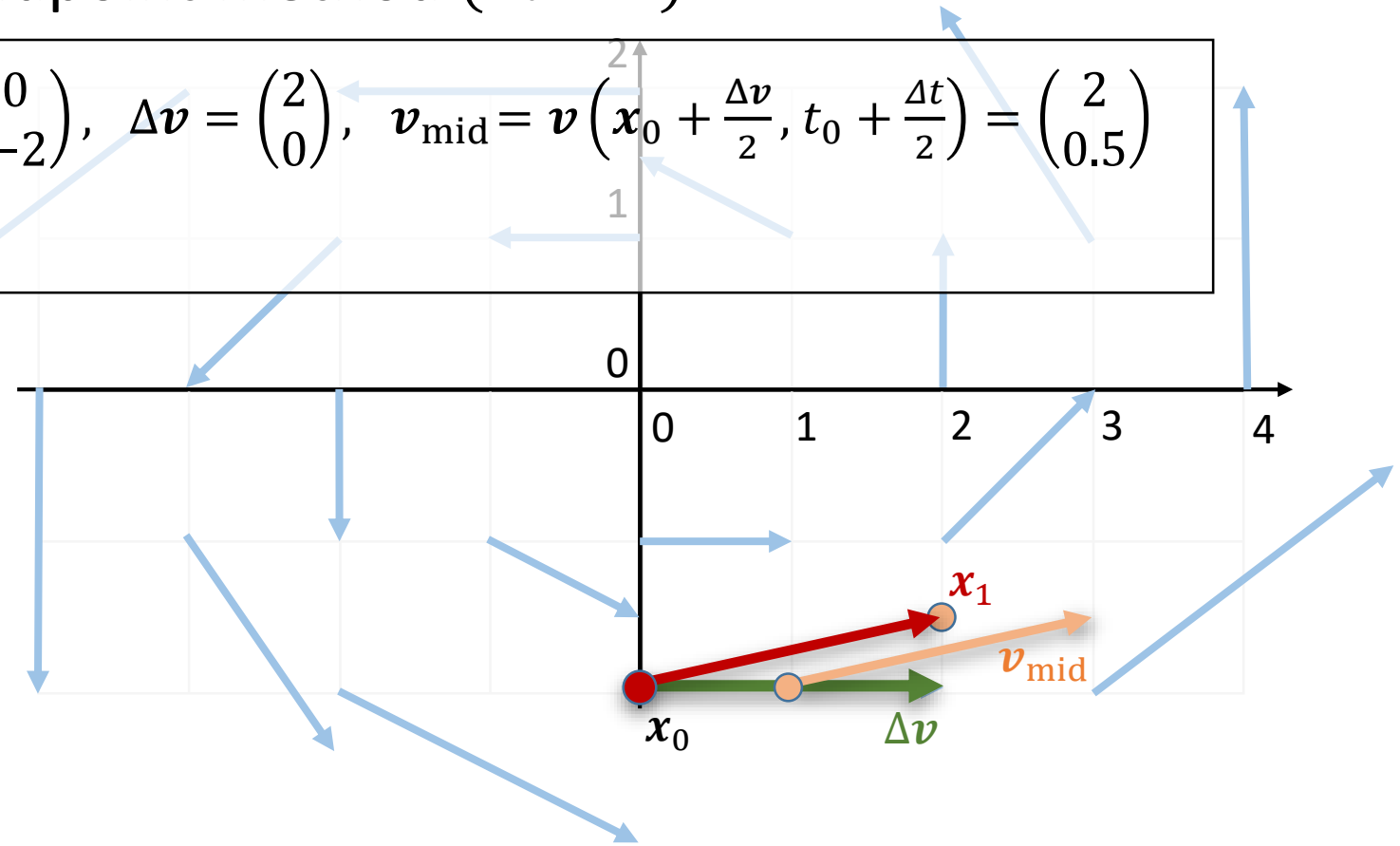
More accurate results with midpoint method

Numerical Integration of ODEs

$$\boldsymbol{v}(x, y, t) = \left(-y,\ {}^{x}/_{2}\right)$$

- Example: Midpoint method ($\Delta t = 1$)

Seed point: $\boldsymbol{x}_0 = \begin{pmatrix} 0 \\ -2 \end{pmatrix}$, $\Delta \boldsymbol{v} = \begin{pmatrix} 2 \\ 0 \end{pmatrix}$, $\boldsymbol{v}_{\text{mid}} = \boldsymbol{v}\left(\boldsymbol{x}_0 + \frac{\Delta \boldsymbol{v}}{2}, t_0 + \frac{\Delta t}{2}\right) = \begin{pmatrix} 2 \\ 0.5 \end{pmatrix}$



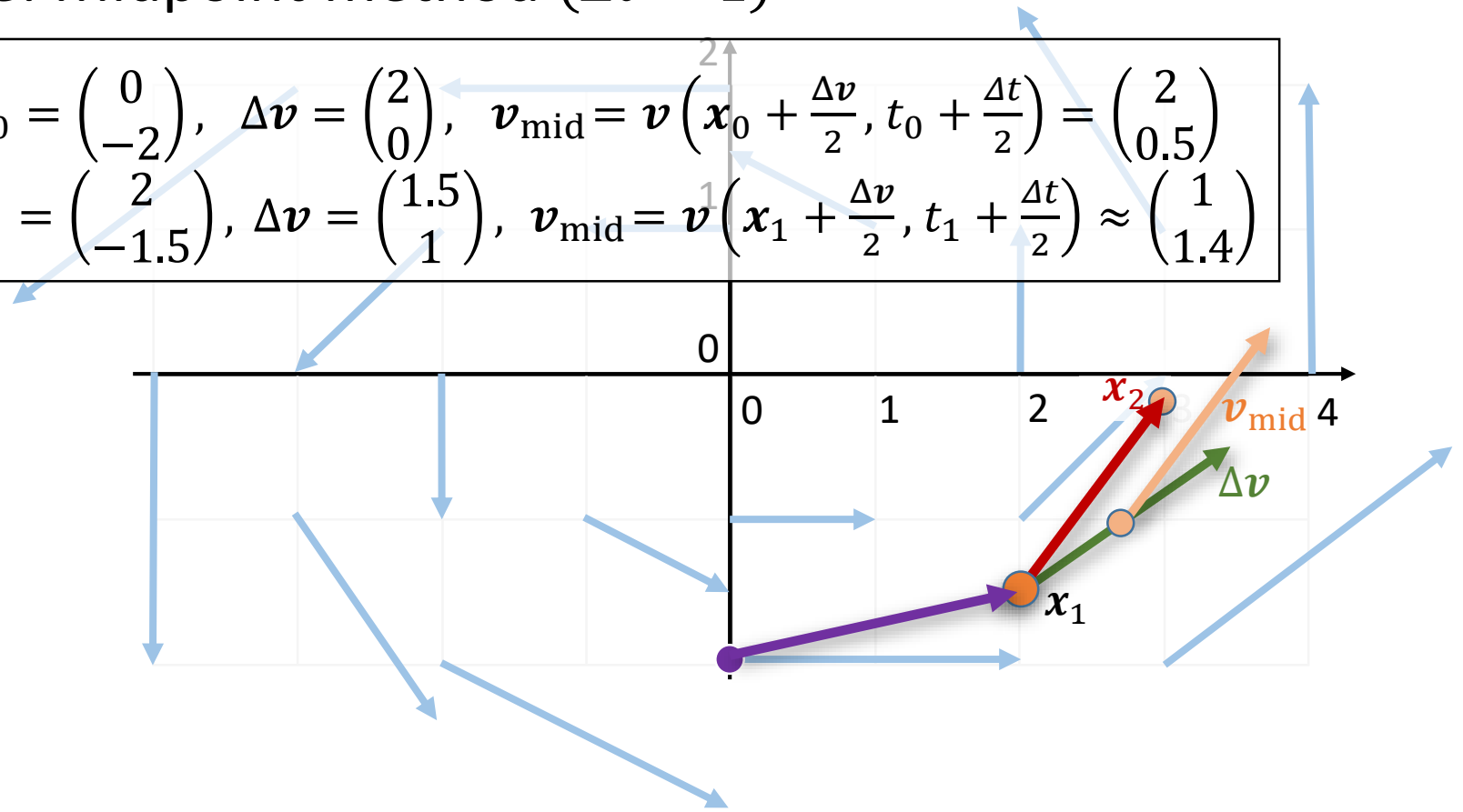
Numerical Integration of ODEs

$$\boldsymbol{v}(x, y, t) = \left(-y,\ {}^{x}/_{2}\right)$$

- Example: Midpoint method ($\Delta t = 1$)

Seed point: $\boldsymbol{x}_0 = \begin{pmatrix} 0 \\ -2 \end{pmatrix}$, $\Delta \boldsymbol{v} = \begin{pmatrix} 2 \\ 0 \end{pmatrix}$, $\boldsymbol{v}_{\text{mid}} = \boldsymbol{v}\left(\boldsymbol{x}_0 + \frac{\Delta \boldsymbol{v}}{2}, t_0 + \frac{\Delta t}{2}\right) = \begin{pmatrix} 2 \\ 0.5 \end{pmatrix}$

New point: $\boldsymbol{x}_1 = \begin{pmatrix} 2 \\ -1.5 \end{pmatrix}$, $\Delta \boldsymbol{v} = \begin{pmatrix} 1.5 \\ 1 \end{pmatrix}$, $\boldsymbol{v}_{\text{mid}} = \boldsymbol{v}\left(\boldsymbol{x}_1 + \frac{\Delta \boldsymbol{v}}{2}, t_1 + \frac{\Delta t}{2}\right) \approx \begin{pmatrix} 1 \\ 1.4 \end{pmatrix}$

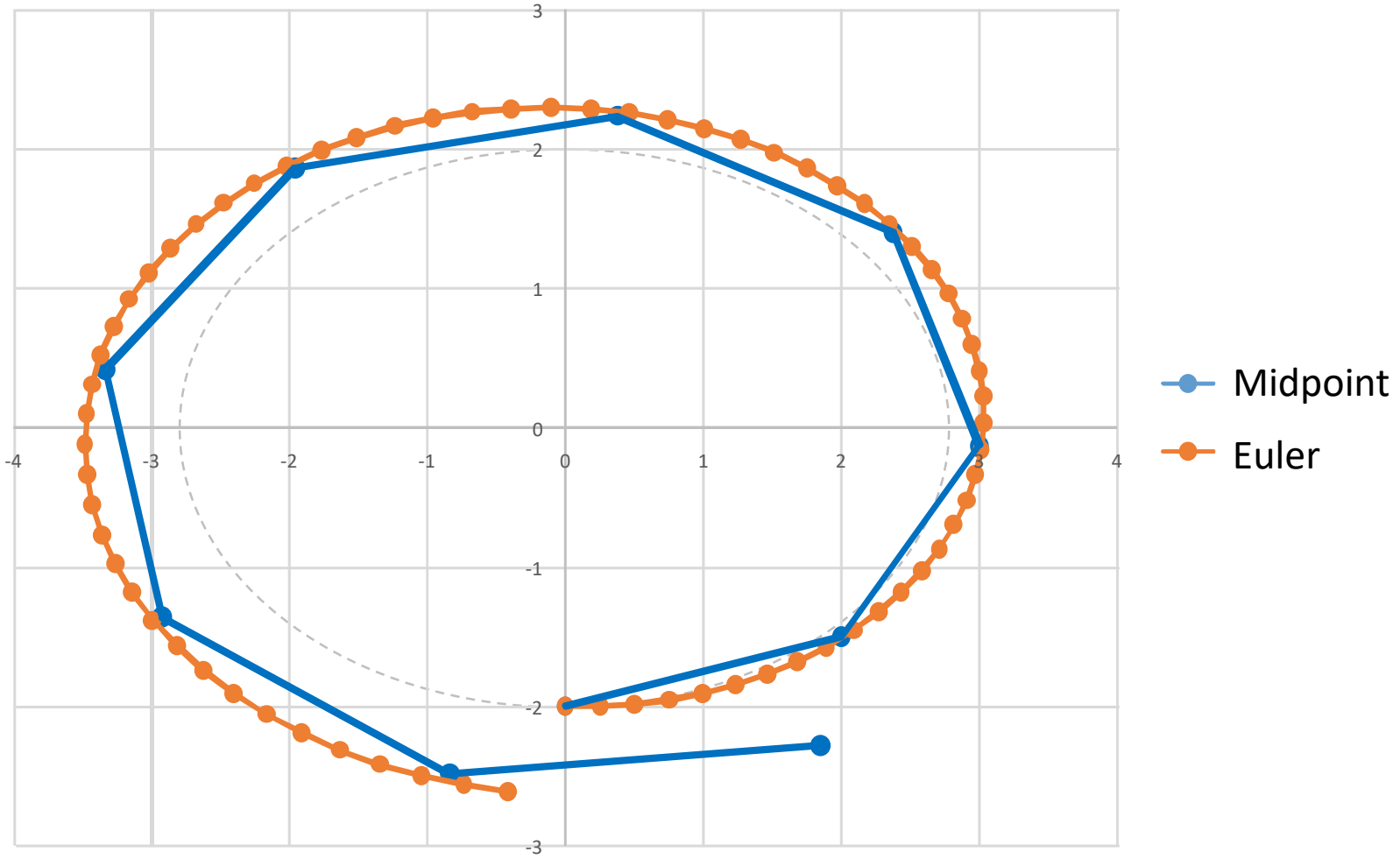


Numerical Integration of ODEs

Comparison: Midpoint method even with $\Delta t = 1$ (9 steps)

better

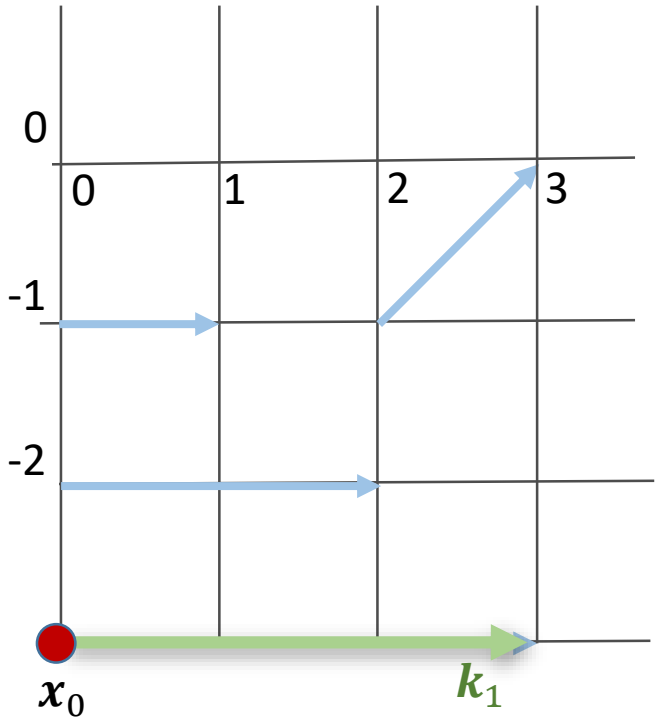
than Euler with $\Delta t = 1/8$ (71 steps)



Numerical Integration of ODEs

- Runge-Kutta of 4th order ($\Delta t = 1$)

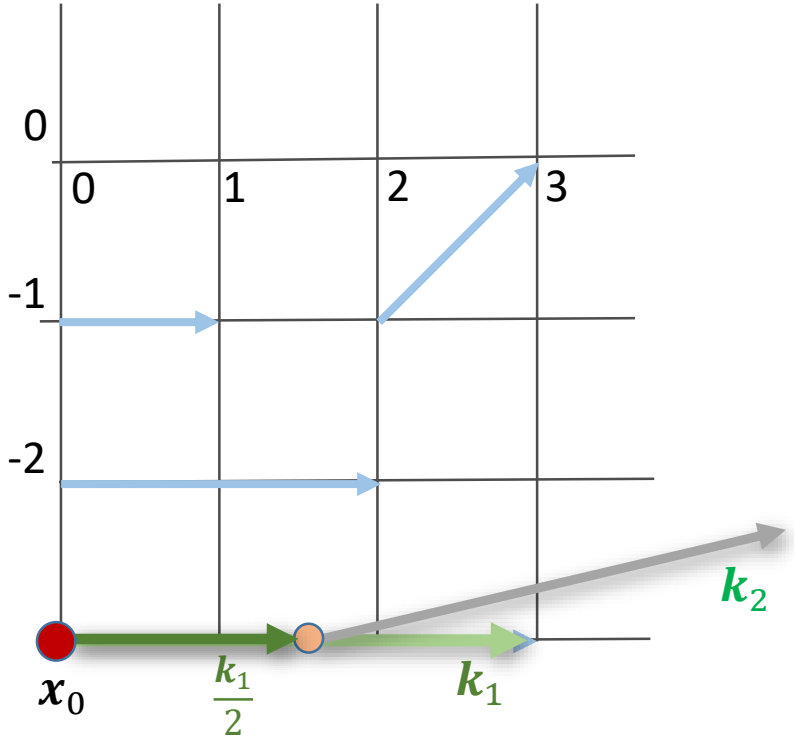
$$\boldsymbol{k}_1 = \Delta t \cdot \boldsymbol{v}(\boldsymbol{s}(t), t)$$



Numerical Integration of ODEs

- Runge-Kutta of 4th order ($\Delta t = 1$)

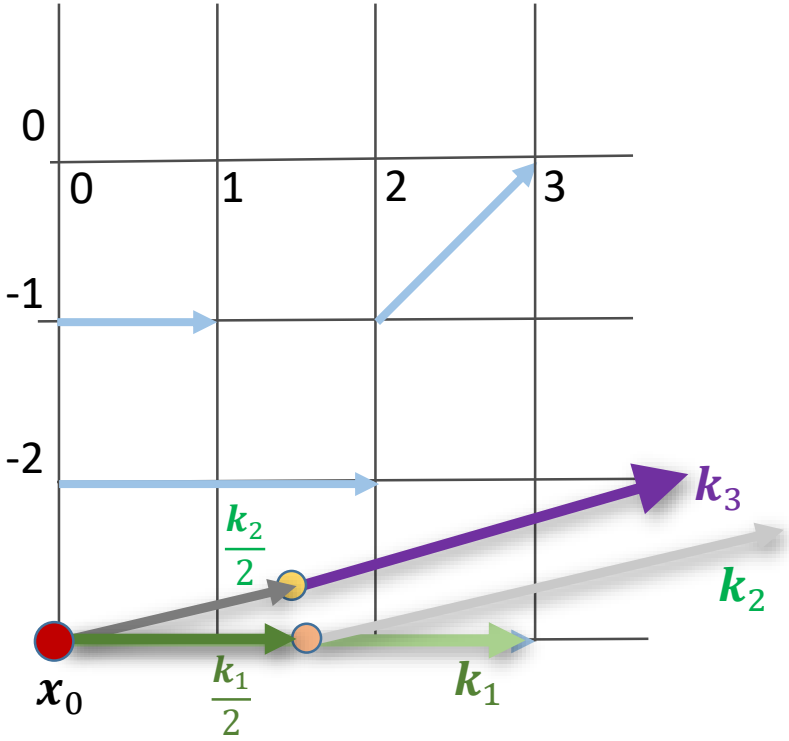
$$\boldsymbol{k}_1 = \Delta t \cdot \boldsymbol{v}(\boldsymbol{s}(t), t)$$
$$\boldsymbol{k}_2 = \Delta t \cdot \boldsymbol{v}\left(\boldsymbol{s}(t) + \frac{\boldsymbol{k}_1}{2}, t + \frac{\Delta t}{2}\right)$$



Numerical Integration of ODEs

- Runge-Kutta of 4th order ($\Delta t = 1$)

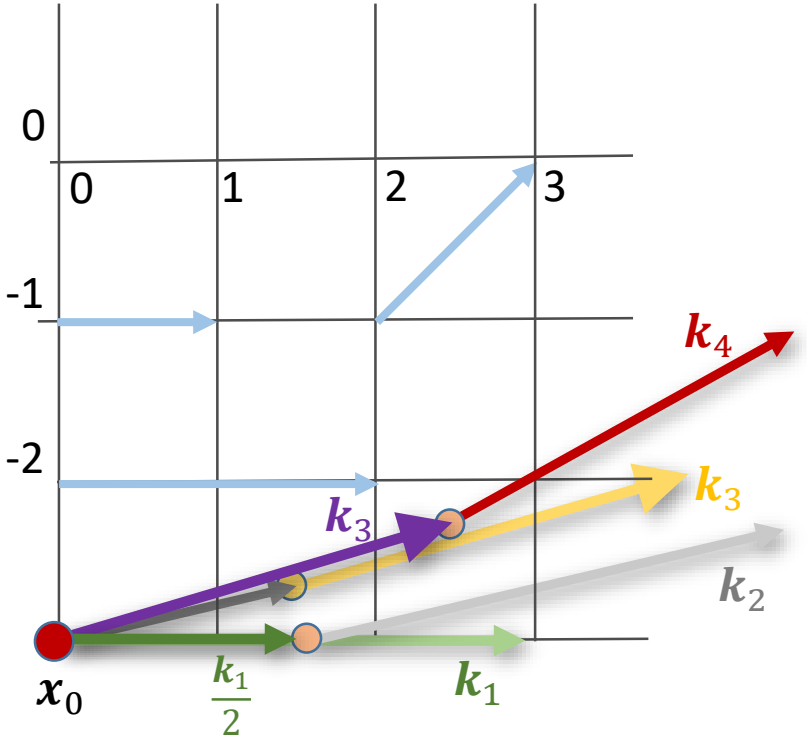
$$\boldsymbol{k}_1 = \Delta t \cdot \boldsymbol{v}(\boldsymbol{s}(t), t)$$
$$\boldsymbol{k}_2 = \Delta t \cdot \boldsymbol{v}\left(\boldsymbol{s}(t) + \frac{\boldsymbol{k}_1}{2}, t + \frac{\Delta t}{2}\right)$$
$$\boldsymbol{k}_3 = \Delta t \cdot \boldsymbol{v}\left(\boldsymbol{s}(t) + \frac{\boldsymbol{k}_2}{2}, t + \frac{\Delta t}{2}\right)$$



Numerical Integration of ODEs

- Runge-Kutta of 4th order ($\Delta t = 1$)

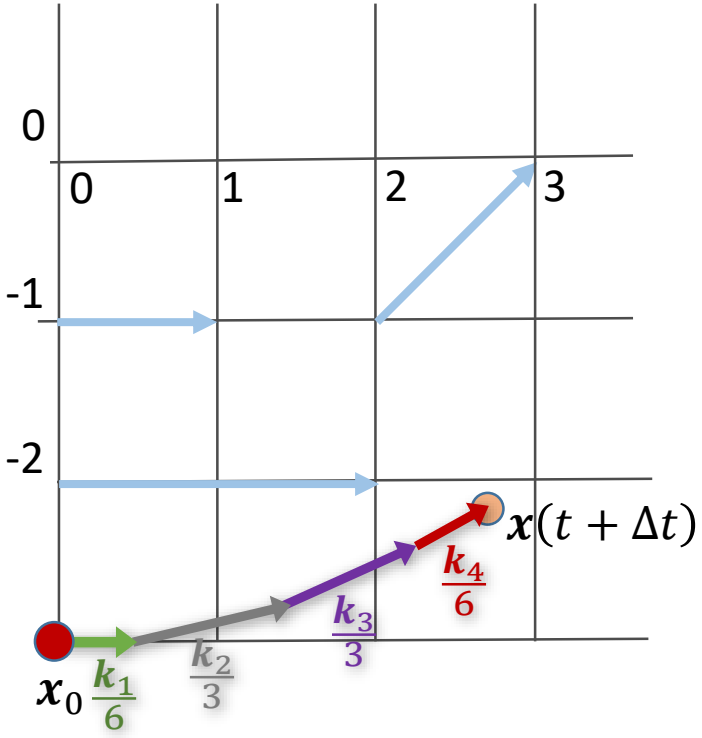
$$\boldsymbol{k}_1 = \Delta t \cdot \boldsymbol{v}(\boldsymbol{s}(t), t)$$
$$\boldsymbol{k}_2 = \Delta t \cdot \boldsymbol{v}\left(\boldsymbol{s}(t) + \frac{\boldsymbol{k}_1}{2}, t + \frac{\Delta t}{2}\right)$$
$$\boldsymbol{k}_3 = \Delta t \cdot \boldsymbol{v}\left(\boldsymbol{s}(t) + \frac{\boldsymbol{k}_2}{2}, t + \frac{\Delta t}{2}\right)$$
$$\boldsymbol{k}_4 = \Delta t \cdot \boldsymbol{v}(\boldsymbol{s}(t) + \boldsymbol{k}_3, t + \Delta t)$$



Numerical Integration of ODEs

- Runge-Kutta of 4th order ($\Delta t = 1$)

$$\boldsymbol{k}_1 = \Delta t \cdot \boldsymbol{v}(\boldsymbol{s}(t), t)$$
$$\boldsymbol{k}_2 = \Delta t \cdot \boldsymbol{v}\left(\boldsymbol{s}(t) + \frac{\boldsymbol{k}_1}{2}, t + \frac{\Delta t}{2}\right)$$
$$\boldsymbol{k}_3 = \Delta t \cdot \boldsymbol{v}\left(\boldsymbol{s}(t) + \frac{\boldsymbol{k}_2}{2}, t + \frac{\Delta t}{2}\right)$$
$$\boldsymbol{k}_4 = \Delta t \cdot \boldsymbol{v}(\boldsymbol{s}(t) + \boldsymbol{k}_3, t + \Delta t)$$



$$\boldsymbol{s}(t + \Delta t) = \boldsymbol{s}(t) + \frac{\boldsymbol{k}_1}{6} + \frac{\boldsymbol{k}_2}{3} + \frac{\boldsymbol{k}_3}{3} + \frac{\boldsymbol{k}_4}{6}$$

Numerical Integration of ODEs

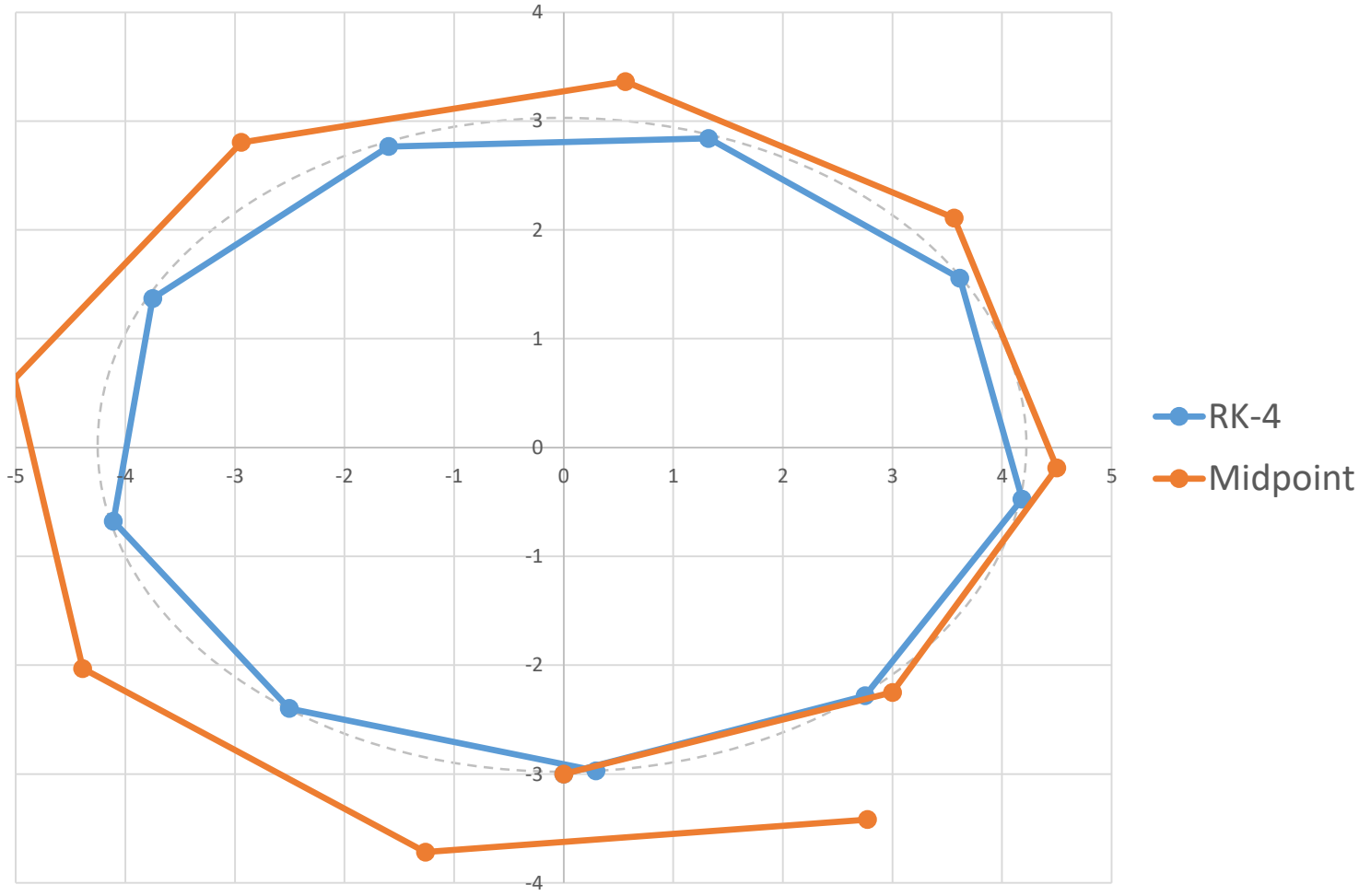
Comparison:
Runge-Kutta 4th order

vs.

midpoint

method

after 9 steps
$(\Delta t = 1)$



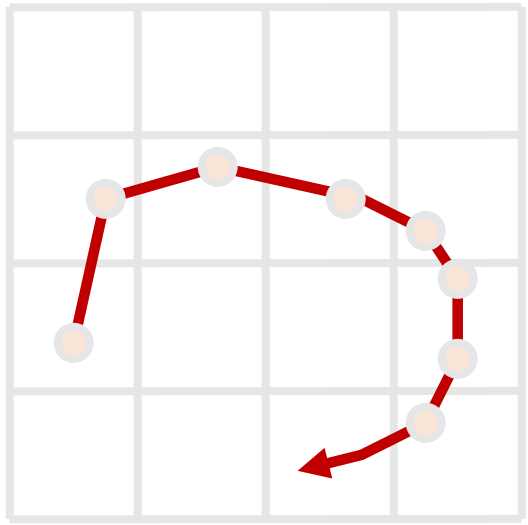
Numerical Integration of ODEs

- Summary
  - Analytic determination of streamlines usually not possible
  - Hence, numerical integration
  - Several methods available
    - Euler: simple, imprecise, especially with larger Δt
    - Runge-Kutta: more accurate in higher orders,
      pays off with complex flows

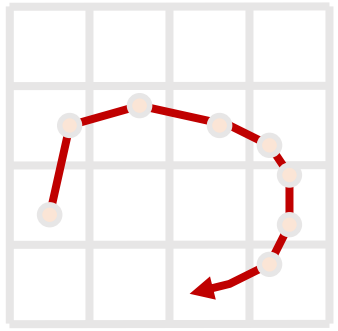
Particle Tracing on Grids

- Vector field given on a grid
  - Solve $\boldsymbol{s}(0) = \boldsymbol{x}_0, \quad \frac{d\boldsymbol{s}(t)}{dt} = \boldsymbol{v}(\boldsymbol{s}(t), t)$

    for the path line
  - Incremental integration
  - Discretized path of the particle



Particle Tracing on Grids



- Most simple case: Cartesian grid
- Basic algorithm:

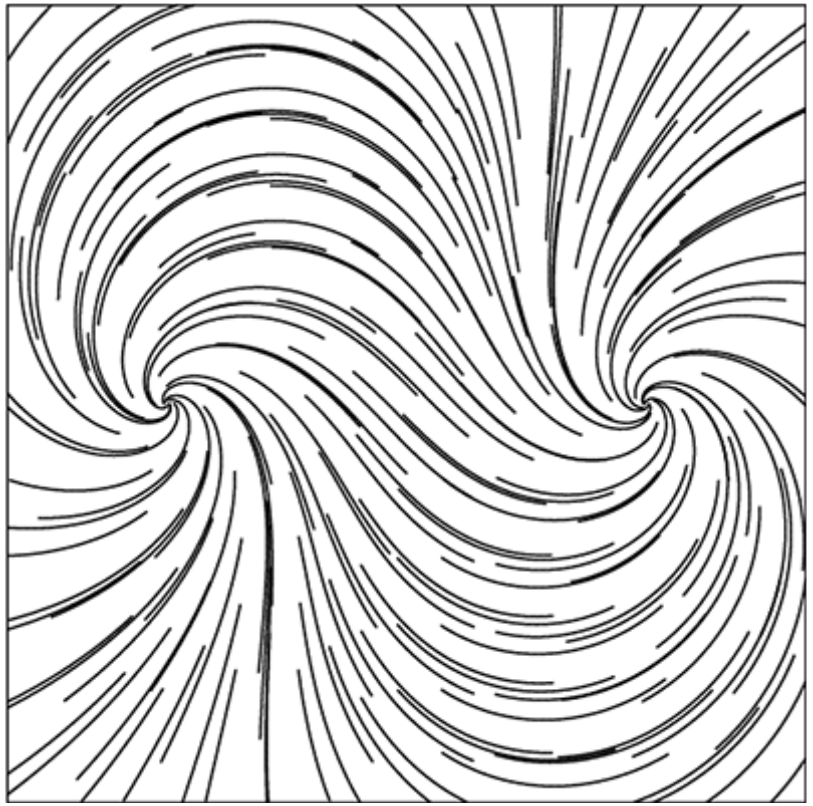
```
Select start point (seed point)
Find cell that contains start point  // point location
While (particle in domain) do
    Interpolate vector field at      // interpolation
      current position
    Integrate to new position        // integration
    Find new cell                    // point location
    Draw line segment between latest
      particle positions
Endwhile
```

Particle Tracing on Grids

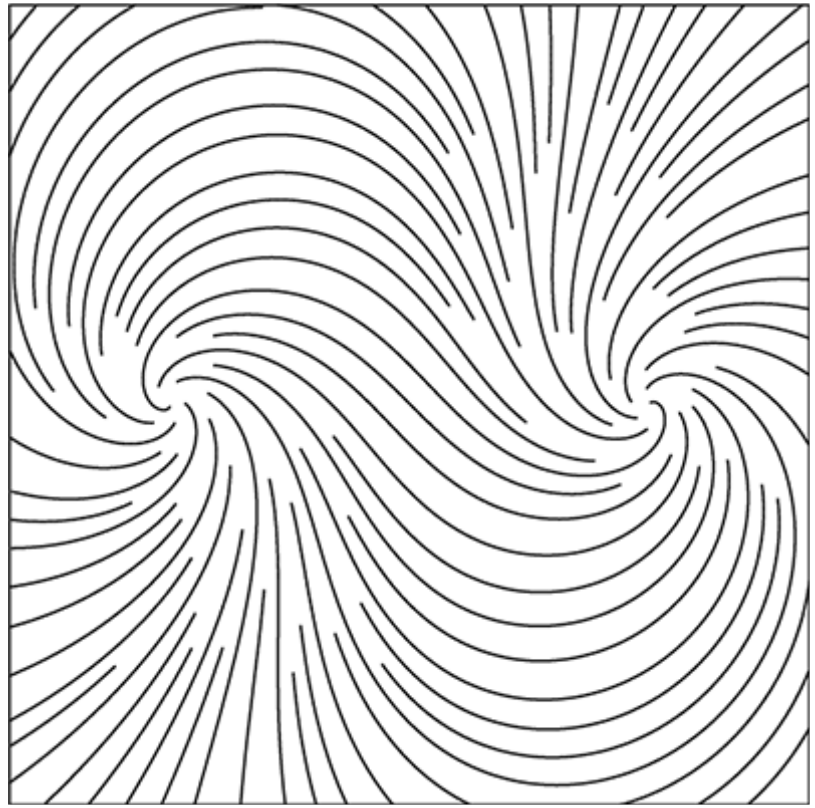
- Point location (cell search) on Cartesian grids
  - Indices of cell directly from position (x, y, z)
  - For example: $i_x = \lfloor (x - x_0) / \Delta x \rfloor$
  - Simple and fast
- Interpolation on Cartesian grids
  - Bilinear (in 2D) or trilinear (in 3D) interpolation
  - Required to compute the vector field (= velocity) inside a cell
  - Component-wise interpolation

Stream line placement in 2D

- Stream line placement
  - Irregular results when using regular grid



Stream line placement from regular grid

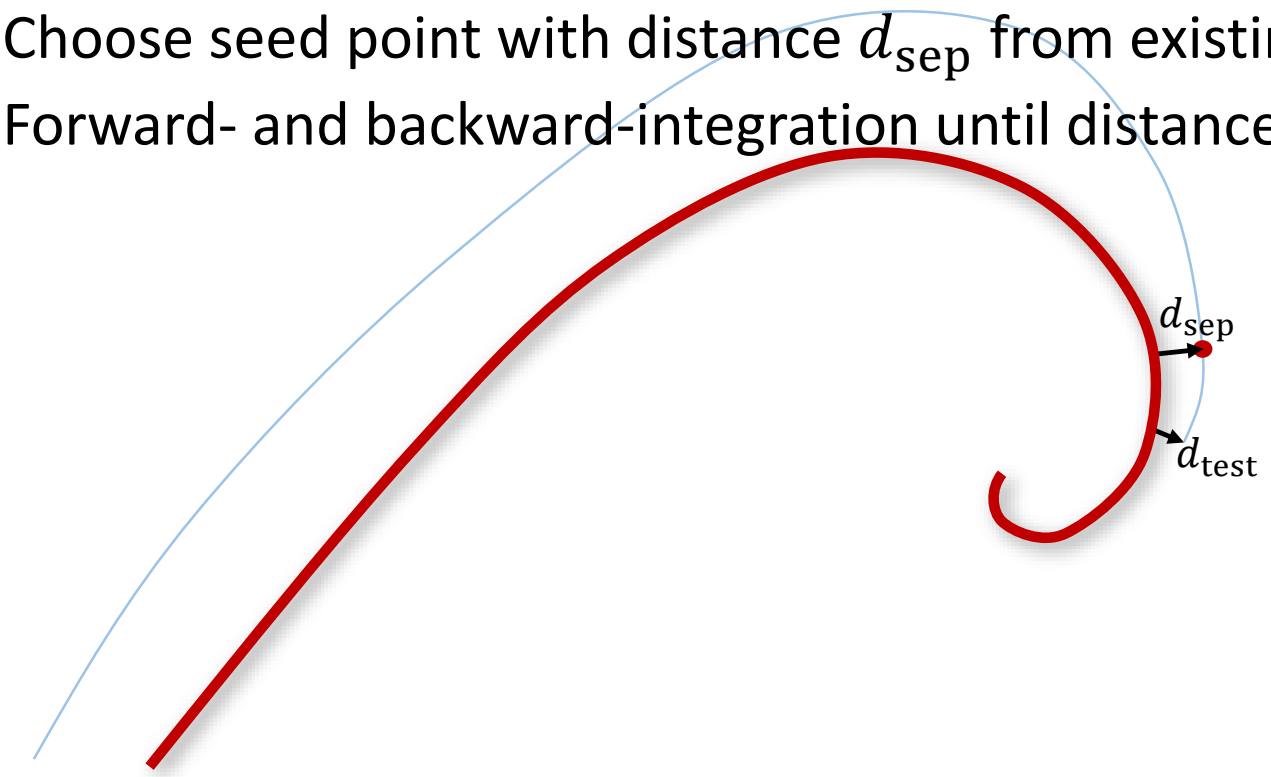


Evenly-spaced stream lines

[Jobard & Lefer 97]

Stream line placement in 2D

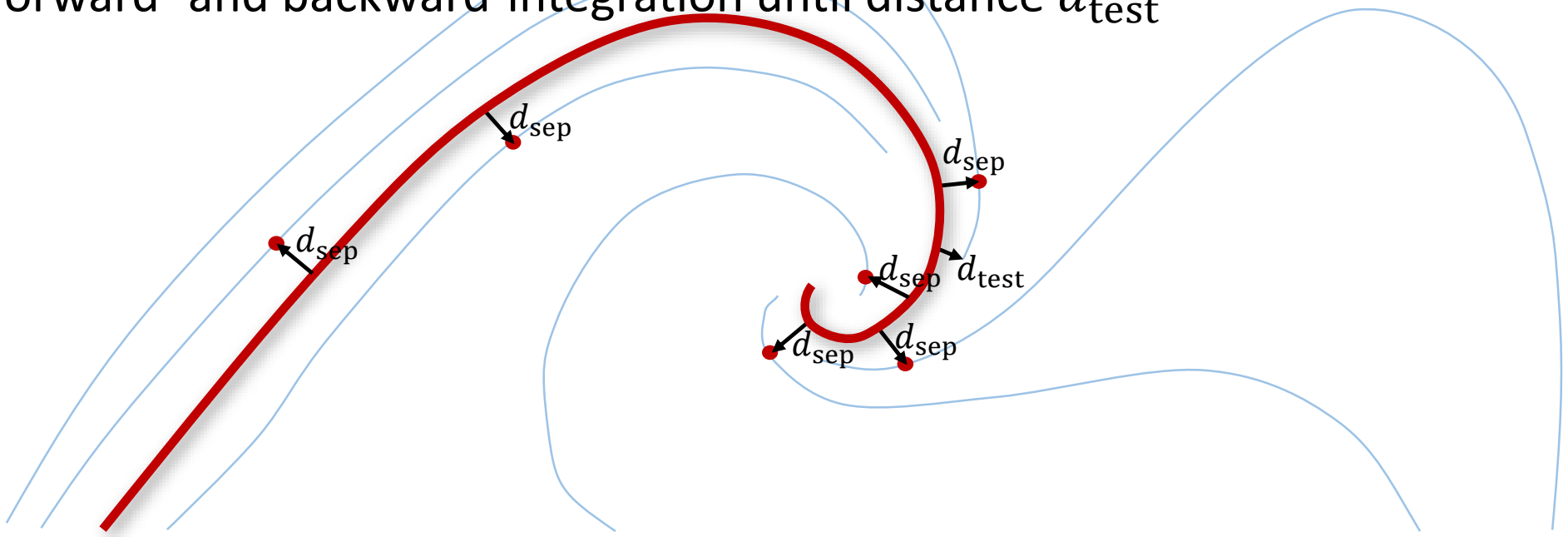
- Evenly-spaced streamlines
- Idea: stream lines should not get too close to each other
  - Choose seed point with distance d_{sep} from existing stream line
  - Forward- and backward-integration until distance d_{test}



[Jobard & Lefer 97]

Stream line placement in 2D

- Evenly-spaced streamlines
- Idea: stream lines should not get too close to each other
  - Choose seed point with distance d_{sep} from existing stream line
  - Forward- and backward-integration until distance d_{test}



[Jobard & Lefer 97]

Stream line placement in 2D

Algorithm:

Compute initial stream line and put into the queue

Initial stream line becomes current stream line

While not finished do:

- Try: get new seed point with distance d_{sep} from current stream line
- If successful then compute new stream line and put into queue
- Else If no more stream lines in queue then exit loop
- Else next stream line in queue becomes current streamline

EndWhile

[Jobard & Lefer 97]

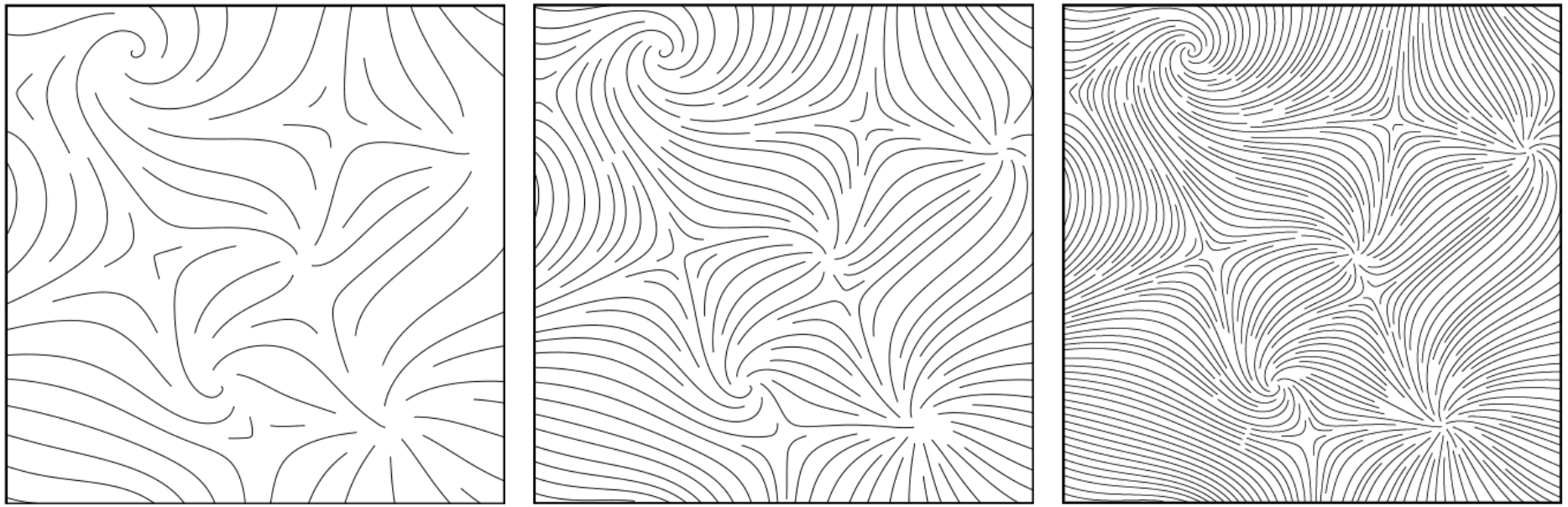
Stream line placement in 2D

- When to stop stream line integration
  - When distance to neighboring stream line $\leq d_{\text{test}}$
  - When stream line leaves flow domain
  - When stream line runs into fixed point $(\boldsymbol{v}(\boldsymbol{x}^*) = 0)$
  - When stream line gets too close to itself
  - After a certain number of maximal steps

[Jobard & Lefer 97]

Stream line placement in 2D

- Variations of d_{sep} in relation to image width

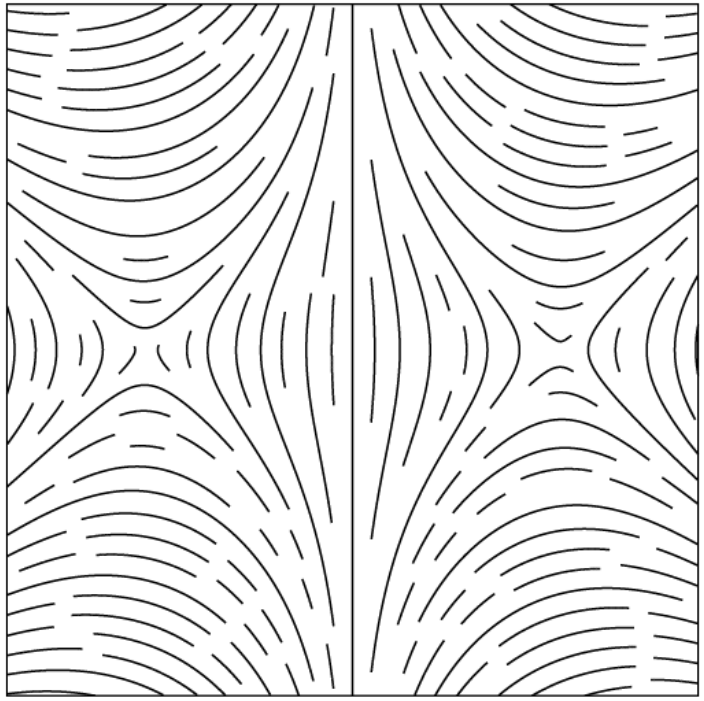


6% 3% 1.5%

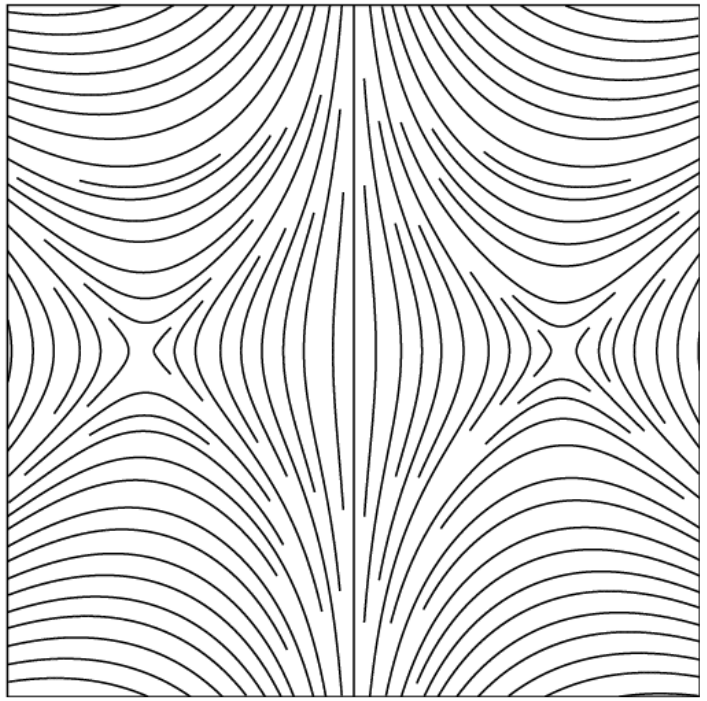
[Jobard & Lefer 97]

Stream line placement in 2D

- Variations of d_{test}



$d_{\text{test}} = 0.9\ d_{\text{sep}}$

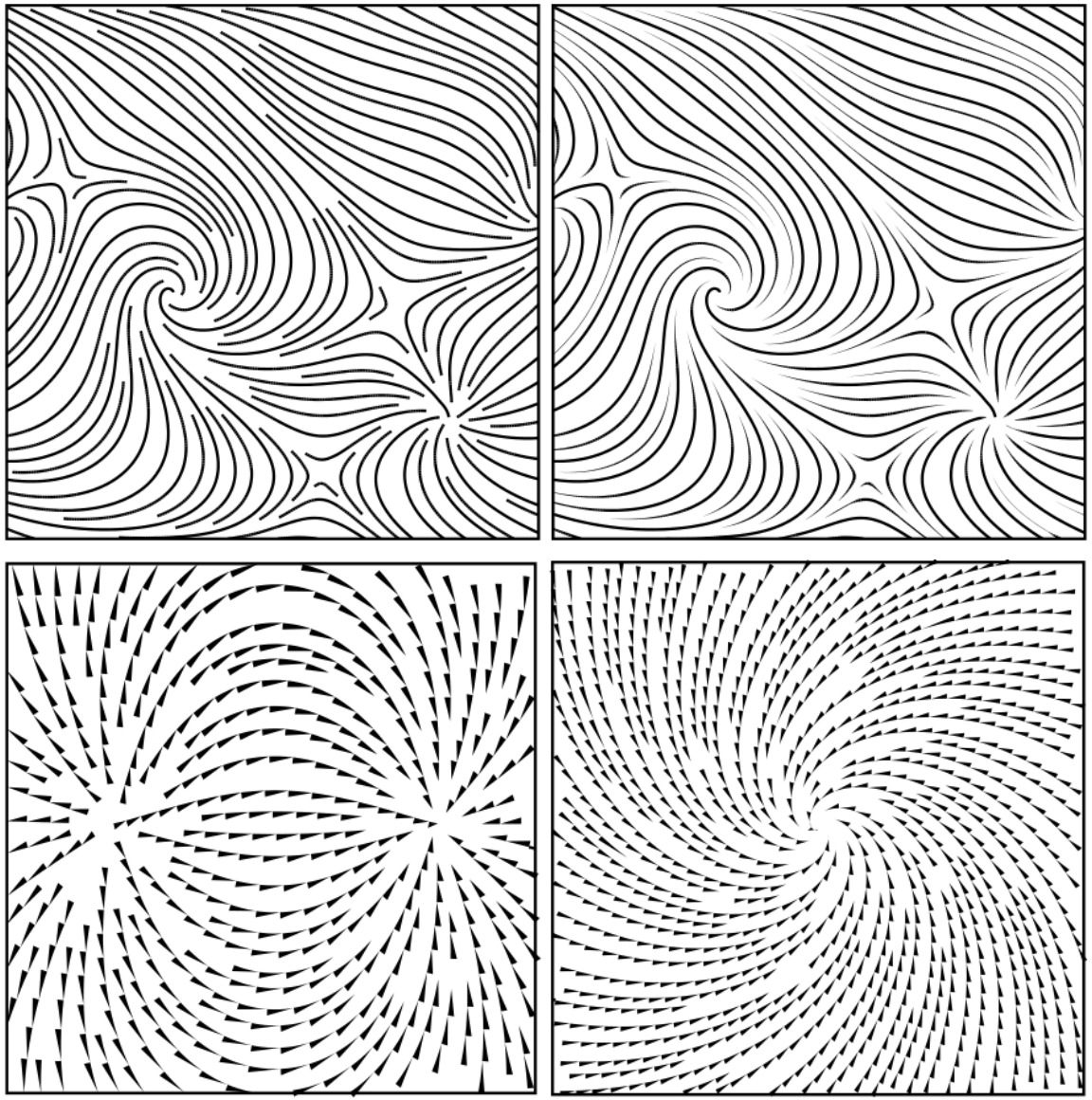


$d_{\text{test}} = 0.5\ d_{\text{sep}}$

[Jobard & Lefer 97]

Stream line placement in 2D

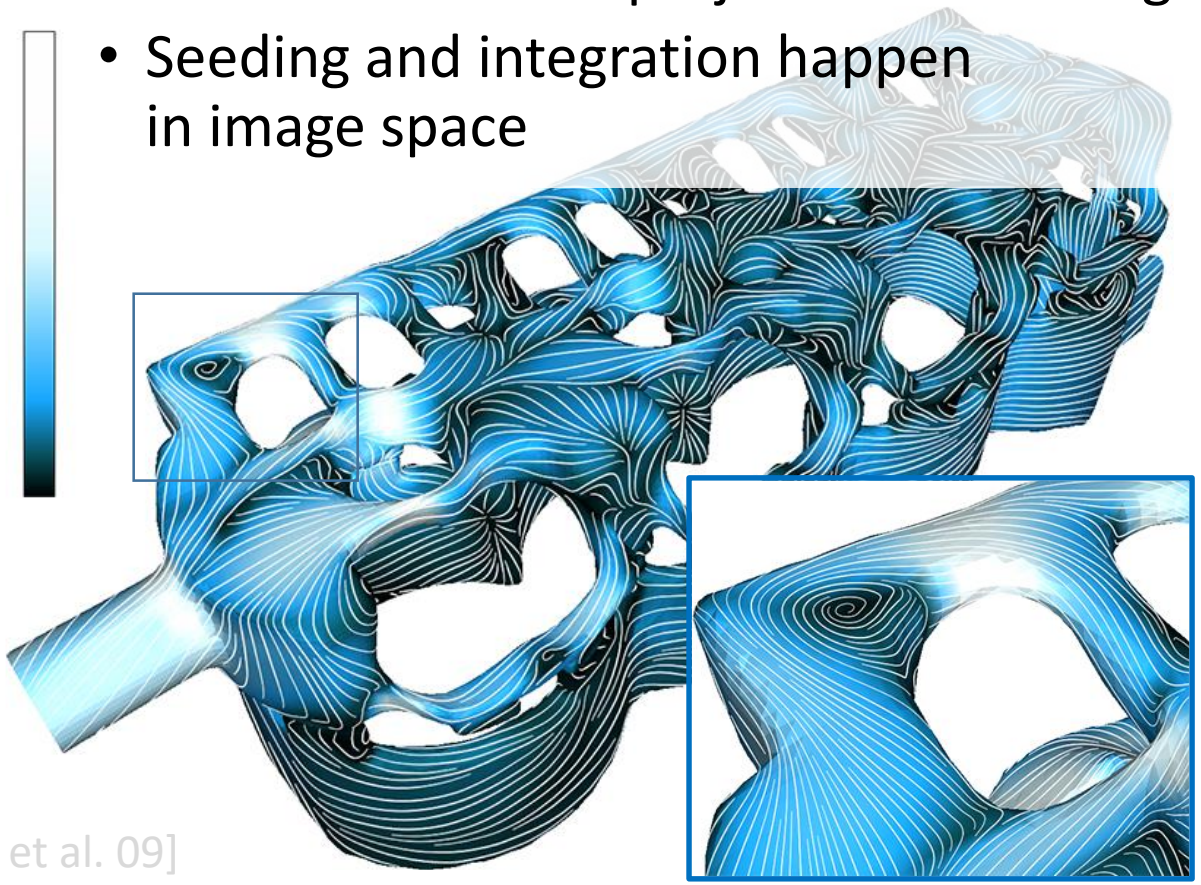
- Change thickness in relation to distance d
  - If $d \geq d_{\text{sep}}$: 1.0
  - If $d < d_{\text{sep}}$: $\frac{d - d_{\text{test}}}{d_{\text{sep}} - d_{\text{test}}}$
- Directional glyphs



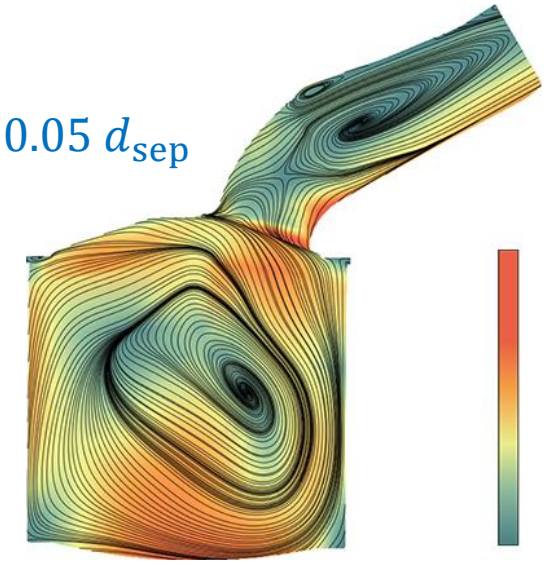
[Jobard & Lefer 97]

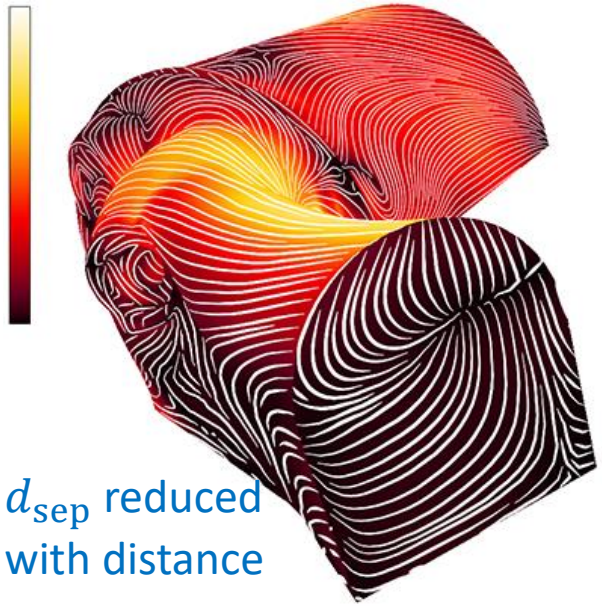
Stream line placement on surfaces

- Image-space technique
  - Vector field is first projected to 2D image
  - Seeding and integration happen in image space



$d_{\text{test}} = 0.05\ d_{\text{sep}}$





d_{sep} reduced with distance

[Spencer et al. 09]

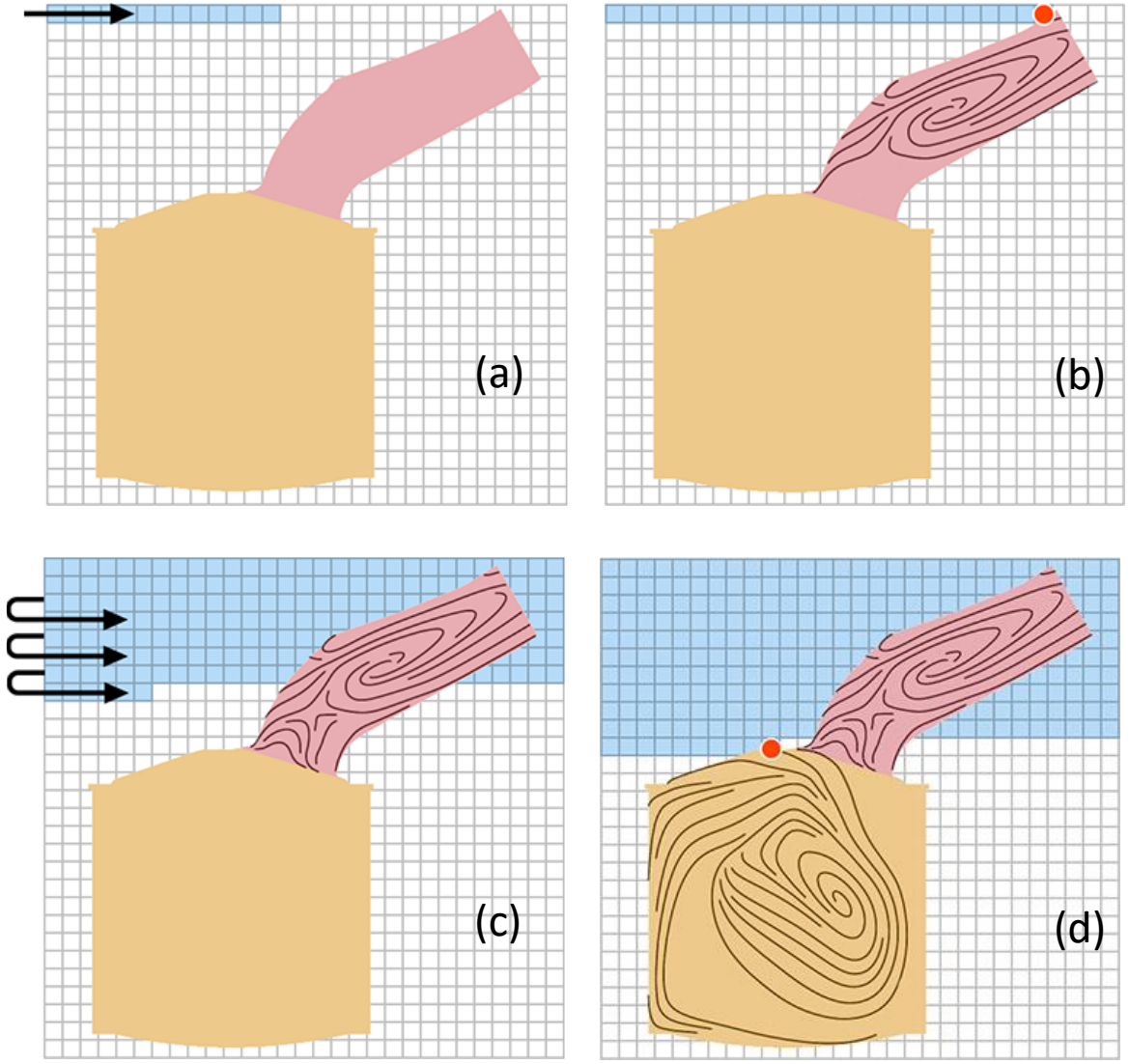
Stream line placement on surfaces

- Vector field is first projected to 2D image

a) 2D image is scanned at intervals d_{sep}

b) Seedpoint is found & stream lines are traced in that region

c) Scanning continues until seedpoint in new region is found (d)

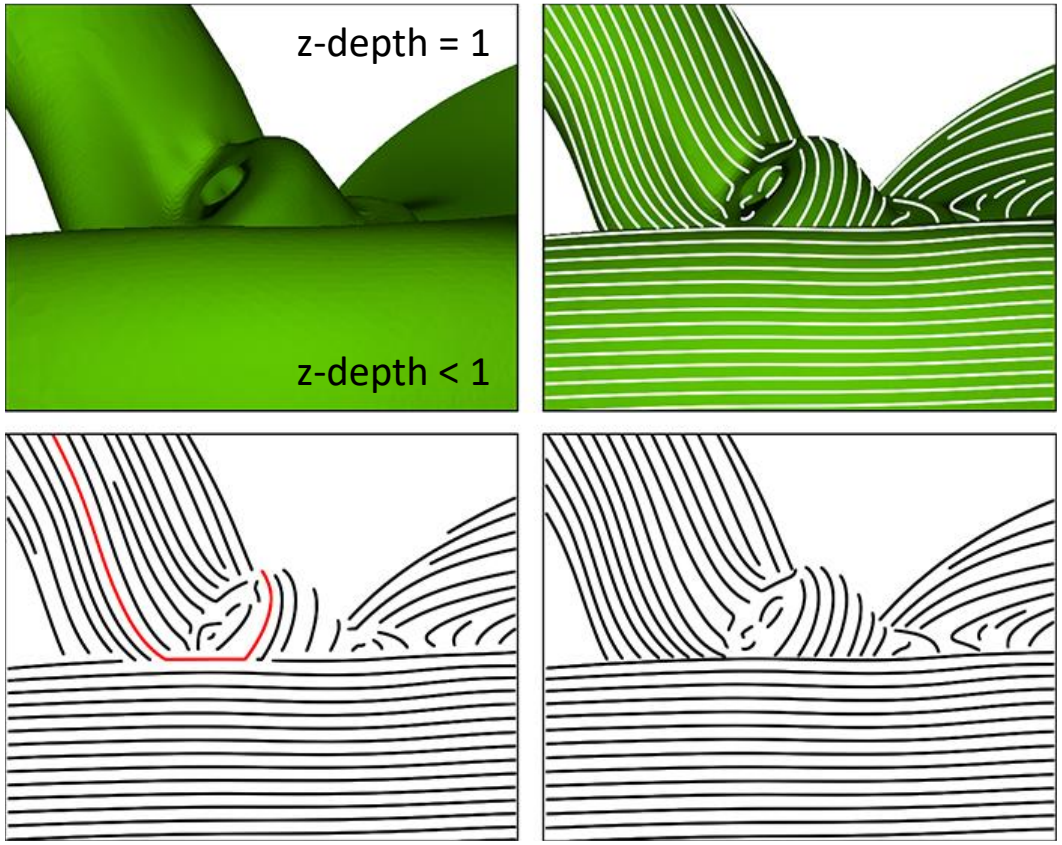


[Spencer et al. 09]

Stream line placement on surfaces

- Discontinuity detection
  - Stop stream line integration when z-depth goes to one (edge of model)

  … or when z-depth changes too abruptly (edge of overlapping regions)



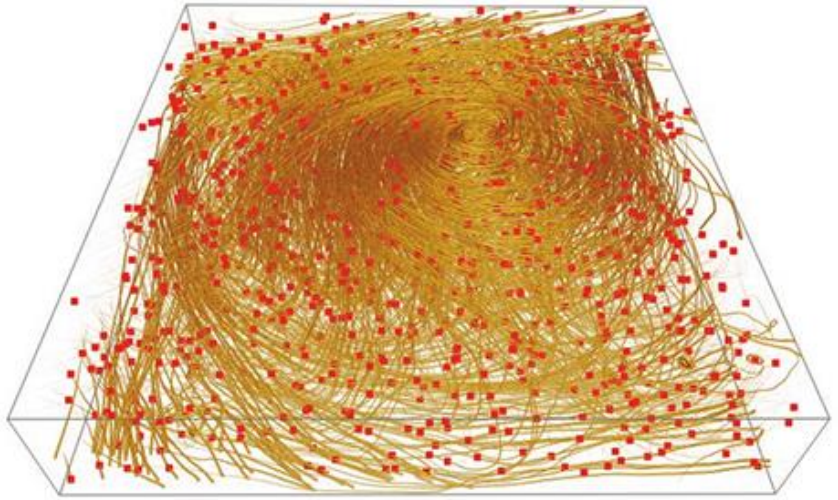
Without detection: stream lines run off edge of surface (red)

With detection: underlying surface is better reflected

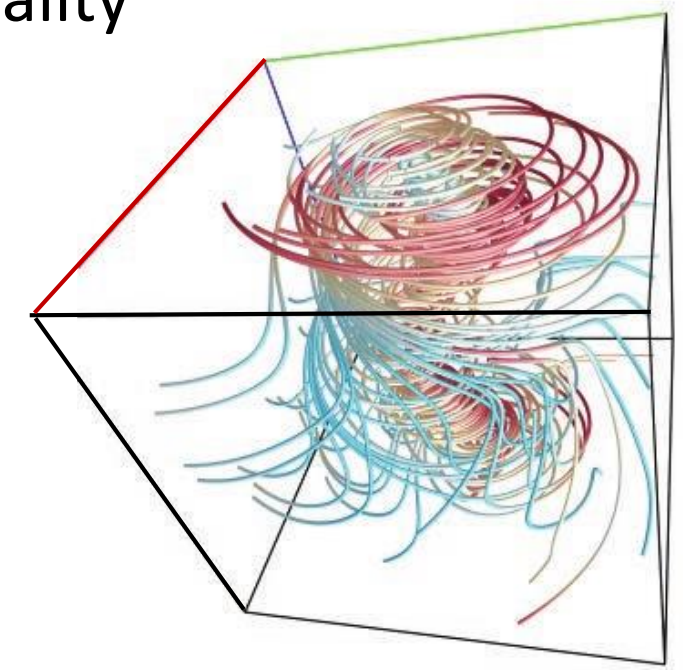
[Spencer et al. 09]

Stream line placement in 3D

- Placement of seeds directly affects the visual quality
  - Too many: scene cluttering (occlusion)
  - Too little: no patterns forming
- Seeding has to be at the right place and in the right amount!



A bad seeding example



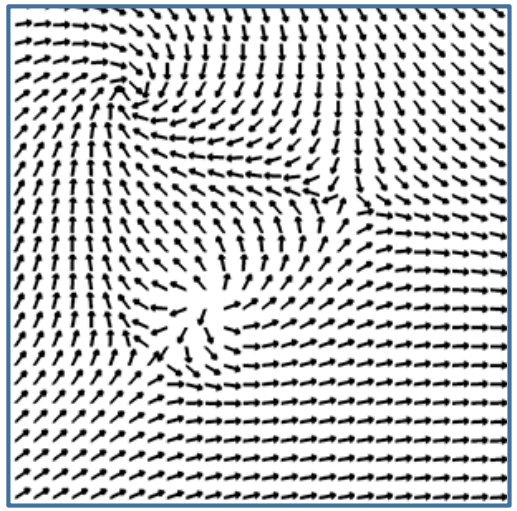
[Tao et al. 13]

Finding best stream lines and best view point from many candidates

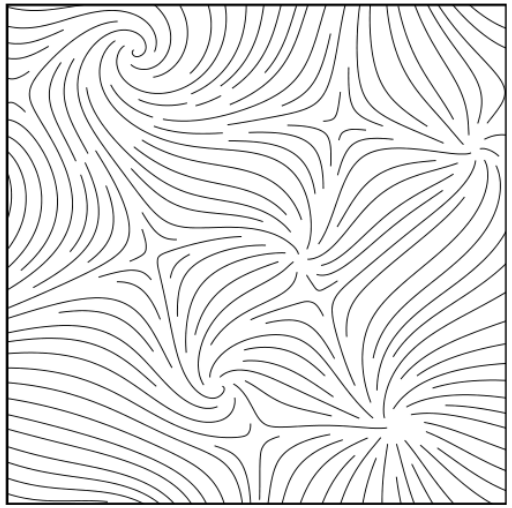
Stream line placement

- Summary
  - Irregular results when using regular grid or random seeding
  - Stream lines should not get too close to each other
  - Seeding/placement techniques typically limed to 2D or surfaces and stationary vector fields
  - Not many methods for 3D and/or time-varying flows yet

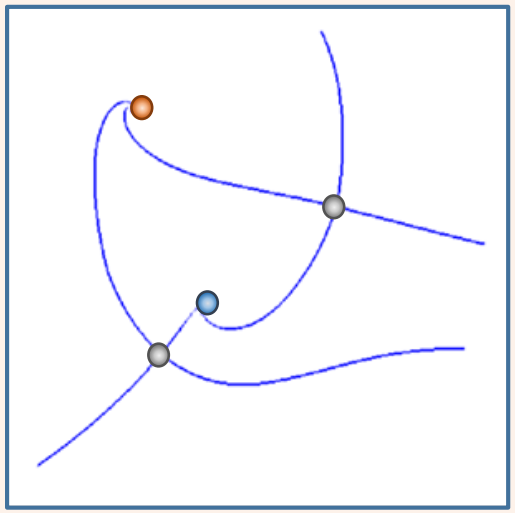
Flow visualization – Approaches



Direct flow visualization
(arrows, color coding, …)

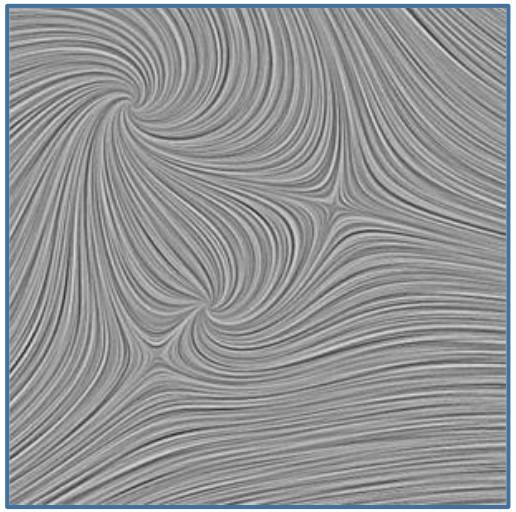


Geometric flow visualization
(stream lines/surfaces, …)



Sparse (feature-based) vis.

- Global computation of flow features
- Vortices, shockwaves, vector field topology



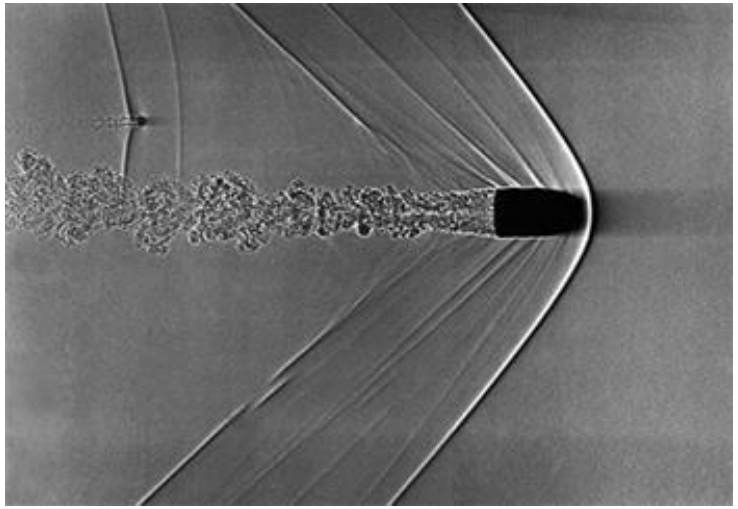
Dense (texture-based) vis.

Sparse Flow Visualization

Flow features

- Vortices
  - One of most prominent features
  - Important in many applications (turbulent flows)

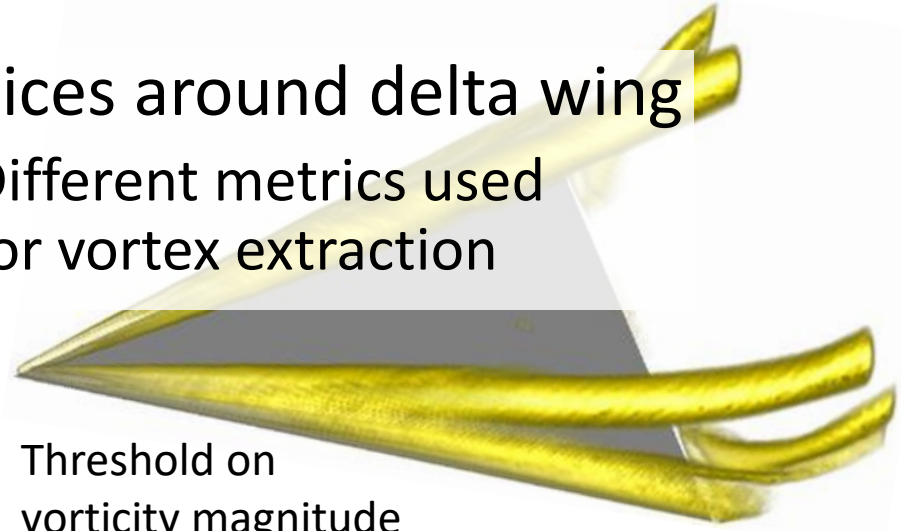
- Shock waves
  - Characterized by sharp discontinuities in flow attributes (pressure, velocity magnitude, …)



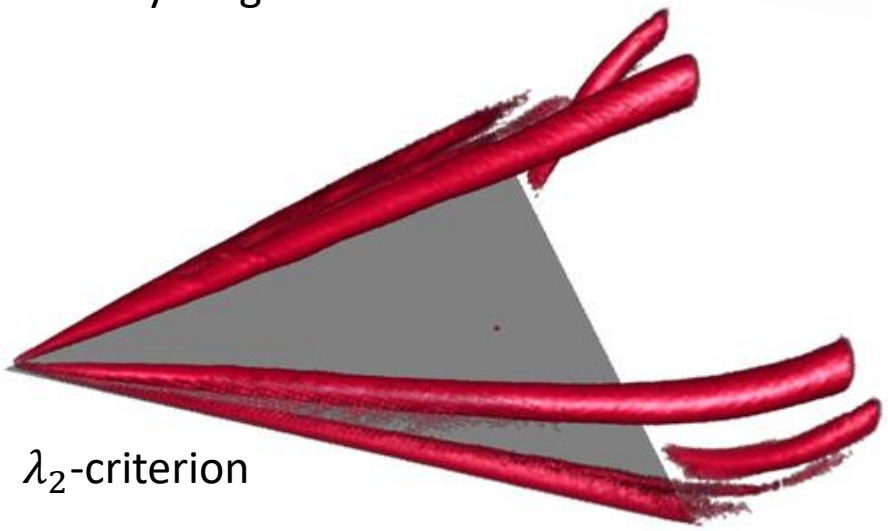
Bullet traveling through air at about 1.5 times sound speed

Flow features

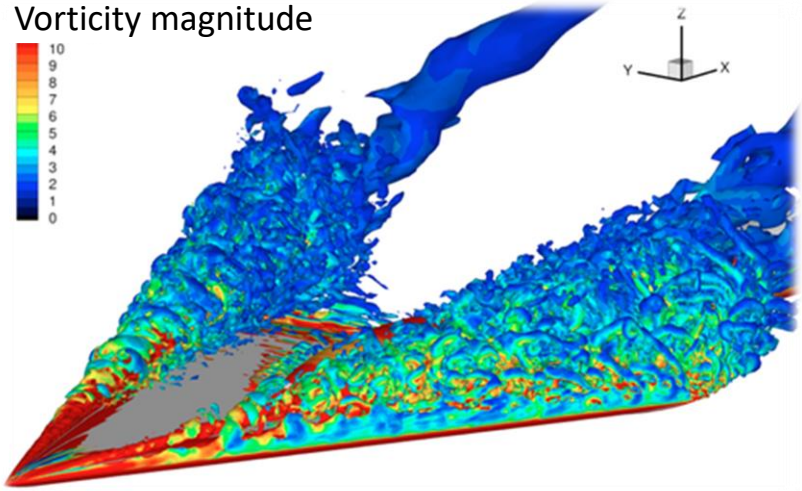
- Vortices around delta wing
  - Different metrics used for vortex extraction

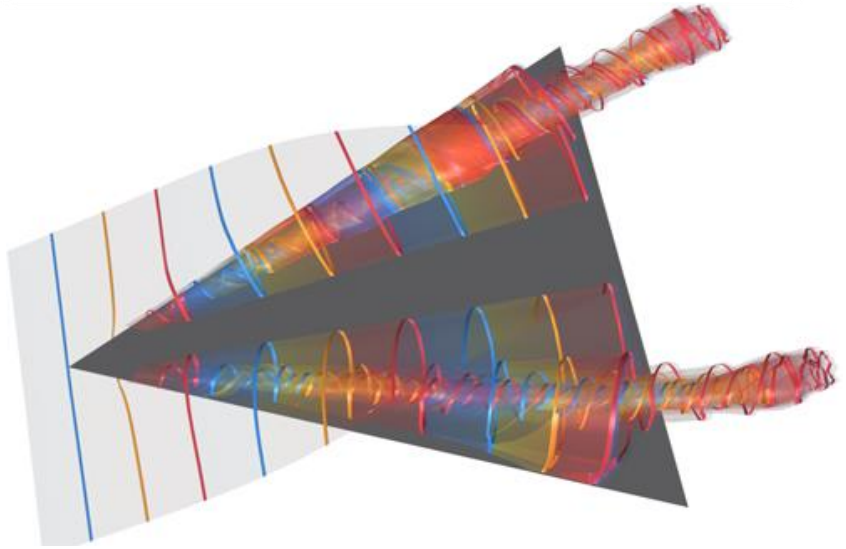


Threshold on vorticity magnitude



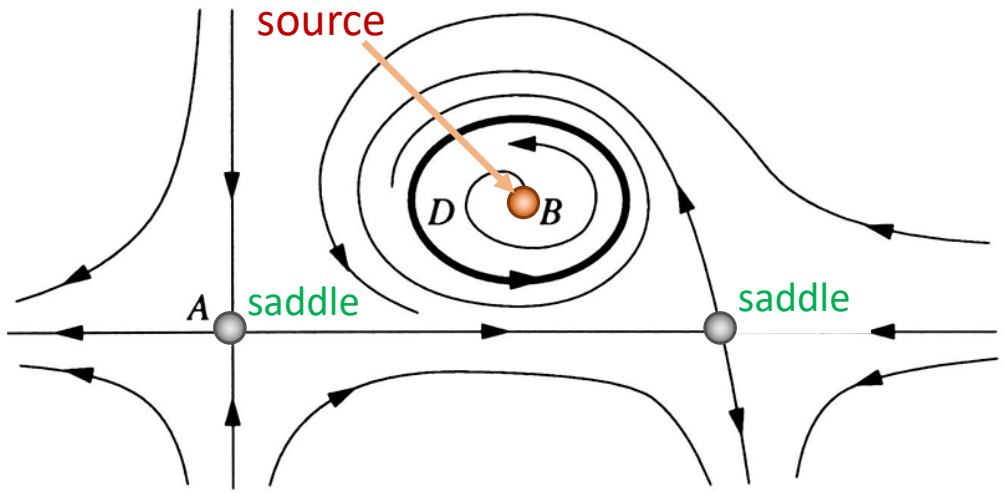
λ_2-criterion

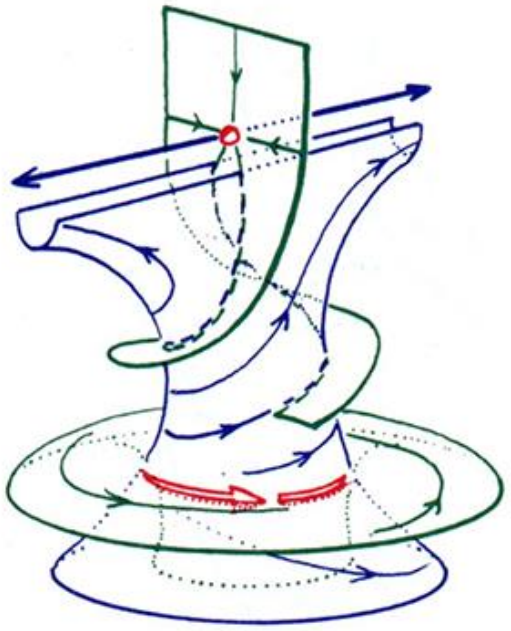




Vector field topology

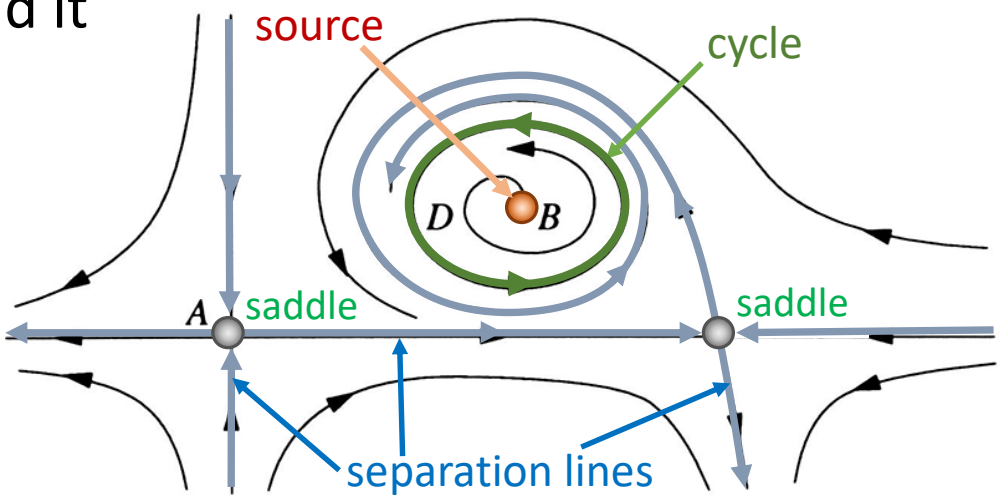
- Idea: Do not draw “all” stream lines, but only the “important” ones
- Show only topological skeleton
  - Connection of critical points
  - Characterization of global flow structures

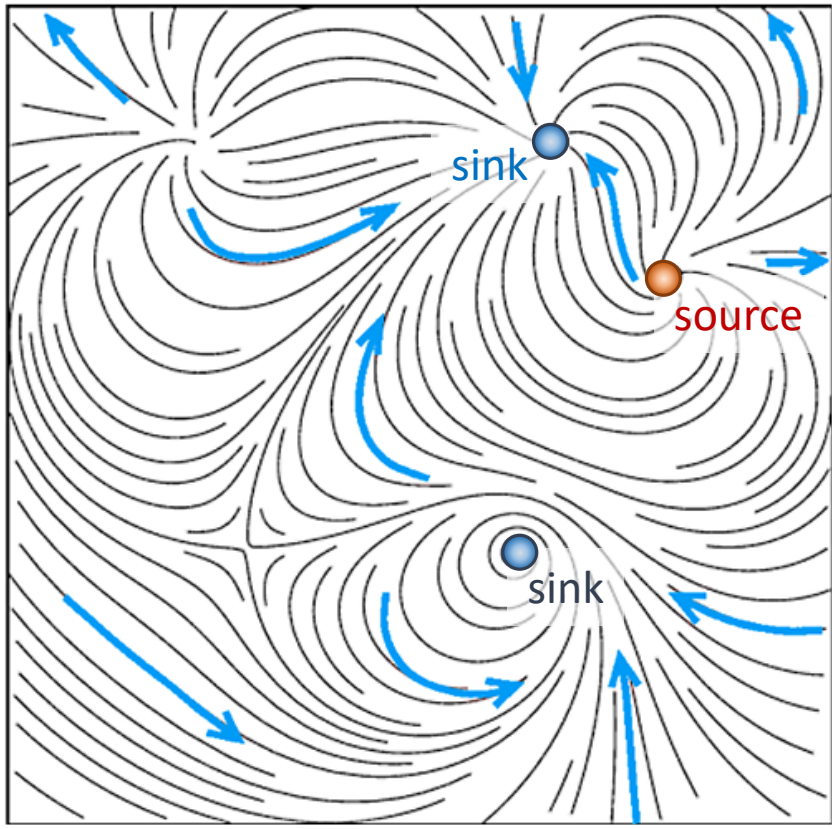




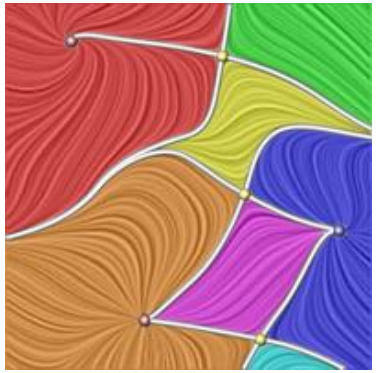
Vector field topology

- Critical points: singularities in vector field such that $\boldsymbol{v}(\boldsymbol{x}^*) = 0$
  - Points where magnitude of vector goes to zero and direction of vector is undefined
  - Stream lines reduced to single point
  - Type of critical point determines flow pattern around it

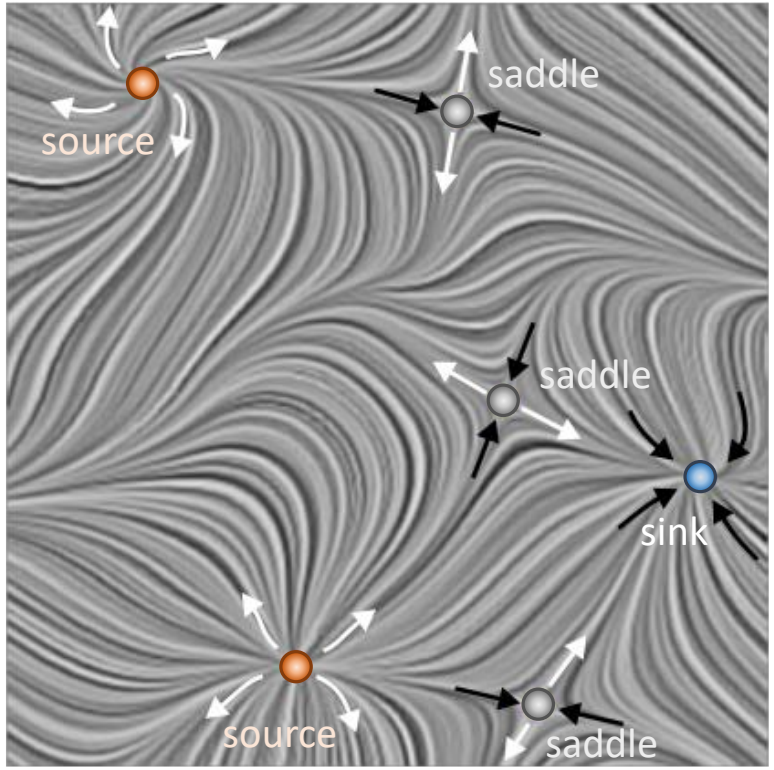




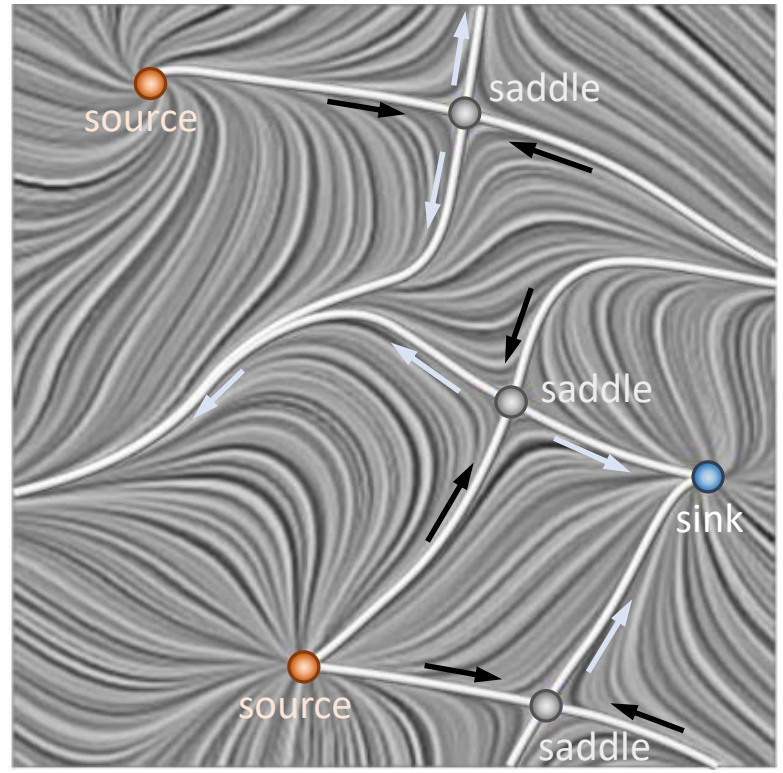
Vector field topology



- Topological skeleton / graph
  - Nodes: critical points
  - Edges: separation lines and cycles
  - Flow divided into regions with similar properties



Critical points



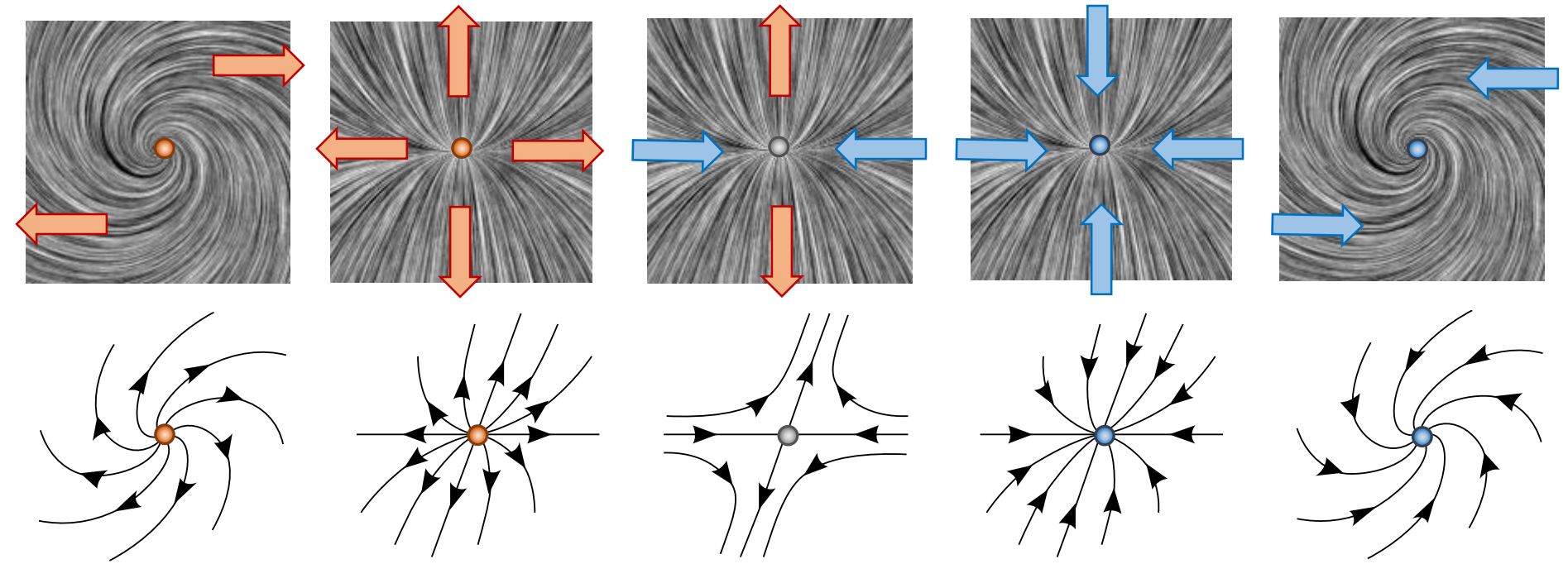
Critical points + separation lines

[Weinkauf]

Vector field topology (2D)

- Critical points in 2D
  - Classified by eigenvalues λ_1, λ_2 of Jacobian matrix (Governs behavior near $\boldsymbol{v}(\boldsymbol{x}^*) = 0$)
  - Sign of real parts of λ_1, λ_2; complex λ_1, λ_2 for rotating flow

Vector field topology (2D)



Circulating source
$\mathrm{Im}(\lambda_{1,2}) \neq 0$
$\mathrm{Re}(\lambda_{1,2}) > 0$

Noncirculating source
$\mathrm{Im}(\lambda_{1,2}) = 0$
$\mathrm{Re}(\lambda_{1,2}) > 0$

Saddle point
$\mathrm{Im}(\lambda_{1,2}) = 0$
$\lambda_1 \lambda_2 < 0$

Noncirculating sink
$\mathrm{Im}(\lambda_{1,2}) = 0$
$\mathrm{Re}(\lambda_{1,2}) < 0$

Circulating sink
$\mathrm{Im}(\lambda_{1,2}) \neq 0$
$\mathrm{Re}(\lambda_{1,2}) < 0$

Vector field topology (2D)

- Eigenvalues and eigenvectors computation

$$\mathbf{J}\boldsymbol{u} = \lambda\boldsymbol{u} \qquad (\mathbf{J} - \lambda\mathbf{I})\boldsymbol{u} = 0$$

$$\det(\mathbf{J} - \lambda\mathbf{I}) = 0$$

$\mathbf{J}$ … Jacobi matrix
$\boldsymbol{u}$ … eigenvector (non-zero)
λ … eigenvalue
$\mathbf{I}$ … identity matrix

Needs to be fulfilled such that $\boldsymbol{u}$ is non-zero

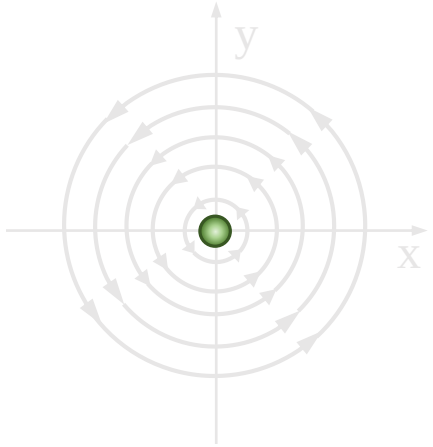
- Example: Curl field $\boldsymbol{v}(x,y) = \begin{pmatrix} -y \\ x \end{pmatrix}$, $\mathbf{J} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$

  Compute eigenvalues λ_1, λ_2:

$$\det\left[\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} - \lambda\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\right] = \det\begin{pmatrix} -\lambda & -1 \\ 1 & -\lambda \end{pmatrix} = \lambda^2 + 1$$

  → $\lambda^2 + 1 = 0$

  We get complex eigenvalues: $\lambda_1 = i,\ \lambda_2 = -i$



Center

$\mathrm{Im}(\lambda_1) = -\mathrm{Im}(\lambda_2) \neq 0$

$\mathrm{Re}(\lambda_{1,2}) = 0$

Vector field topology (2D)

- Example: Vector field $\boldsymbol{v}(x,y) = \begin{pmatrix} x+y \\ 4x-2y \end{pmatrix}$, $\mathbf{J} = \begin{pmatrix} 1 & 1 \\ 4 & -2 \end{pmatrix}$

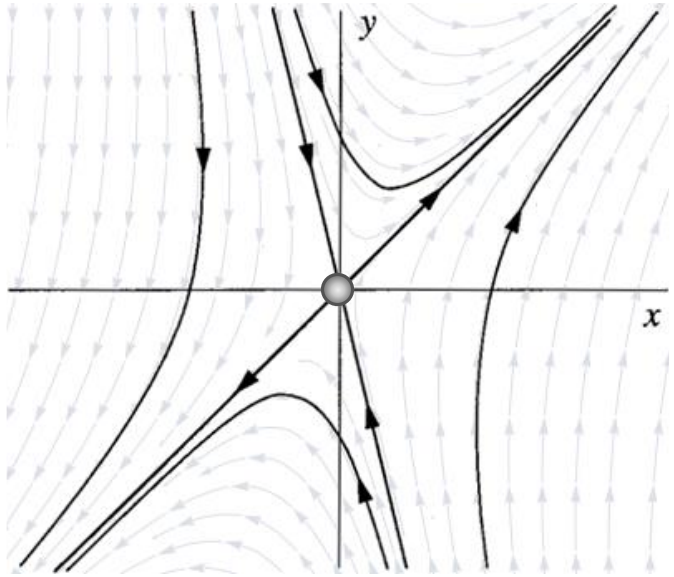
  Compute eigenvalues λ_1, λ_2:

  $$\det\left[\begin{pmatrix} 1 & 1 \\ 4 & -2 \end{pmatrix} - \lambda \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\right] = \det\begin{pmatrix} 1-\lambda & 1 \\ 4 & -2-\lambda \end{pmatrix} = \lambda^2 + \lambda - 6$$

  $\rightarrow \lambda^2 + \lambda - 6 = 0$

  $\rightarrow \lambda_{1,2} = \frac{-1 \pm \sqrt{1+24}}{2}$

  We get real eigenvalues: $\lambda_1 = -3,\ \lambda_2 = 2$



Saddle point

$\mathrm{Im}(\lambda_{1,2}) = 0$

$\lambda_1 \lambda_2 < 0$

Computing a determinant

$$\det\begin{pmatrix} a & b \\ c & d \end{pmatrix} = ad - bc$$

Solving a quadratic equation

$$ax^2 + bx + c = 0$$

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Vector field topology (2D)

- **Example:** Vector field $\boldsymbol{v}(x,y) = \begin{pmatrix} x+y \\ 4x-2y \end{pmatrix}$, $\mathbf{J} = \begin{pmatrix} 1 & 1 \\ 4 & -2 \end{pmatrix}$

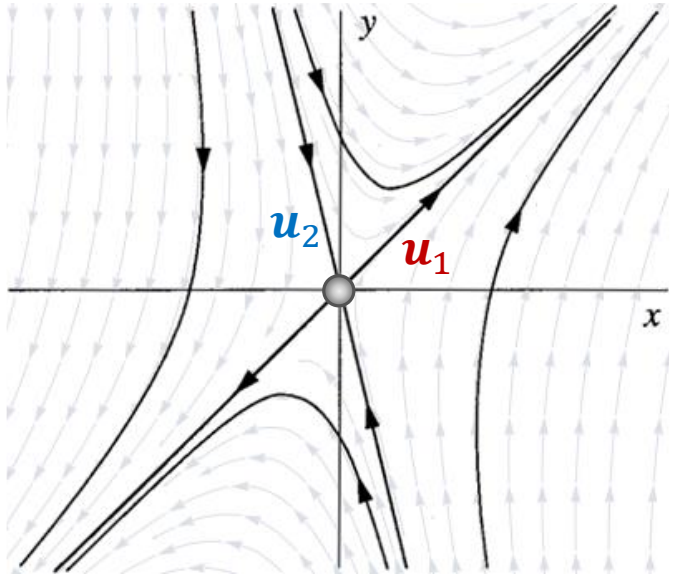
  Eigenvalues: $\lambda_1 = 2,\ \lambda_2 = -3$

  Compute eigenvectors $\boldsymbol{u}_1, \boldsymbol{u}_2$: $(\mathbf{J} - \lambda\mathbf{I})\boldsymbol{u} = 0$

  $$\begin{pmatrix} 1-\lambda & 1 \\ 4 & -2-\lambda \end{pmatrix} \begin{pmatrix} u_x \\ u_y \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

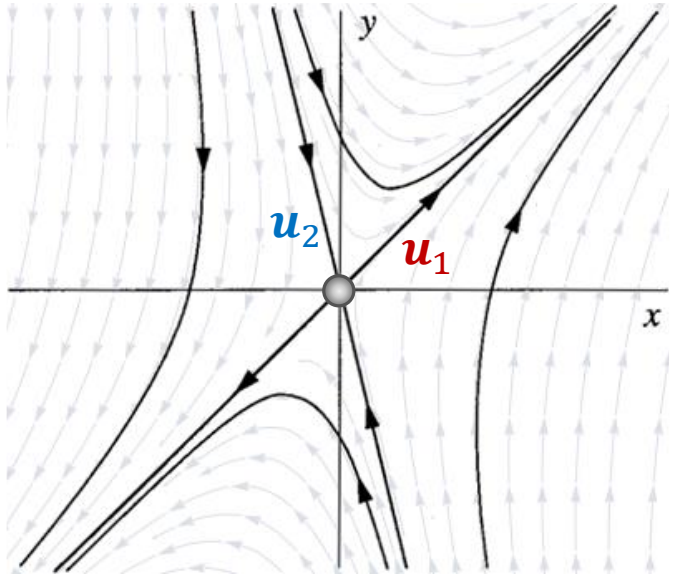
  For $\lambda_1 = 2$, this yields $\boldsymbol{u}_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$

  For $\lambda_2 = -3$, this yields $\boldsymbol{u}_2 = \begin{pmatrix} 1 \\ -4 \end{pmatrix}$



Vector field topology (2D)

- Mapping to graphical primitives: streamlines
  - Start streamlines close to critical points
  - Initial direction along the eigenvectors (forward+backward integration)
- End particle tracing at
  - Other “real” critical points
  - Interior boundaries
  - Boundaries of computational domain



How to find & classify critical points?

- **Example:** Vector field $\boldsymbol{v}(x,y) = (-x^2 + xy + \frac{1}{2}y + \frac{1}{4},\ x^2 + xy - \frac{1}{2}y - \frac{1}{4})$
- How to find critical points analytically? → $\boldsymbol{v}(\boldsymbol{x}^*) = 0$

$$I: -x^2 + xy + \frac{1}{2}y + \frac{1}{4} = 0$$

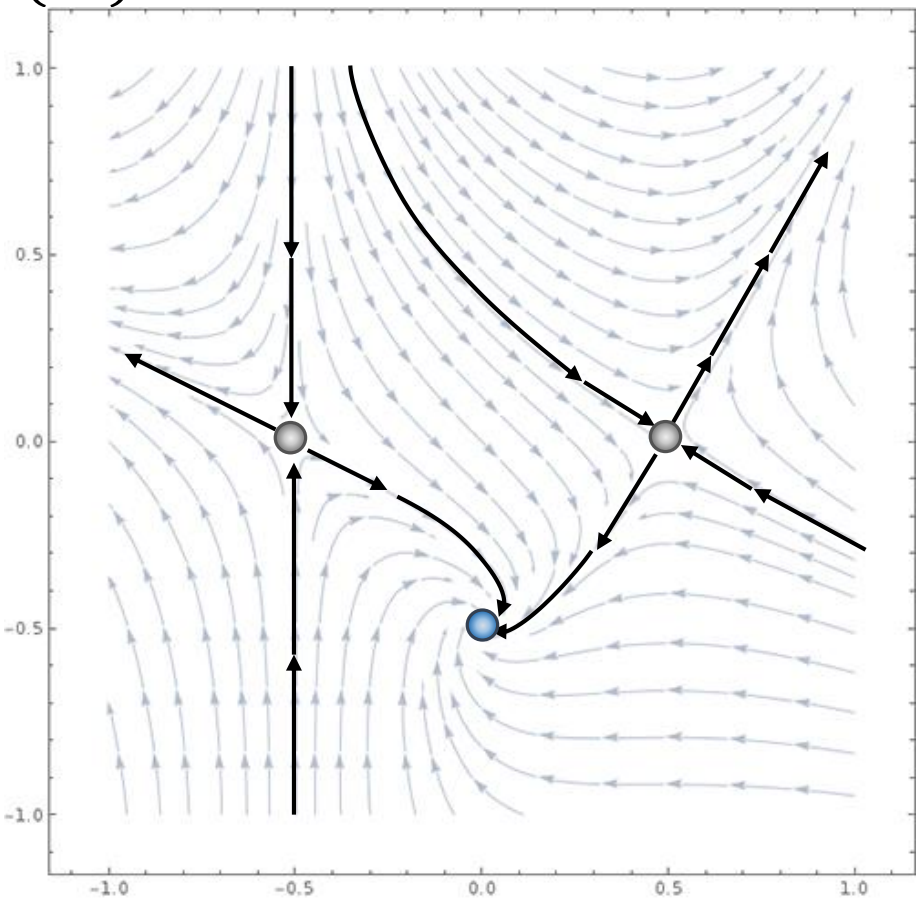
$$II: \quad x^2 + xy - \frac{1}{2}y - \frac{1}{4} = 0$$

$$I + II: \quad 2xy = 0$$

Solution: $x_1 = 0 \rightarrow \frac{1}{2}y + \frac{1}{4} = 0,\ y_1 = -\frac{1}{2}$

Solution: $y_{2,3} = 0 \rightarrow x^2 - \frac{1}{4} = 0,\ x_{2,3} = \pm\frac{1}{2}$

3 critical points: $\left(0, -\frac{1}{2}\right), \left(\frac{1}{2}, 0\right), \left(-\frac{1}{2}, 0\right)$



How to find & classify critical points?

- **Example:** Vector field $\boldsymbol{v}(x,y) = (-x^2 + xy + \frac{1}{2}y + \frac{1}{4},\ x^2 + xy - \frac{1}{2}y - \frac{1}{4})$

3 critical points: $\left(0, -\frac{1}{2}\right), \left(\frac{1}{2}, 0\right), \left(-\frac{1}{2}, 0\right)$

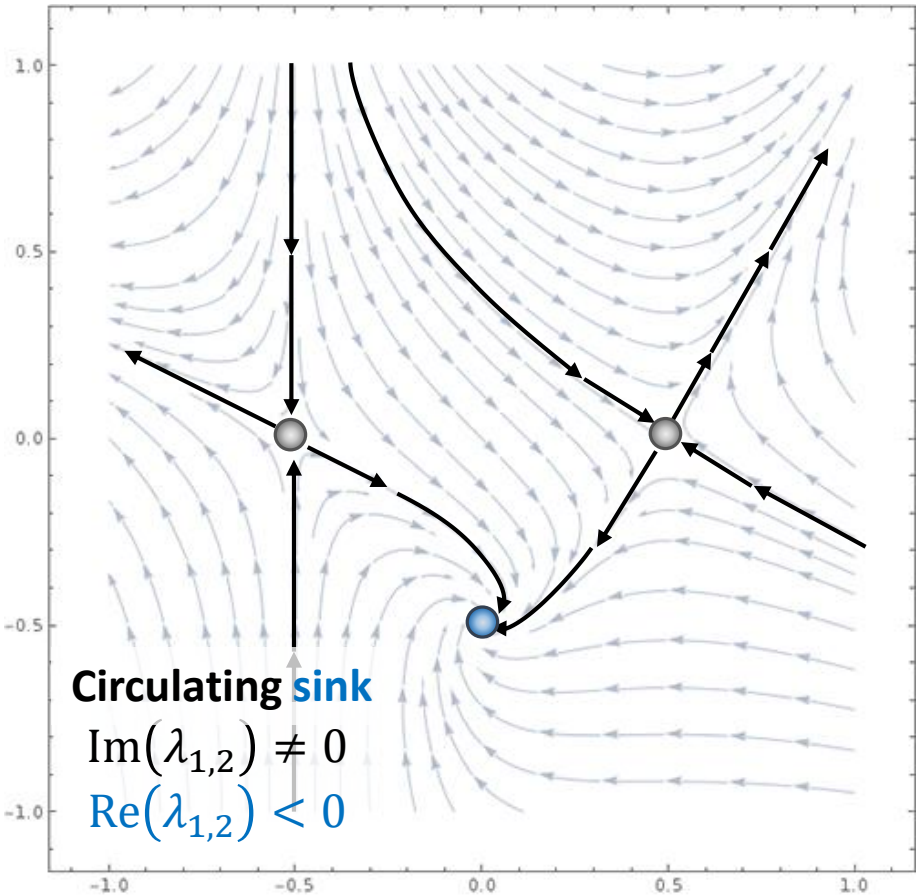
- We compute $\mathbf{J} = \begin{pmatrix} -2x + y & x + 0.5 \\ 2x + y & x - 0.5 \end{pmatrix}$
- Evaluate $\mathbf{J}$ at the critical point $\left(0, -\frac{1}{2}\right)$

$$\det\begin{pmatrix} -0.5 - \lambda & 0.5 \\ -0.5 & -0.5 - \lambda \end{pmatrix} = \lambda^2 + \lambda + \frac{1}{2}$$

$$\rightarrow \lambda^2 + \lambda + \frac{1}{2} = 0$$

$$\rightarrow \lambda_{1,2} = \frac{-1 \pm \sqrt{1-2}}{2}$$

We get eigenvalues $\lambda_{1,2} = -\frac{1}{2} \pm \frac{1}{2}i$



How to find & classify critical points?

- **Example:** Vector field $\boldsymbol{v}(x,y) = (-x^2 + xy + \frac{1}{2}y + \frac{1}{4},\ x^2 + xy - \frac{1}{2}y - \frac{1}{4})$

3 critical points: $\left(0, -\frac{1}{2}\right), \left(\frac{1}{2}, 0\right), \left(-\frac{1}{2}, 0\right)$

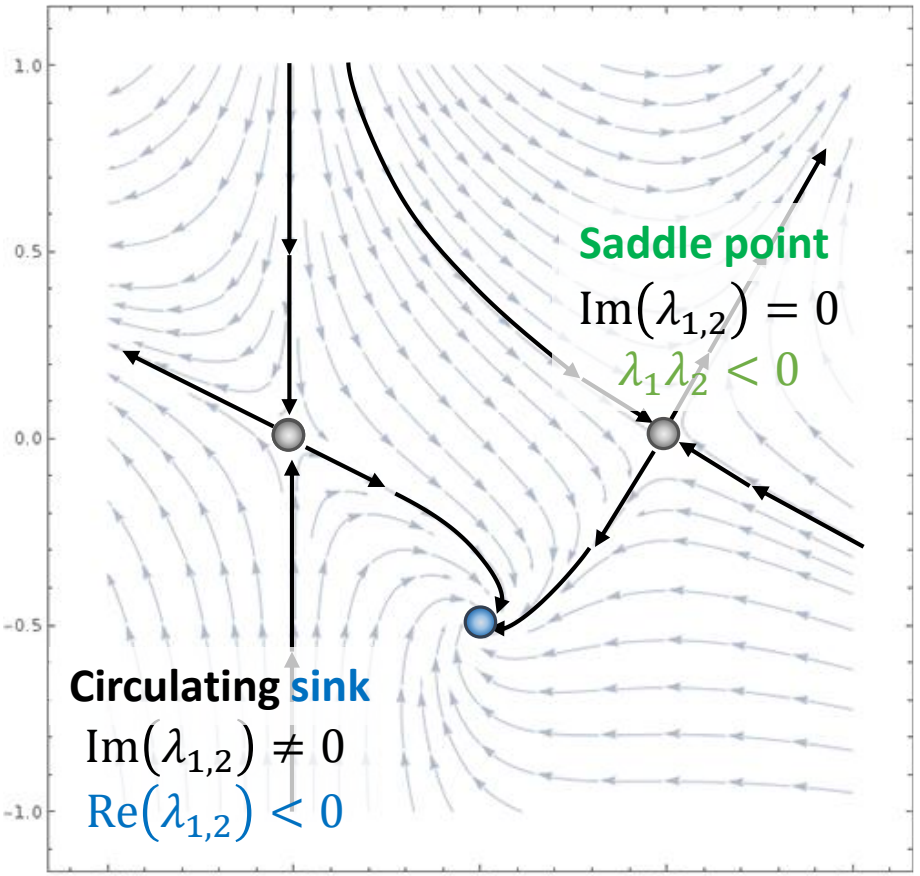
- We compute $\mathbf{J} = \begin{pmatrix} -2x + y & x + 0.5 \\ 2x + y & x - 0.5 \end{pmatrix}$
- Evaluate $\mathbf{J}$ at the critical point $\left(\frac{1}{2}, 0\right)$

$$\det\left[\begin{pmatrix} -1-\lambda & 1 \\ 1 & -\lambda \end{pmatrix}\right] = \lambda^2 + \lambda - 1$$

$$\rightarrow \lambda^2 + \lambda - 1 = 0$$

$$\rightarrow \lambda_{1,2} = \frac{-1 \pm \sqrt{1+4}}{2}$$

We get eigenvalues $\lambda_{1,2} = \frac{-1 \pm \sqrt{5}}{2}$



How to find & classify critical points?

- **Example:** Vector field $\boldsymbol{v}(x,y) = (-x^2 + xy + \frac{1}{2}y + \frac{1}{4},\ x^2 + xy - \frac{1}{2}y - \frac{1}{4})$

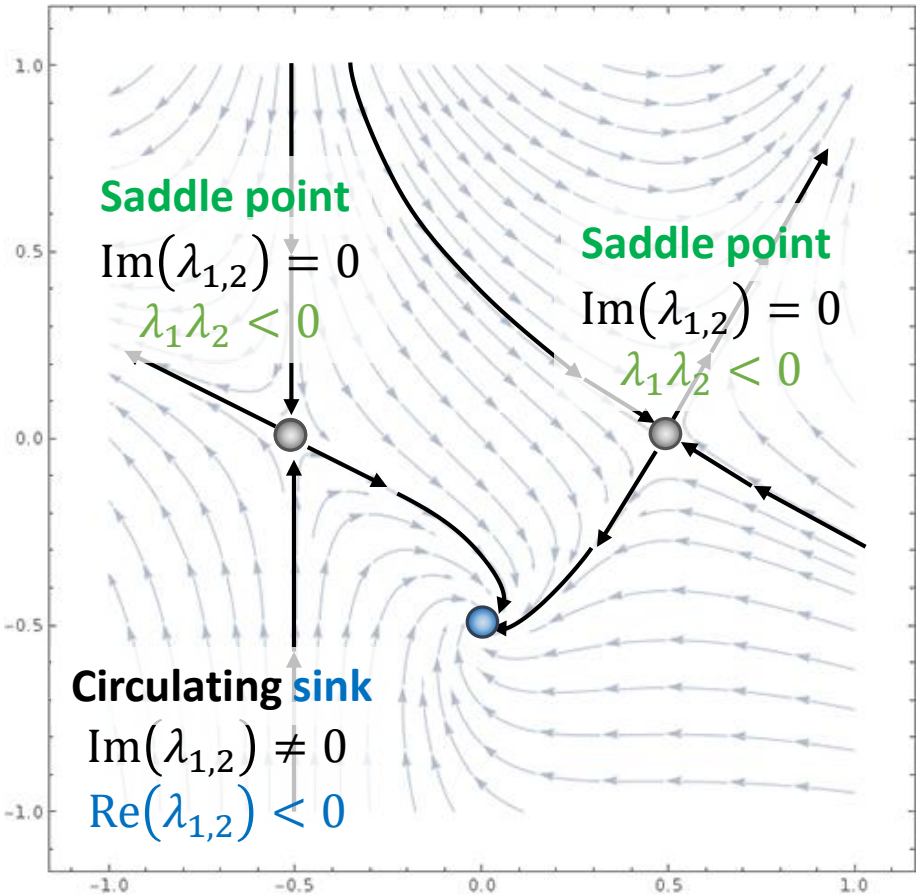
3 critical points: $\left(0, -\frac{1}{2}\right), \left(\frac{1}{2}, 0\right), \left(-\frac{1}{2}, 0\right)$

- We compute $\mathbf{J} = \begin{pmatrix} -2x + y & x + 0.5 \\ 2x + y & x - 0.5 \end{pmatrix}$
- Evaluate $\mathbf{J}$ at the critical point $\left(-\frac{1}{2}, 0\right)$

$$\det\begin{pmatrix} 1-\lambda & 0 \\ -1 & -1-\lambda \end{pmatrix} = \lambda^2 - 1$$

$$\rightarrow \lambda^2 - 1 = 0$$

We get eigenvalues $\lambda_{1,2} = \pm 1$



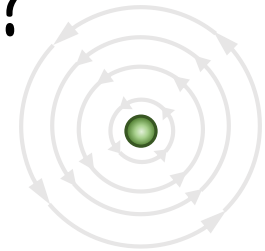
How to find & classify critical points?

Summary

- How to find critical points $\boldsymbol{x}^*$?
  - Points where $\boldsymbol{v}(\boldsymbol{x}^*) = 0$
- How to classify critical points $\boldsymbol{x}^*$?
  - Jacobi matrix $\mathbf{J} = \begin{pmatrix} \frac{\partial}{\partial x}\boldsymbol{v}_x & \frac{\partial}{\partial y}\boldsymbol{v}_x \\ \frac{\partial}{\partial x}\boldsymbol{v}_y & \frac{\partial}{\partial y}\boldsymbol{v}_y \end{pmatrix}$ governs the behavior near $\boldsymbol{x}^*$
  - For each $\boldsymbol{x}^*$, calculate eigenvalues λ_1, λ_2 of Jacobi matrix $\mathbf{J}$
  - Solve equation $\det(\mathbf{J} - \lambda\mathbf{I}) = 0$

How to find & classify critical points?

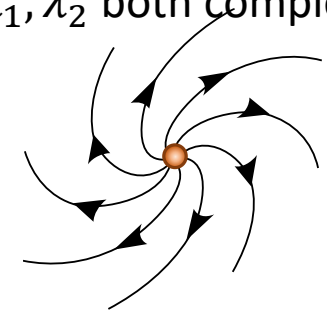
- Classify critical points by eigenvalue analysis
  - λ_1, λ_2 both positive → local repulsion (source)
  - λ_1, λ_2 both negative → local attraction (sink)
  - $\lambda_1 \lambda_2 < 0$ → saddle point
  - λ_1, λ_2 both complex → rotation around $\boldsymbol{x}^*$



Center

$\mathrm{Im}(\lambda_{1,2}) \neq 0$

$\mathrm{Re}(\lambda_{1,2}) = 0$

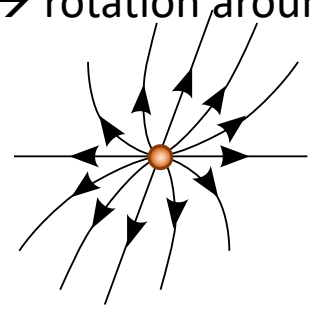


Circulating source
(repelling focus)

$\mathrm{Im}(\lambda_{1,2}) \neq 0$

$\mathrm{Re}(\lambda_{1,2}) > 0$

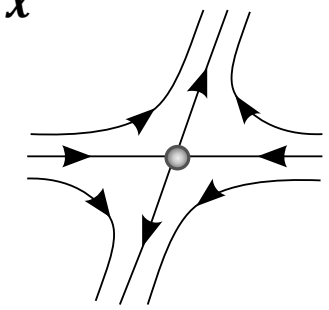
$\lambda_{1,2} = a \pm bi$



Noncirculating source
(repelling node)

$\mathrm{Im}(\lambda_{1,2}) = 0$

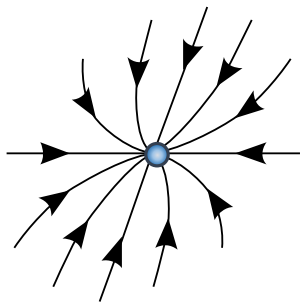
$\mathrm{Re}(\lambda_{1,2}) > 0$



Saddle point

$\mathrm{Im}(\lambda_{1,2}) = 0$

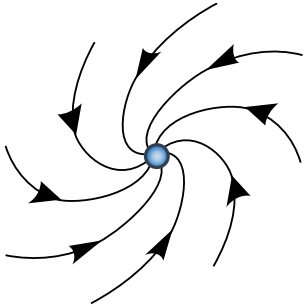
$\lambda_1 \lambda_2 < 0$



Noncirculating sink
(attracting node)

$\mathrm{Im}(\lambda_{1,2}) = 0$

$\mathrm{Re}(\lambda_{1,2}) < 0$



Circulating sink
(attracting focus)

$\mathrm{Im}(\lambda_{1,2}) \neq 0$

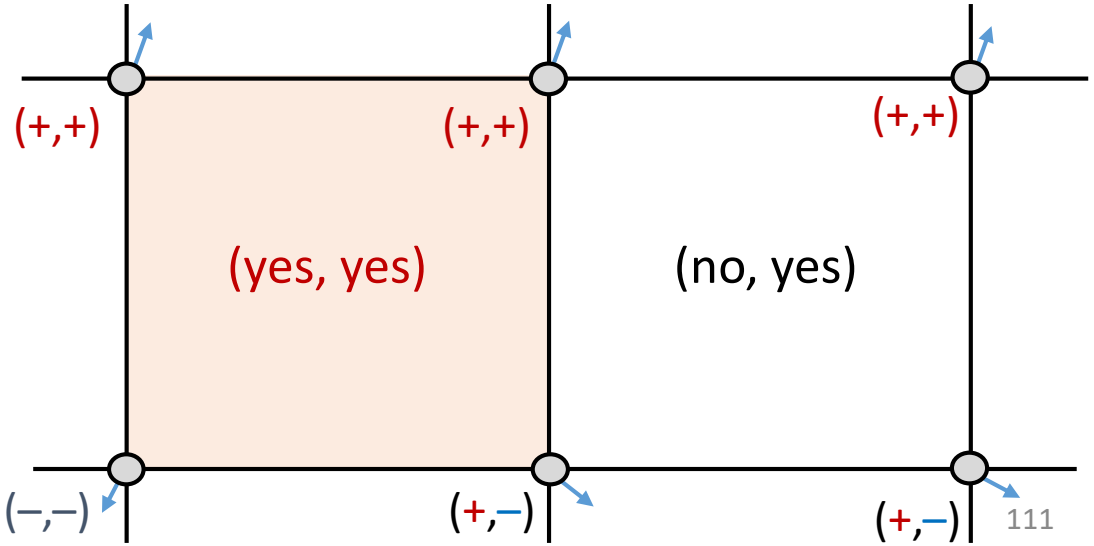
$\mathrm{Re}(\lambda_{1,2}) < 0$

$\lambda_{1,2} = a \pm bi$

How to find critical points?

How to find critical points in sampled data?

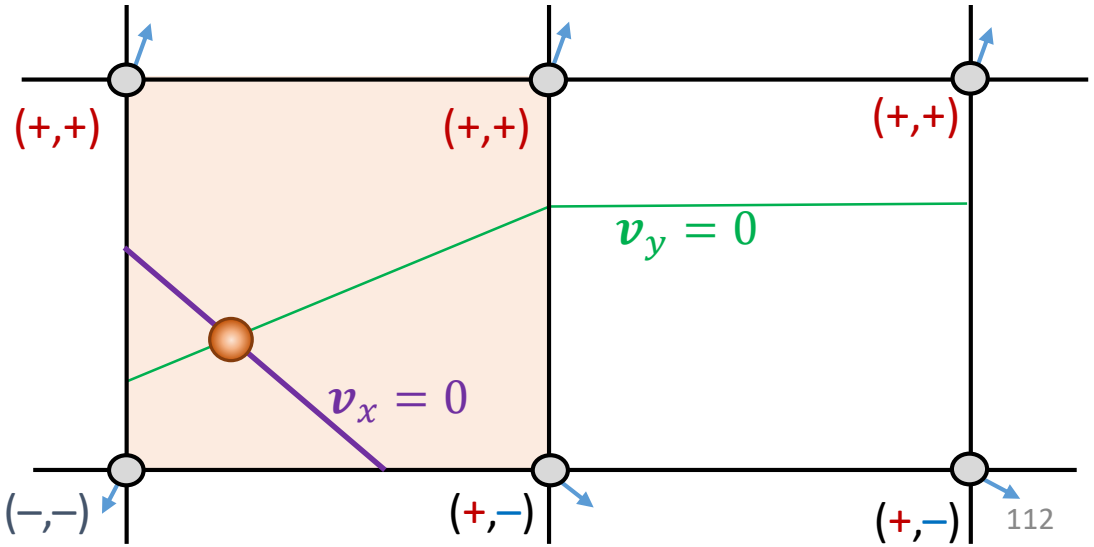
- Cell search (for cells that contain critical points)
  - Mark vertices by (+, +), (–, –), (+, –) or (–,+), depending on the signs of $\boldsymbol{v}_x$ and $\boldsymbol{v}_y$
  - Determine cells with vertices where the sign changes in both components → these are cells that contain critical points
  - Find the critical points by interpolation



[Effenberg & Weiskopf 10]

How to find critical points?

- How to find critical points within a cell?
  - Approximation: Determine the intersection of the isolines (c = 0) of the two vector components



[Effenberg & Weiskopf 10]

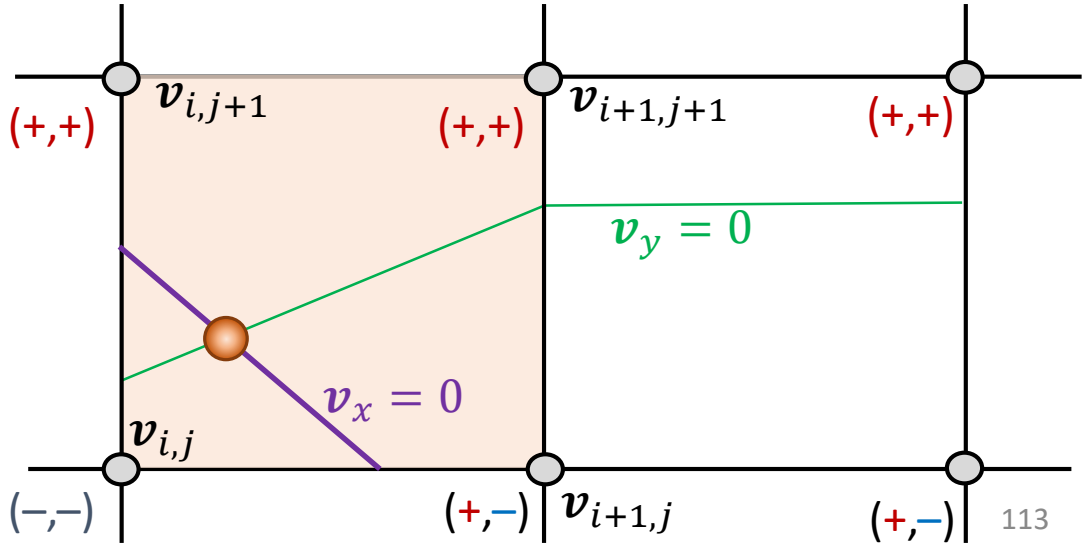
How to find critical points?

- How to find critical points within a cell?
  - Approximation: Determine the intersection of the isolines (c = 0) of the two vector components
  - True solution: Solve two bilinear (or one quadratic) equation(s)

$$(1-\alpha)(1-\beta)\boldsymbol{v}_{i,j} + \alpha(1-\beta)\boldsymbol{v}_{i+1,j} + (1-\alpha)\,\beta\,\boldsymbol{v}_{i,j+1} + \alpha\,\beta\,\boldsymbol{v}_{i+1,j+1} = 0$$

  - Critical points are the (real) solutions within the cell boundaries

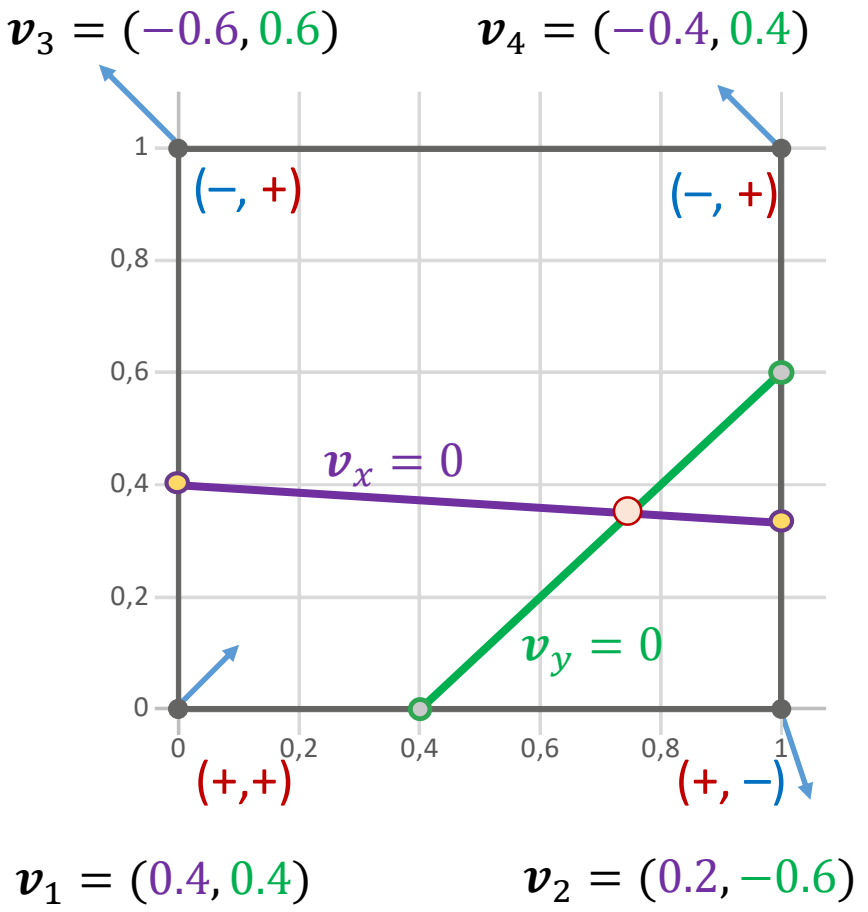
$$\alpha, \beta \in [0,1]$$



[Effenberg & Weiskopf 10]

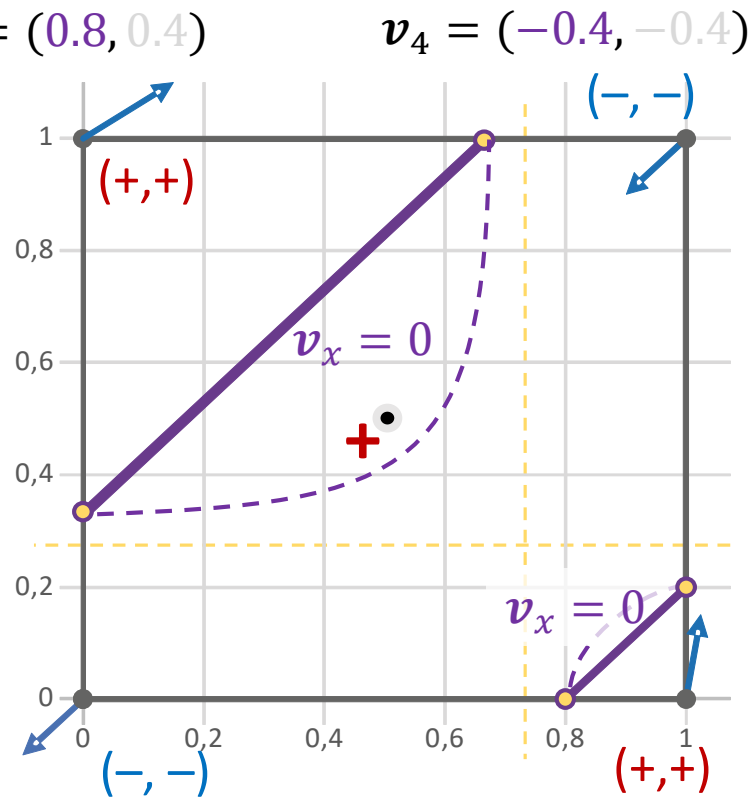
How to find critical points?

- Example
  - Mark vertices by (+, +), (–, –), (+, –) or (–,+), depending on the signs of $\boldsymbol{v}_x$ and $\boldsymbol{v}_y$
  - Sign changes in both components
  - Evaluate isoline $\boldsymbol{v}_x = 0$
  - Evaluate isoline $\boldsymbol{v}_y = 0$
  - Determine intersection of isolines



How to find critical points?

Example

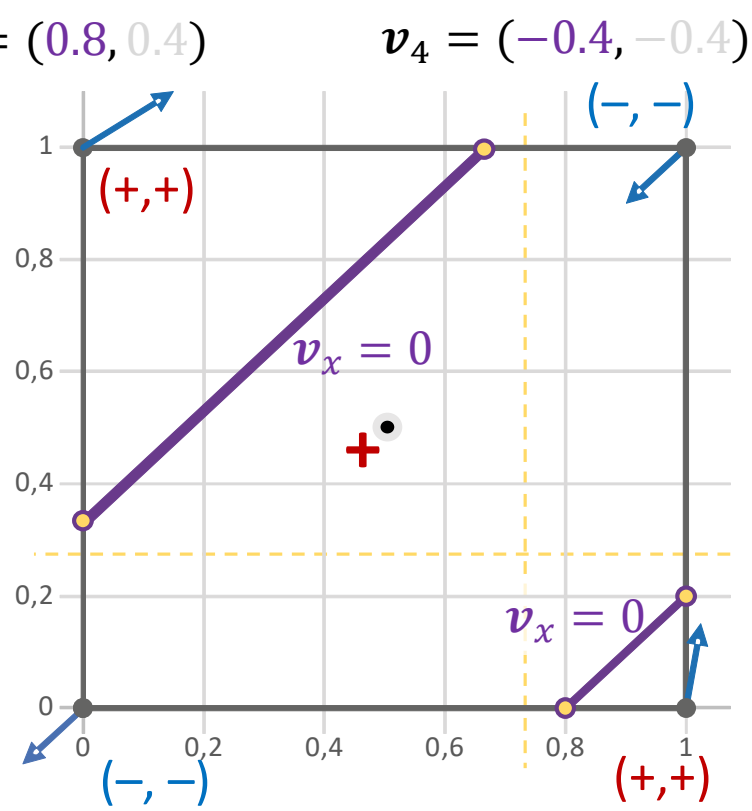


Midpoint decider does not work in this case
→ Consider the order of intersection points

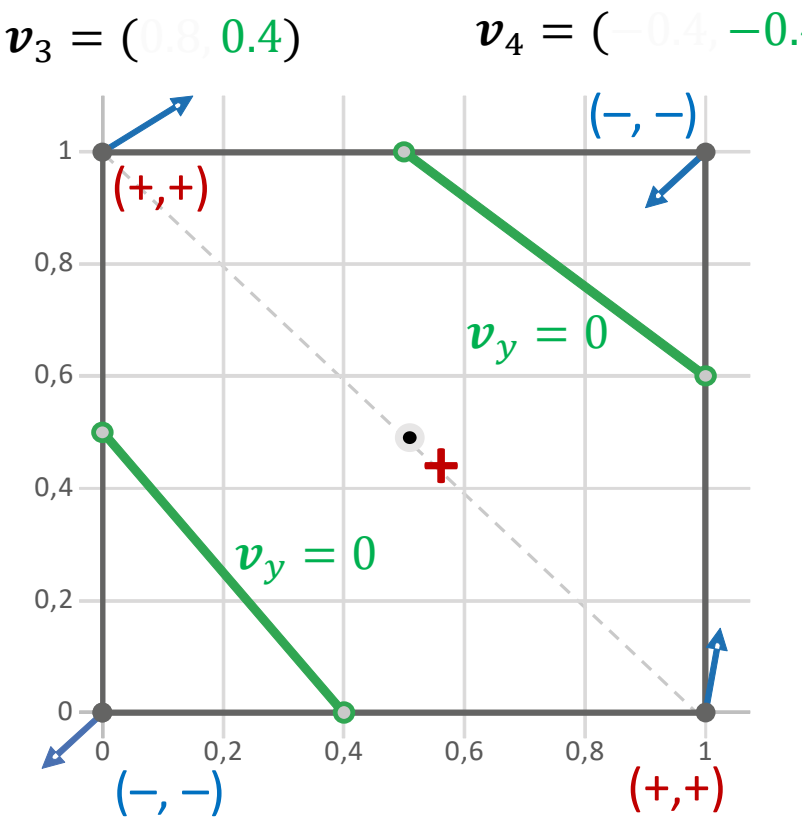
Evaluate isoline $v_x = 0$

How to find critical points?

Example



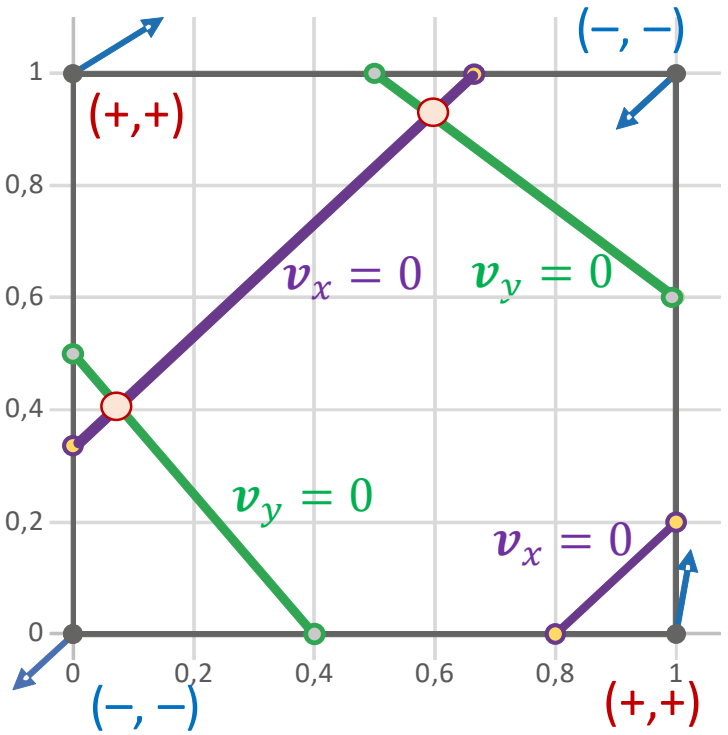
Evaluate isoline $v_x = 0$



Evaluate isoline $v_y = 0$

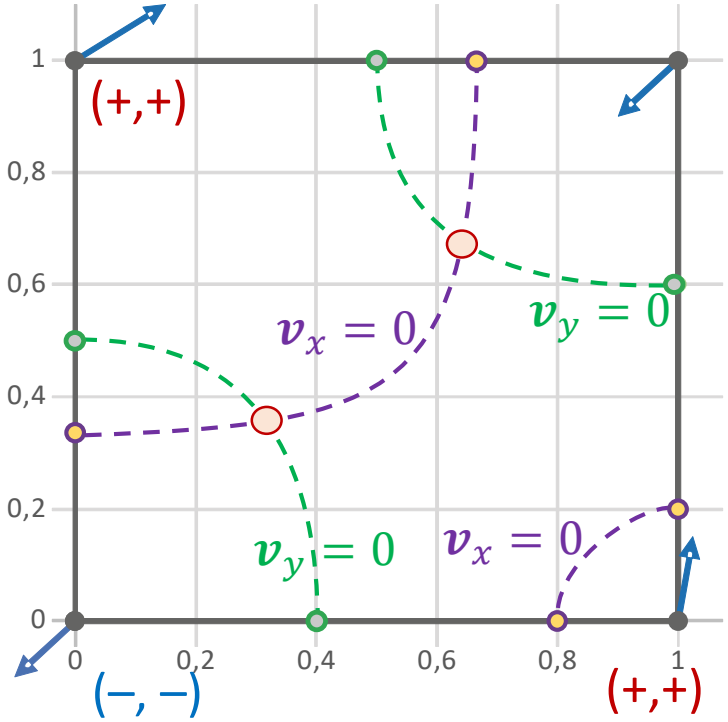
How to find critical points?

Example



Determine intersection of isolines

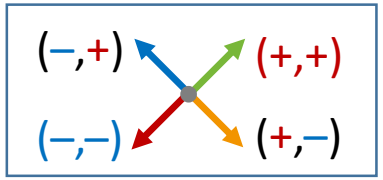
Saddle + **source** or **sink**

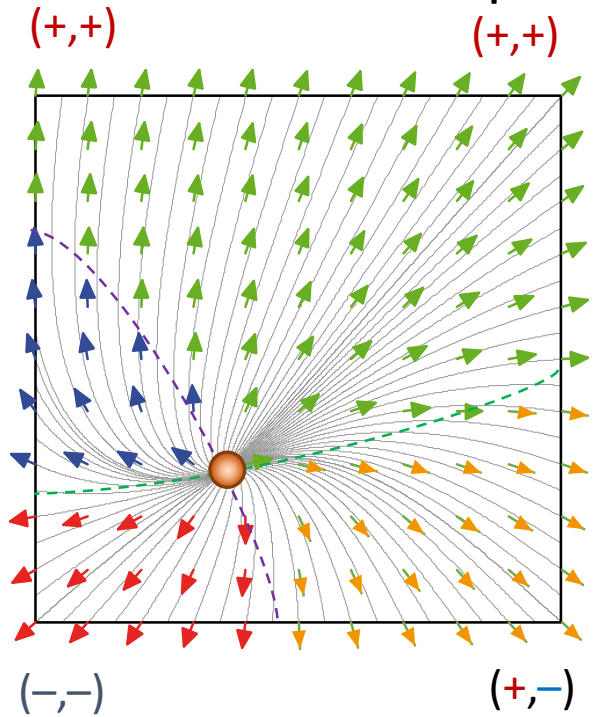


Note: Point locations are only an approximation, because true isolines are hyperbolas

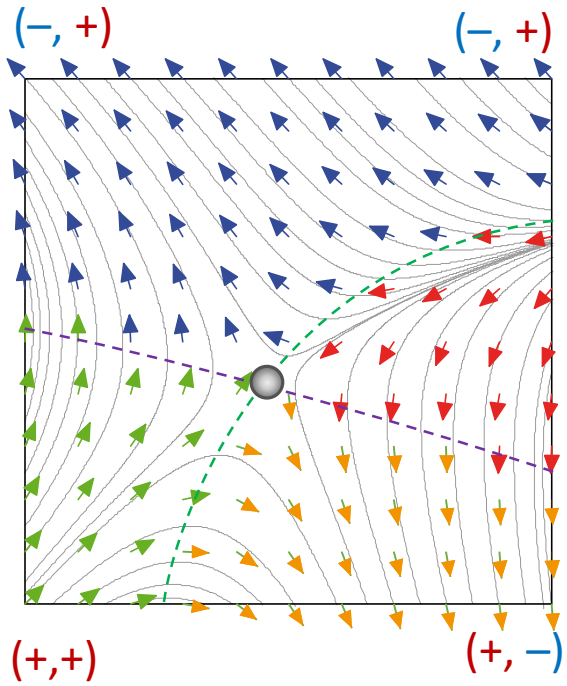
How to find critical points?

- Examples of sign configurations
  - 74 representative classes with one or two critical points within a cell

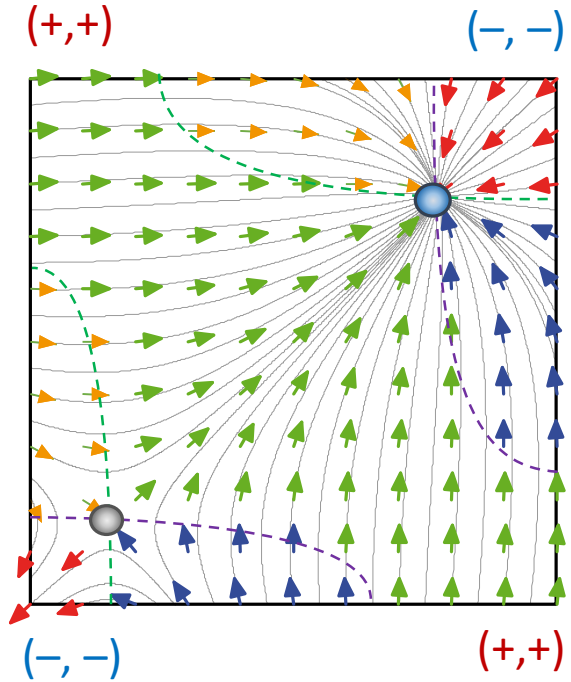




Source



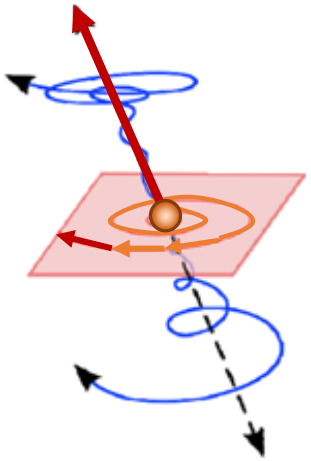
Saddle



Saddle + sink

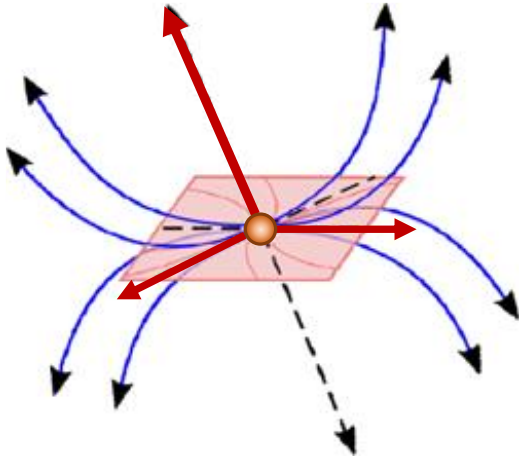
Vector field topology (3D)

- Critical points in 3D
  - More complicated
  - Line and surface separatices exist
- Examples



Spiral source

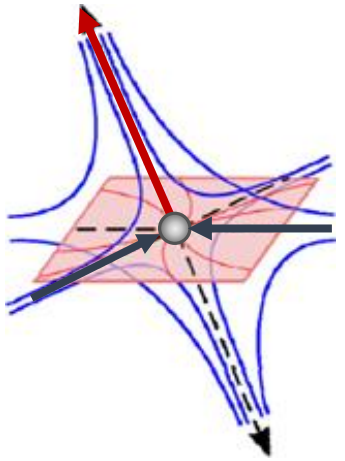
$\mathrm{Im}(\lambda_1) = 0,$
$\mathrm{Im}(\lambda_{2,3}) \neq 0$
$\mathrm{Re}(\lambda_{1,2,3}) > 0$



Noncirculating source

$\mathrm{Im}(\lambda_{1,2,3}) = 0$

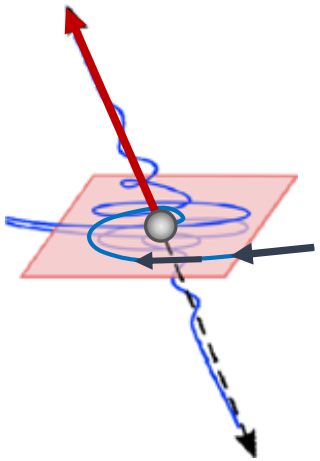
$\mathrm{Re}(\lambda_{1,2,3}) > 0$



Noncirculating saddle

$\mathrm{Im}(\lambda_{1,2,3}) = 0$

$\mathrm{Re}(\lambda_1) > 0,$
$\mathrm{Re}(\lambda_{2,3}) < 0$



Spiral saddle

$\mathrm{Im}(\lambda_1) = 0,$
$\mathrm{Im}(\lambda_{2,3}) \neq 0$
$\mathrm{Re}(\lambda_1) > 0,$
$\mathrm{Re}(\lambda_{2,3}) < 0$

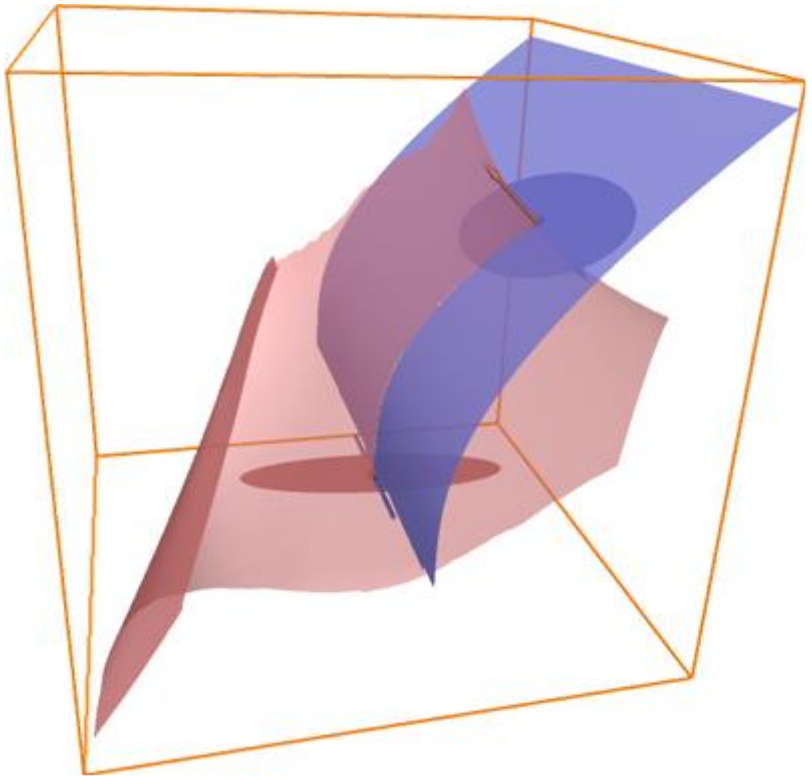
[Peikert & Sadlo 10]

Vector field topology (3D)

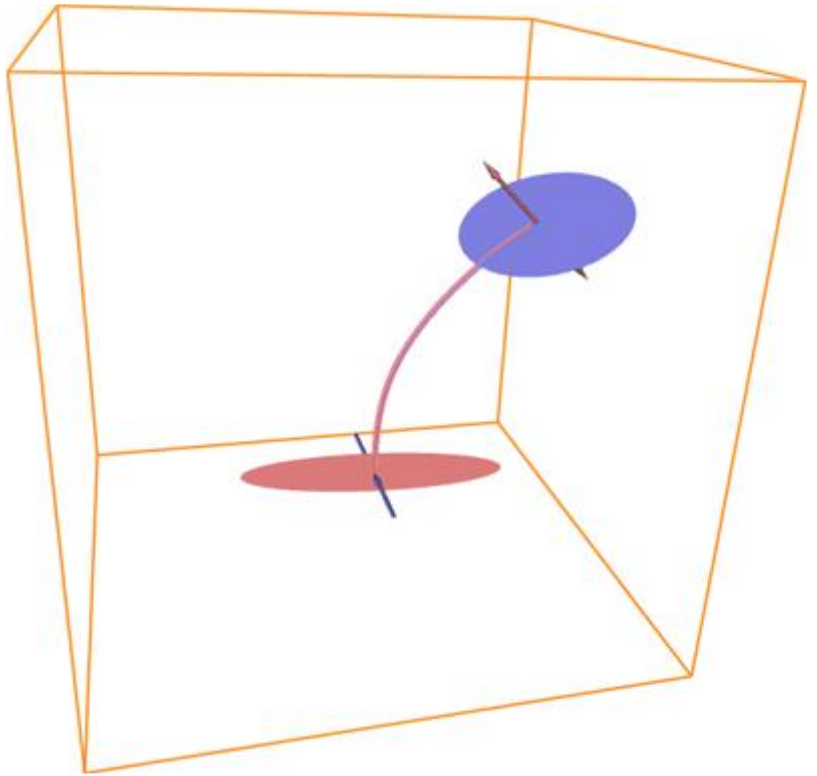
- Topology on surfaces
  - Critical points + separation lines applied to projections of vector field onto polygonal surface

Vector field topology (3D)

- Saddle connectors in 3D



Separation surfaces of two saddles

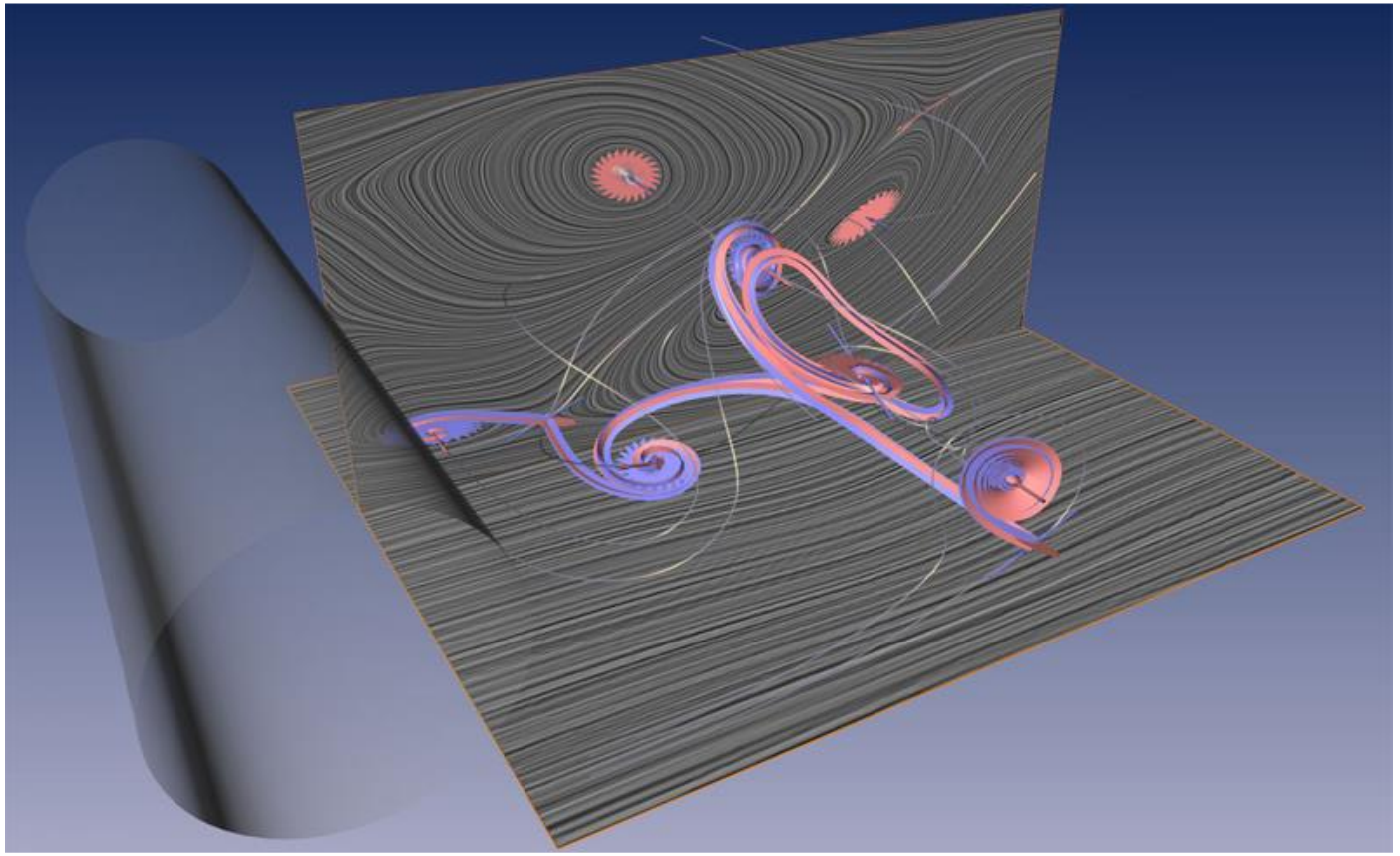


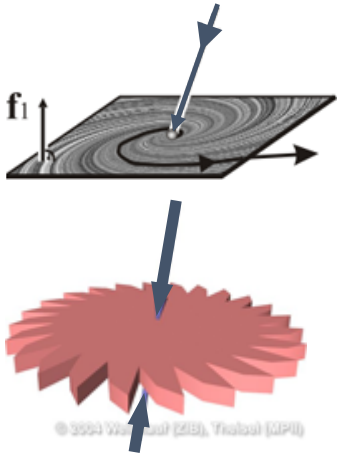
The intersection of the separation surfaces is the saddle connector

[Weinkauf & Theisel 05]

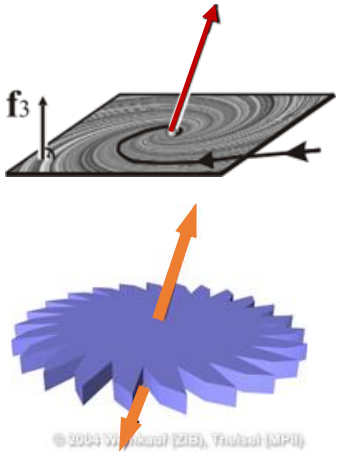
Vector field topology (3D)

- Saddle connectors in 3D





Repelling
spiral saddle



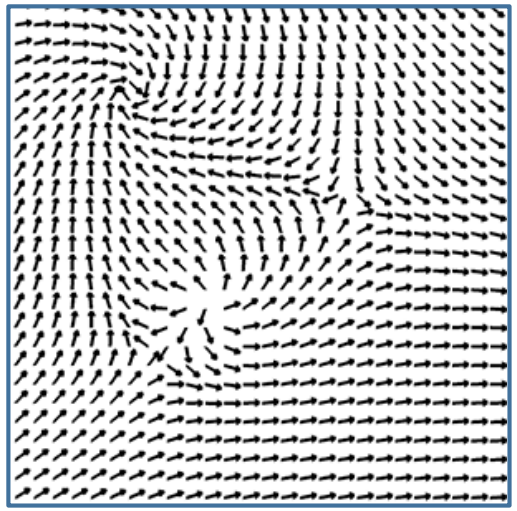
Attracting
spiral saddle

[Weinkauf & Theisel 05]

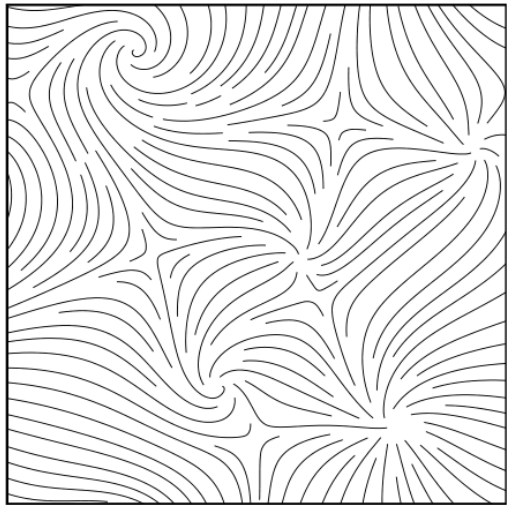
Vector field topology

- Summary
  - Draw only relevant stream lines (topological skeleton)
  - Partition domain into regions with similar flow features
  - Based on critical points
  - Good for 2D steady flows
  - Unsteady flows?
  - 3D?

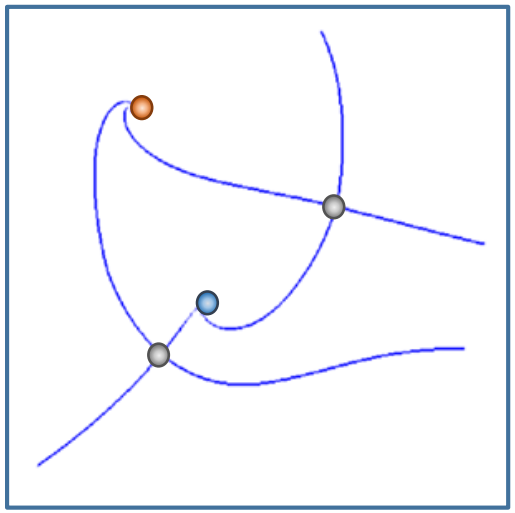
Flow visualization – Approaches



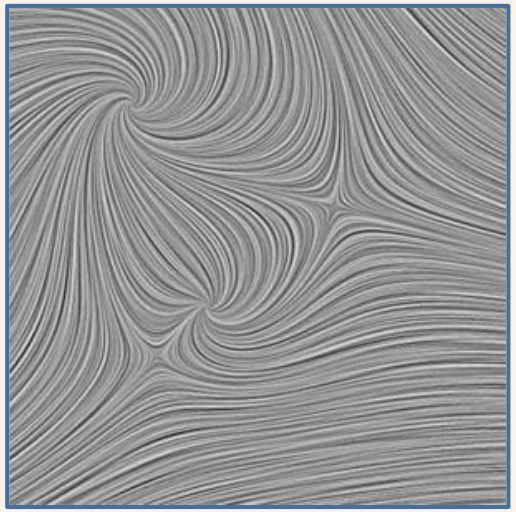
Direct flow visualization
(arrows, color coding, …)



Geometric flow visualization
(stream lines/surfaces, …)



Sparse (feature-based) vis.

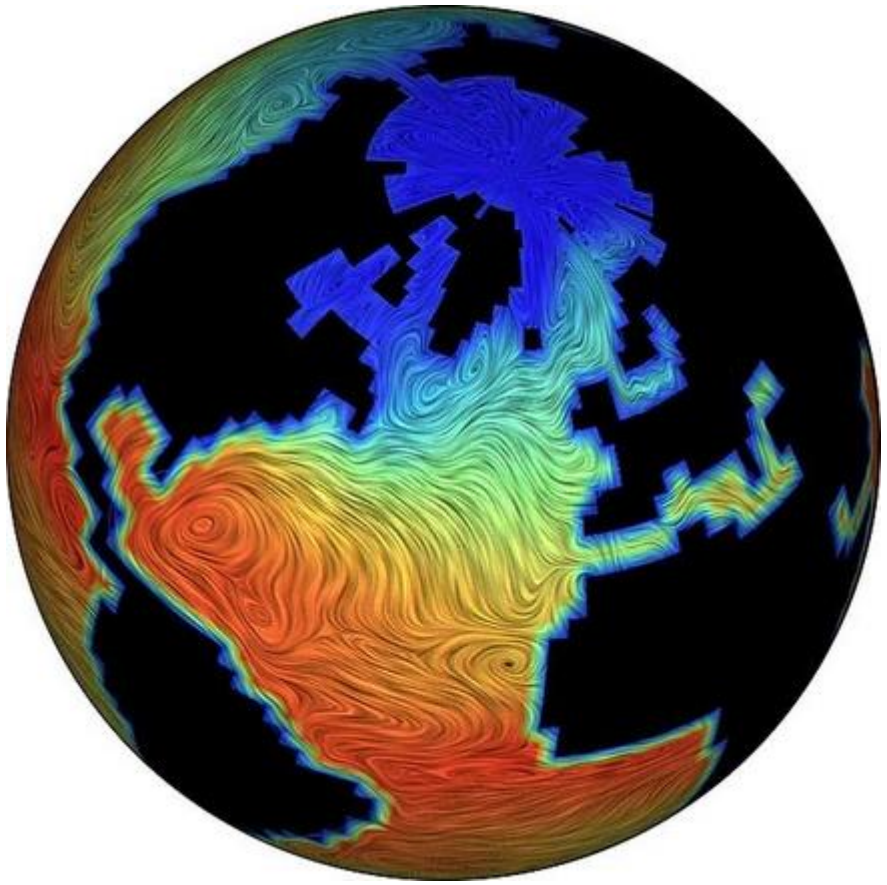


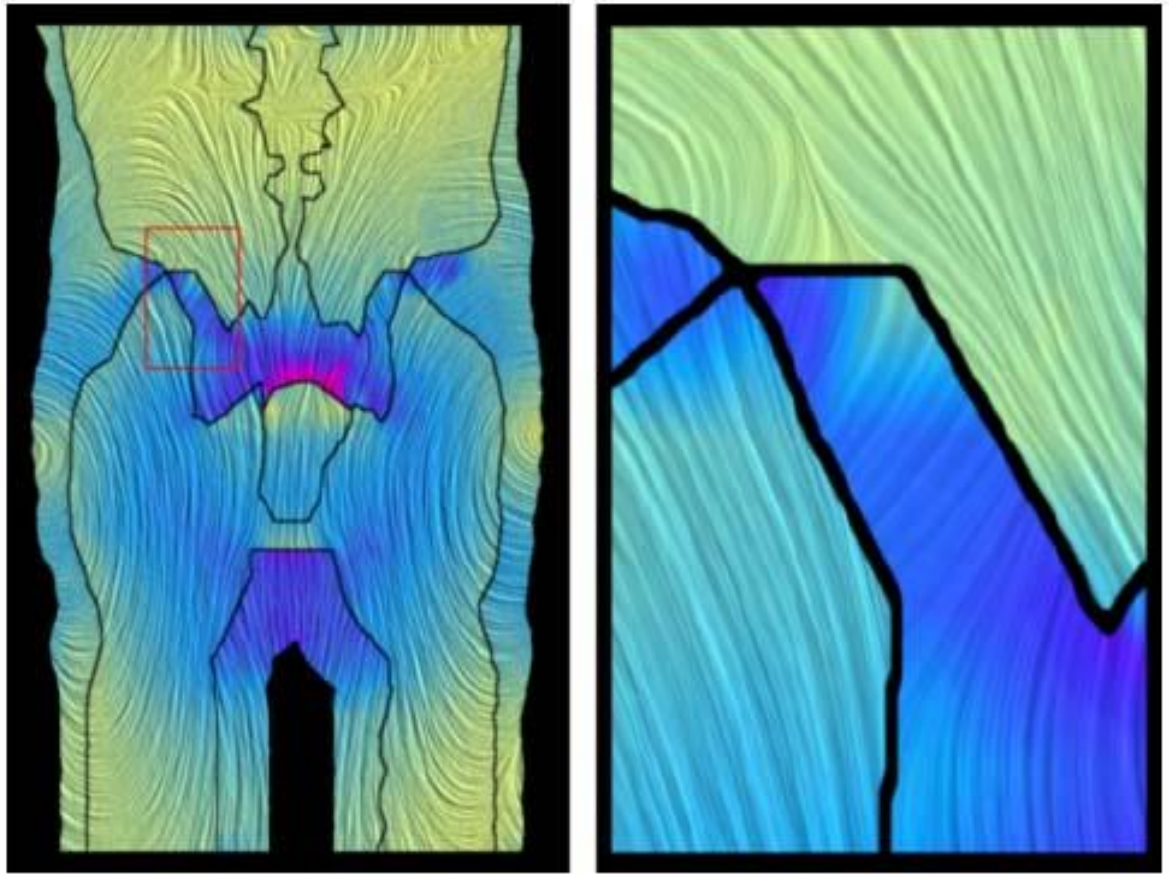
Dense (texture-based) vis.

Dense–based Visualization

Texture–based flow visualization

- Global method to visualize vector fields





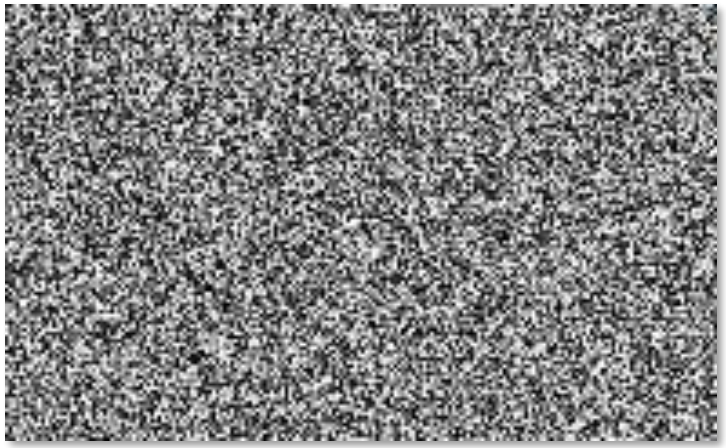
Why texture-based flow vis.

- Dense sampling
  - Better coverage of information
  - Critical point detection and classification
  - (Partially) solved problem of seeding

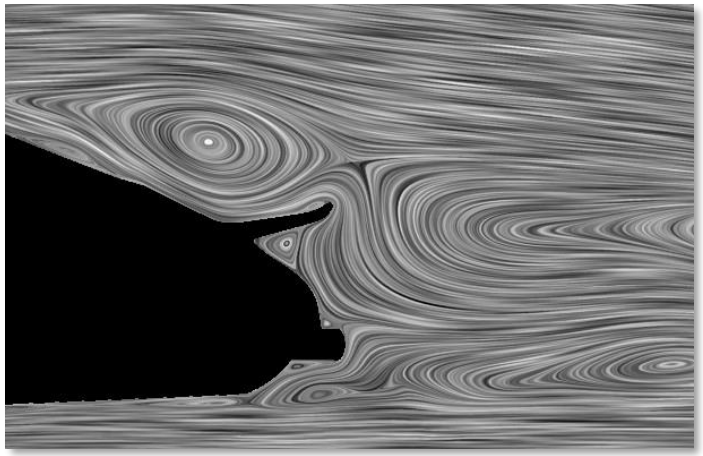
- Flexibility in visual representation
  - Good controllability of visual style
  - From line-like (crisp) to fuzzy

Line Integral Convolution

- Global visualization technique (not only one particle path)
- Start with a random texture (white noise)
- Smear out the texture along trajectories of vector field
- Results in low correlation between neighboring lines but high correlation along them (in flow direction)



White noise (no correlation)

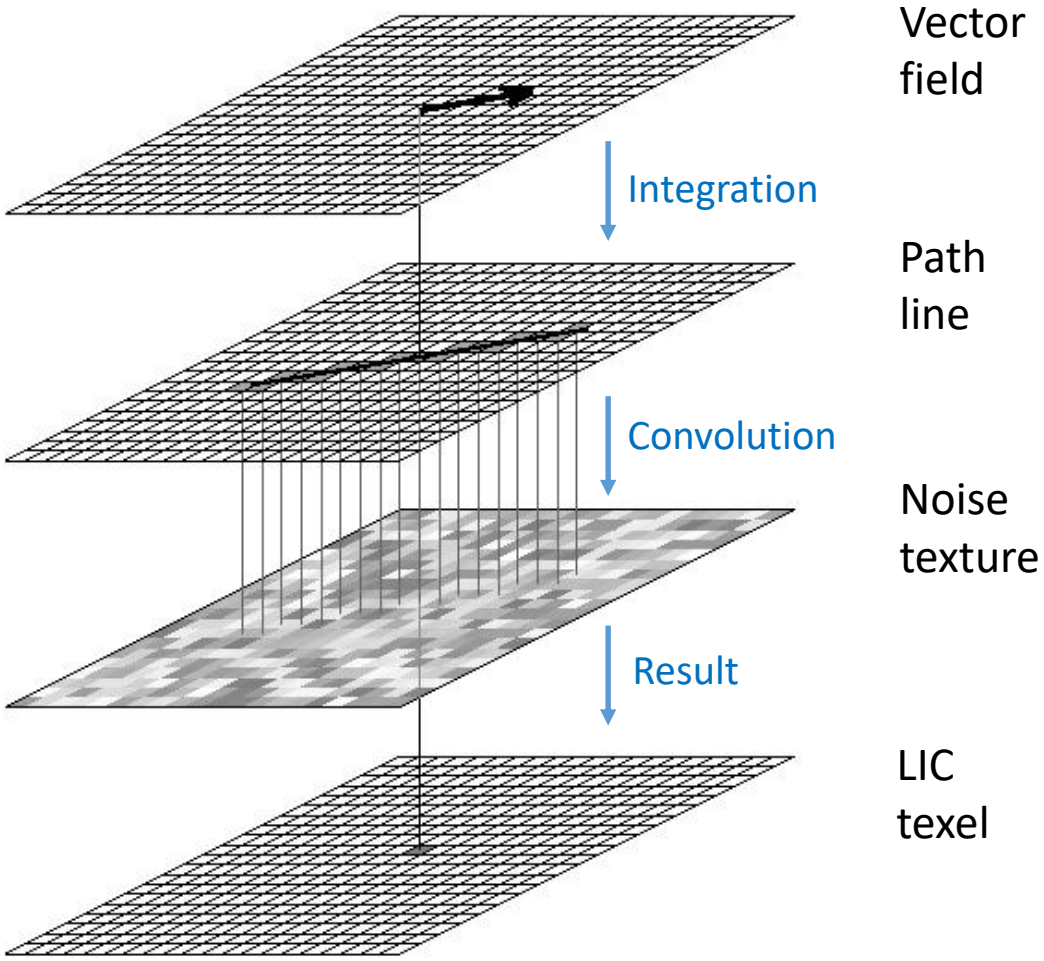


Texture values along trajectories are correlated (visually coherent)

[Cabral & Leedom 93]

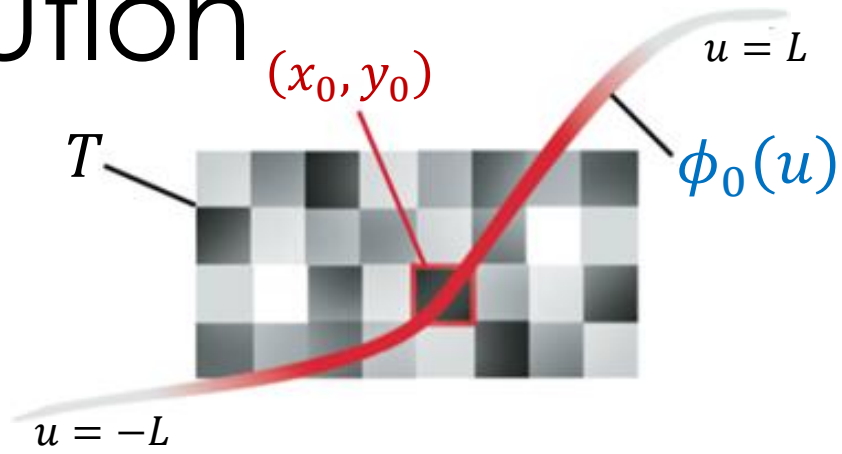
Line Integral Convolution

- Algorithm for 2D LIC
  - Smear out - convolve - noise texture along characteristic lines



[Cabral & Leedom 93]

Line Integral Convolution

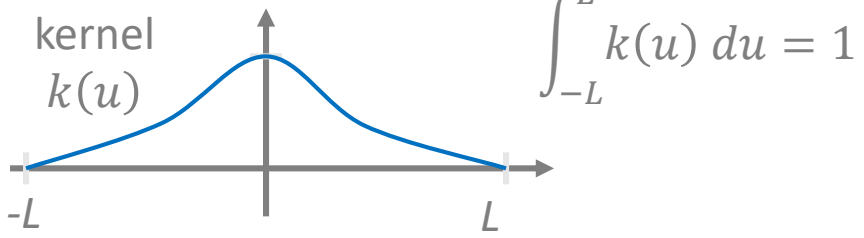


- Algorithm for 2D LIC
  - Let $\phi_0(u)$ be the stream line containing the point (x_0, y_0)
  - $T(x, y)$ is the randomly generated noise texture
  - Compute the intensity at (x_0, y_0) as

$$I(x_0, y_0) = \int_{-L}^{L} k(u) \cdot T\big(\phi_0(u)\big)\, du$$

convolution with a kernel $k(u)$

Smoothing filter kernel
- Finite support [-*L*, *L*]
- Normalized, usually symmetric
- E.g., Gaussian or box filter



$$\int_{-L}^{L} k(u)\, du = 1$$

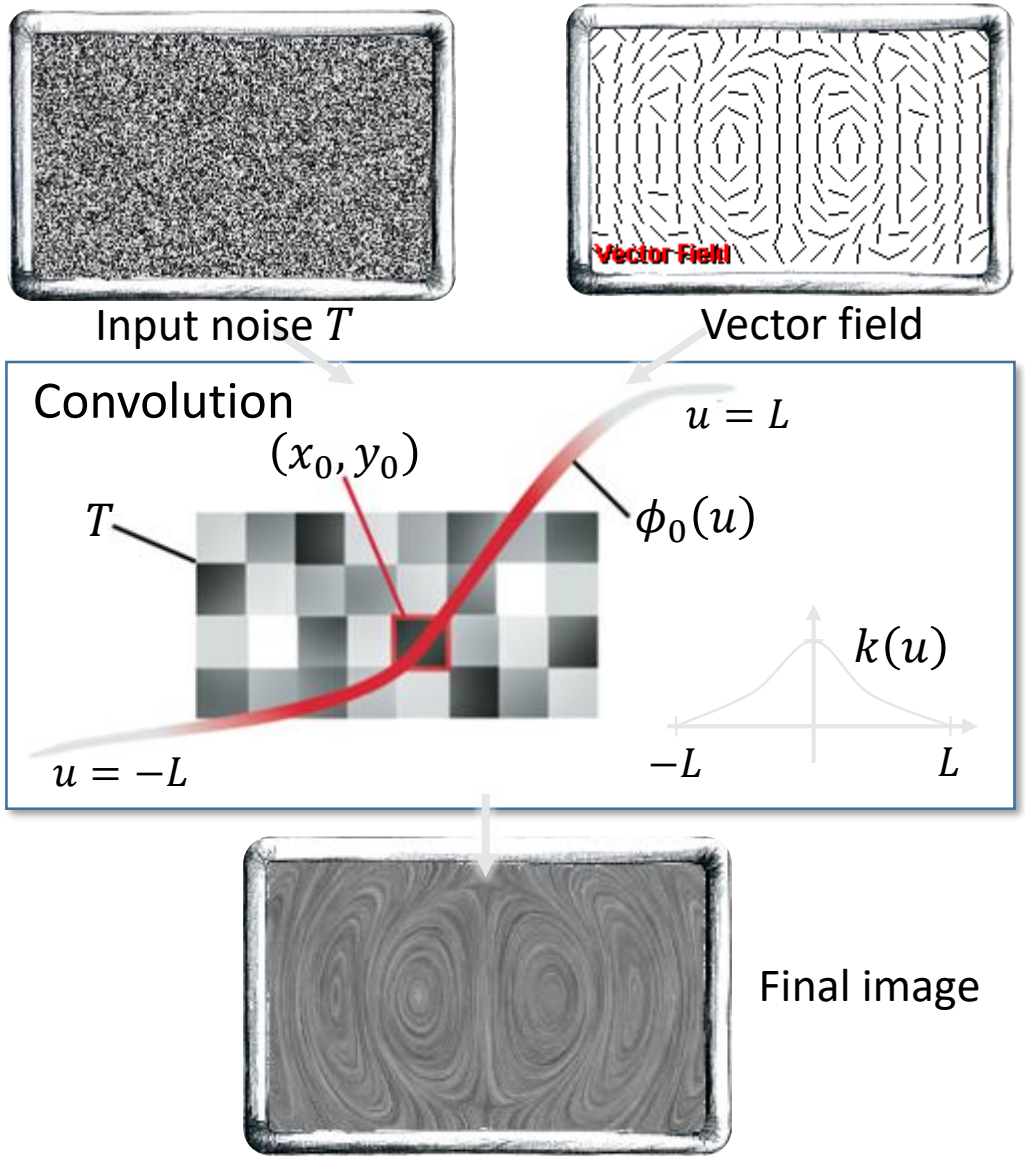
[Cabral & Leedom 93]

Line Integral Convolution

$$I(x_0, y_0) = \int_{-L}^{L} k(u) \cdot T\big(\phi_0(u)\big)\, du$$

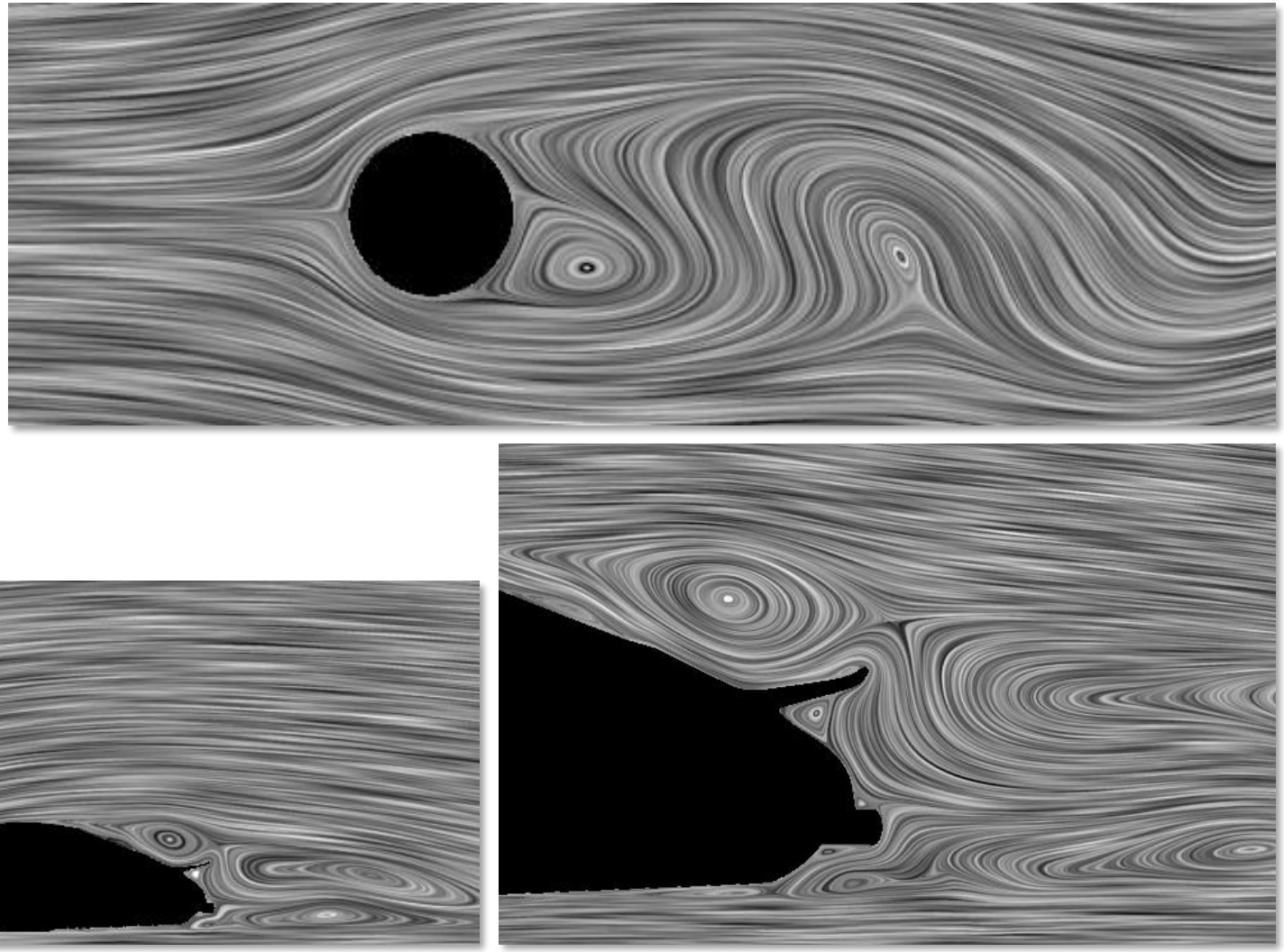
- LIC is a convolution of
  - a noise texture $T(x, y)$
  - and a smoothing filter $k(u)$

- Noise texture values are picked up along the stream line $\phi_0(u)$ through $T\big(\phi_0(u)\big)$



Line Integral Convolution

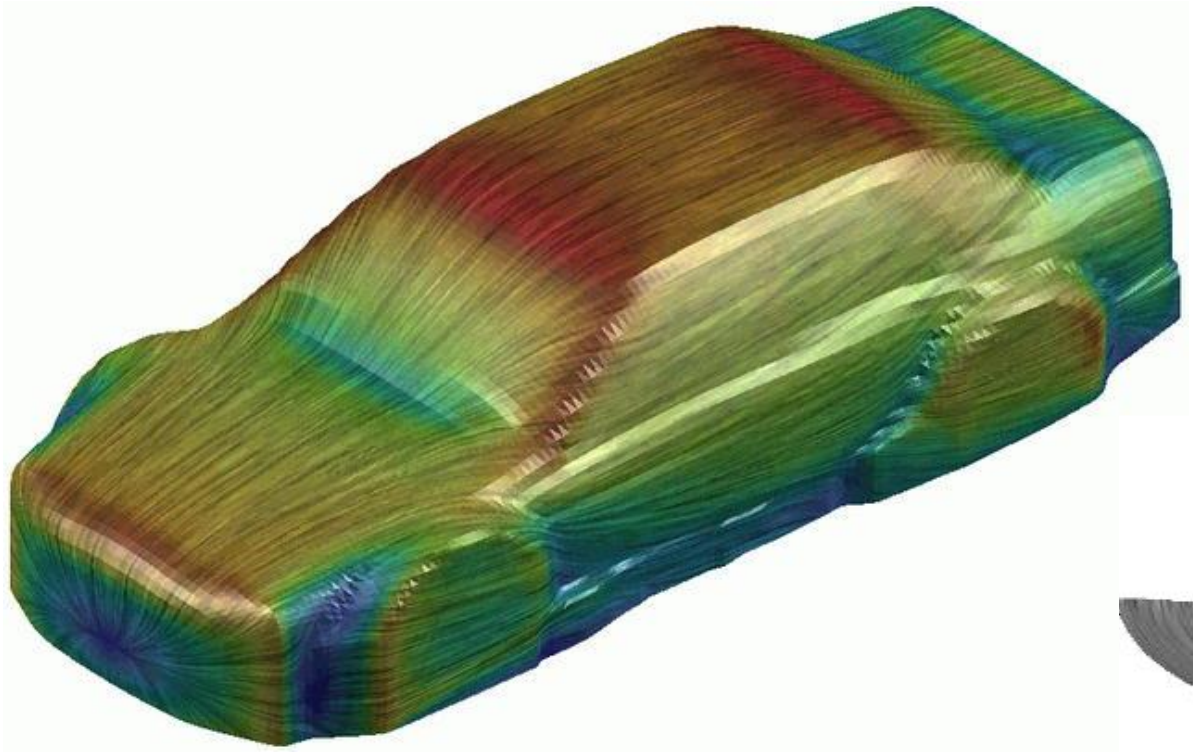
- LIC in 2D

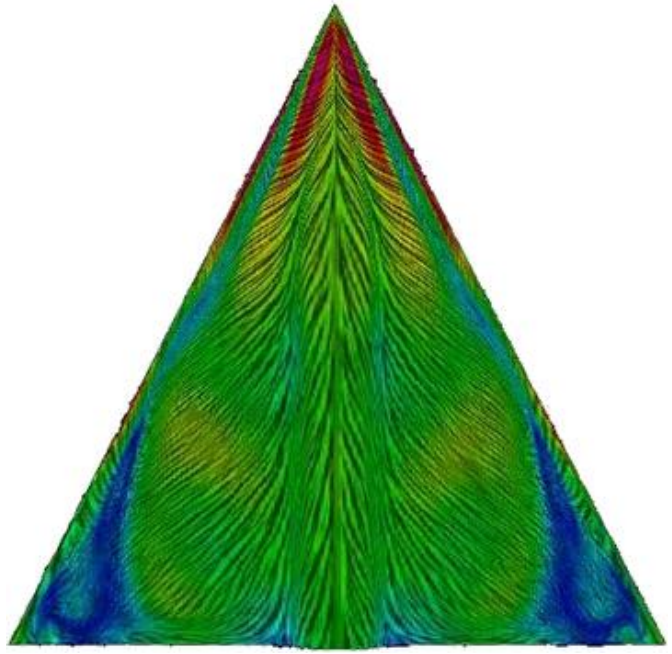


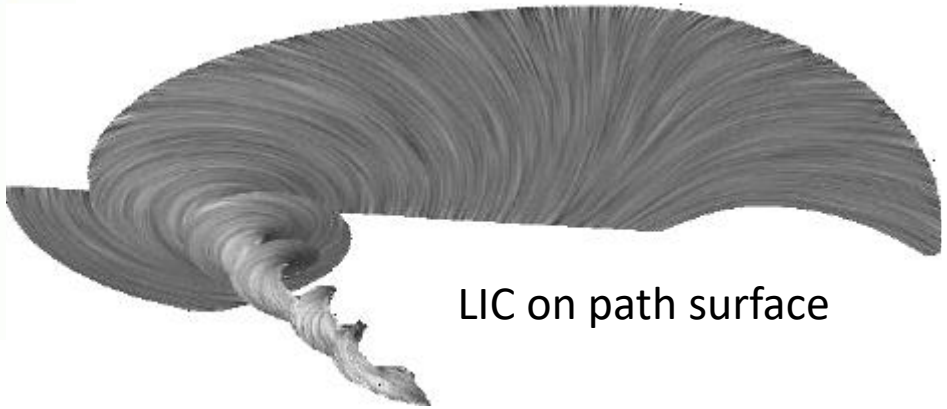
[Cabral & Leedom 93]

Line Integral Convolution

- LIC on surfaces

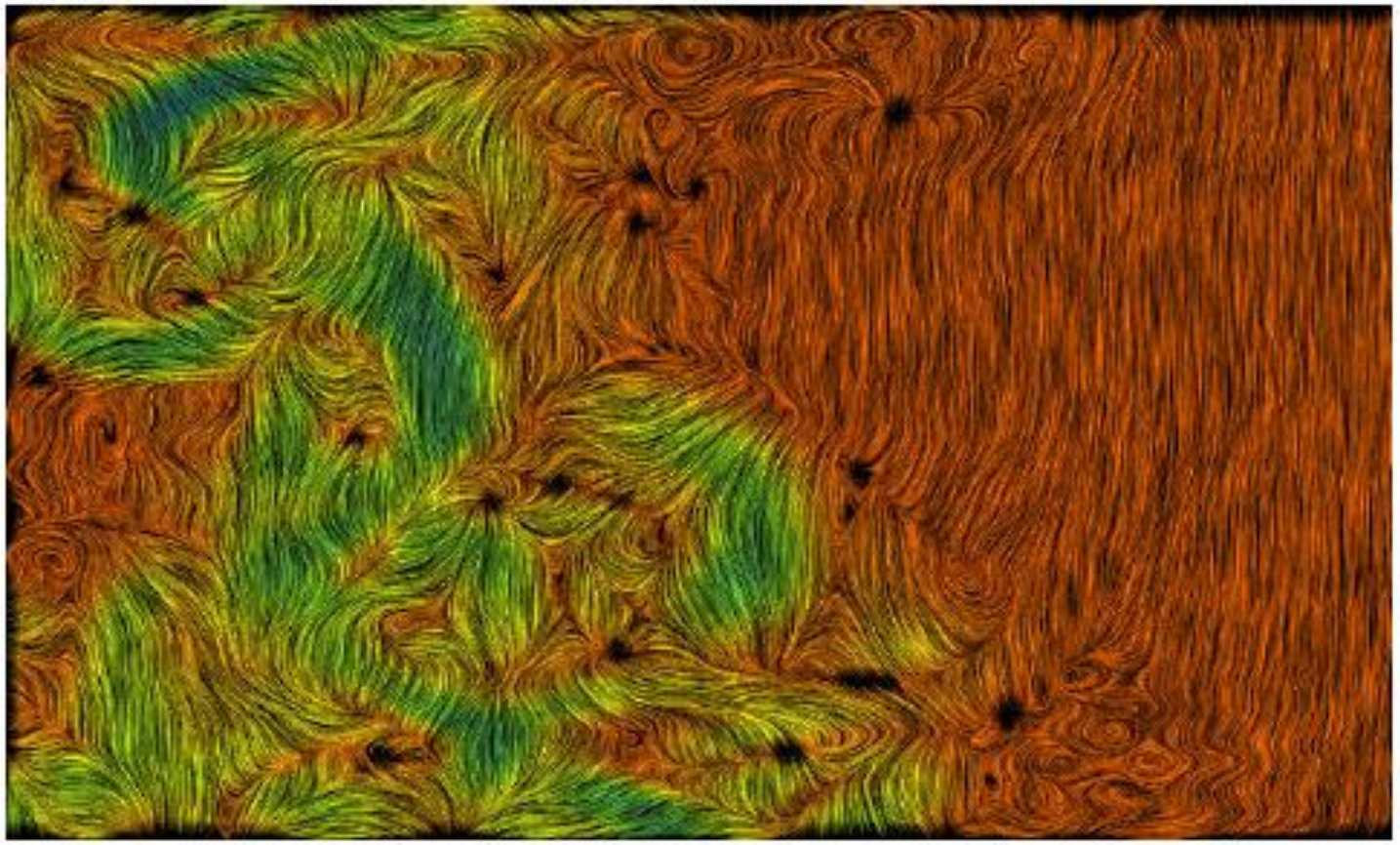






LIC on path surface

Line Integral Convolution

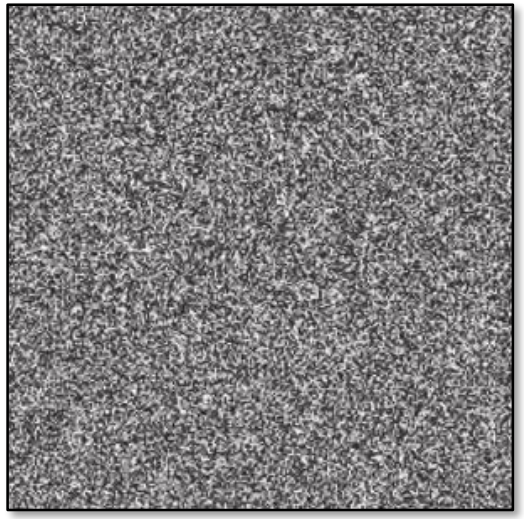


LIC and color coding of velocity magnitude

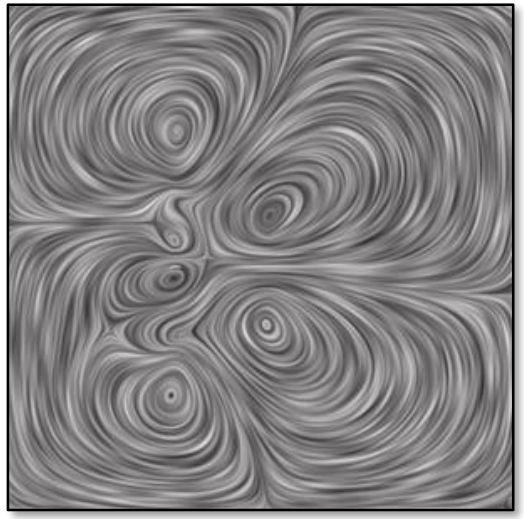
[Cabral & Leedom 93]

Line Integral Convolution

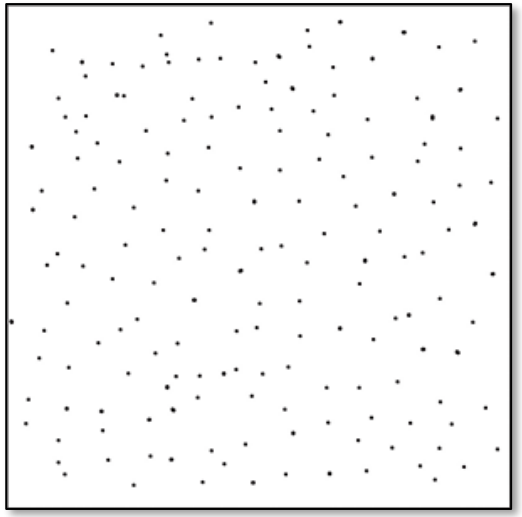
- Impact of input noise



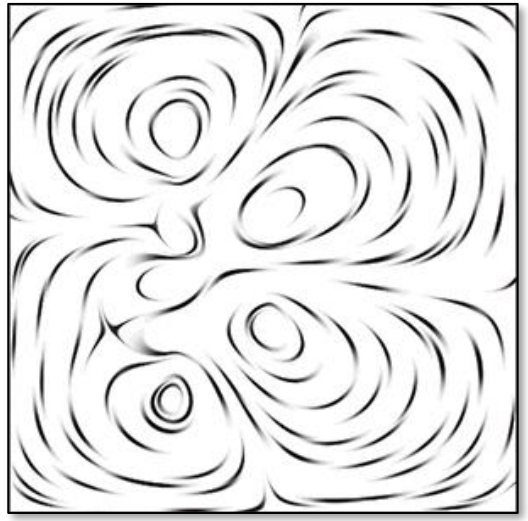
White noise texture



LIC result for white noise



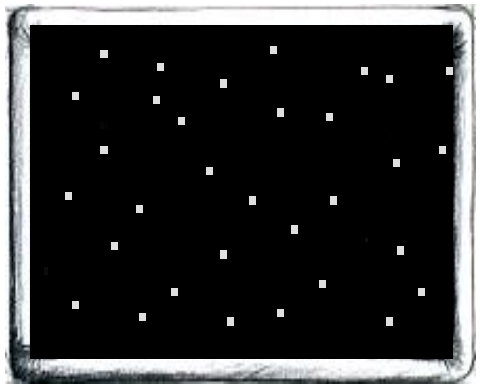
Sparse noise texture

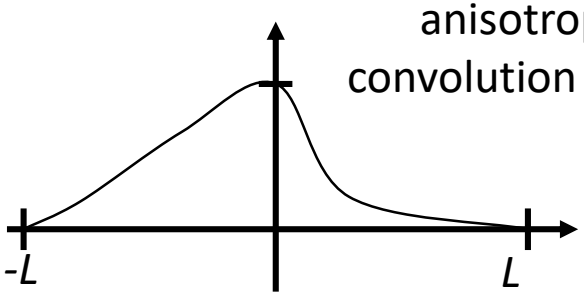


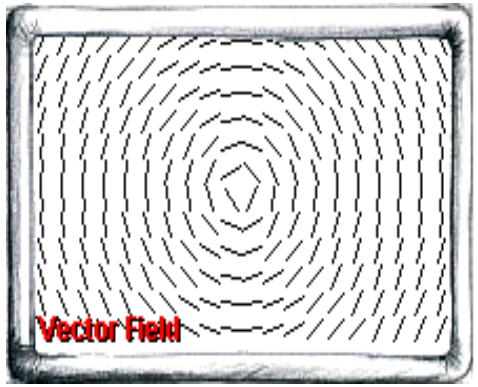
LIC result for sparse noise

Line Integral Convolution

- Oriented LIC (OLIC)
  - Visualizes orientation (in addition to direction)
  - Uses a sparse texture; i.e. smearing of individual drops
  - Asymmetric convolution kernel

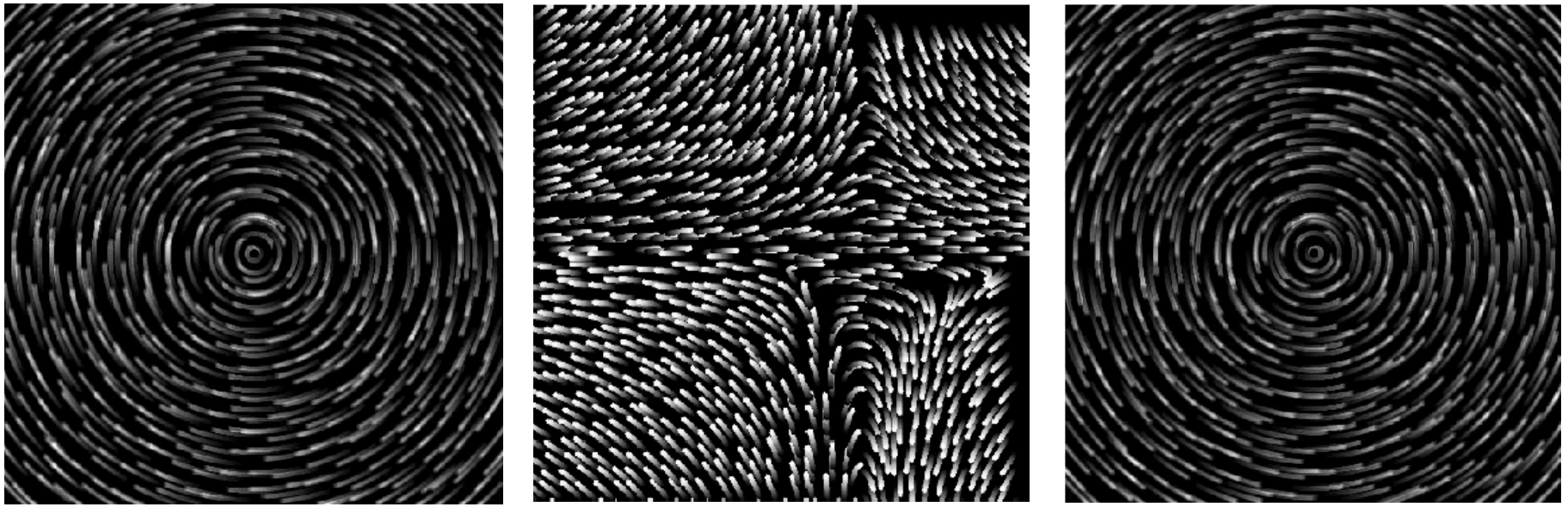






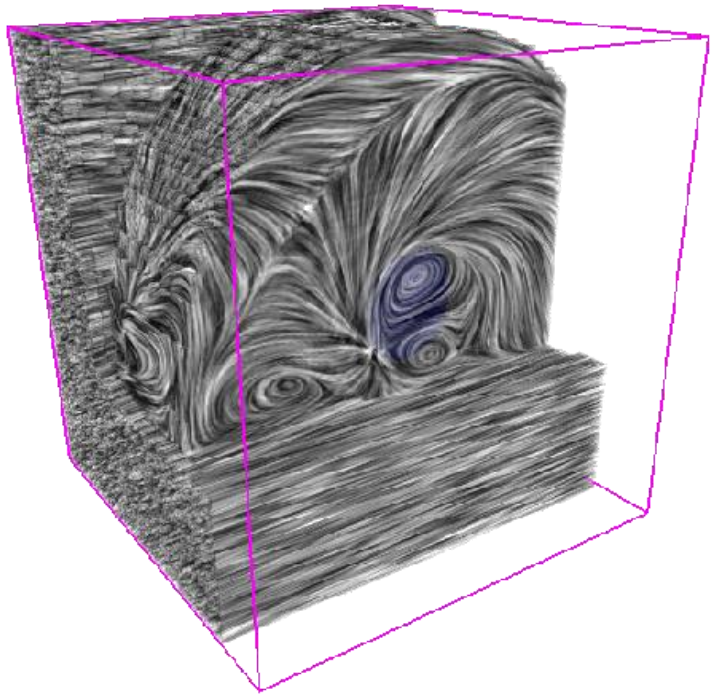
Line Integral Convolution

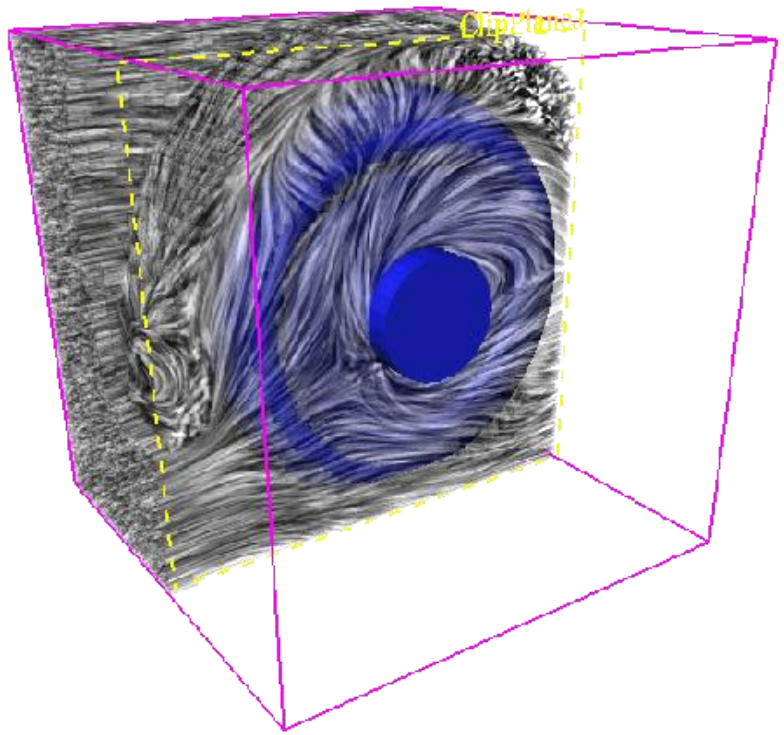
- Oriented LIC (OLIC)

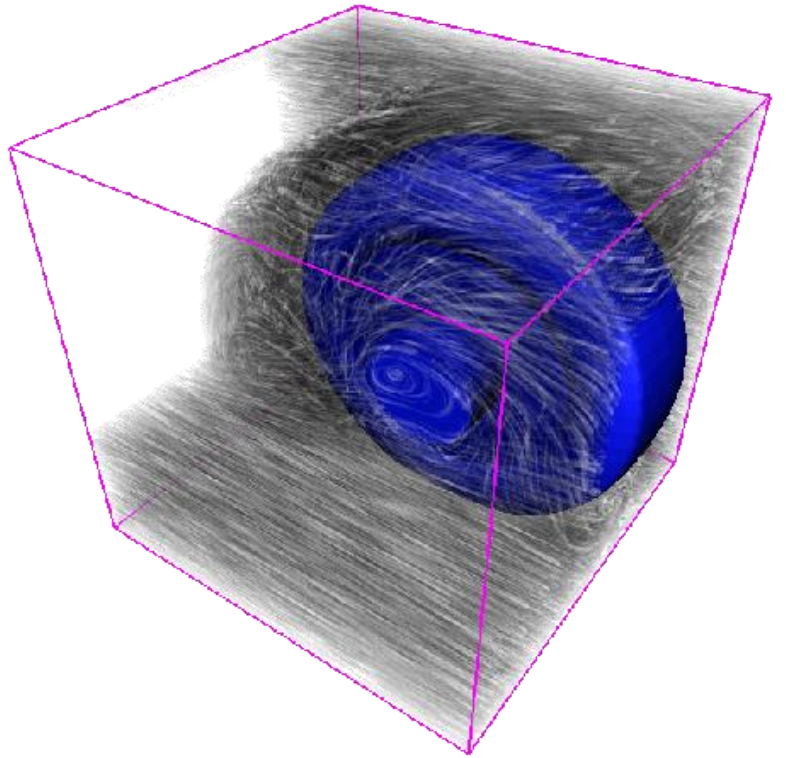


Line Integral Convolution

- 3D LIC
  - Only good if non-relevant data is discarded



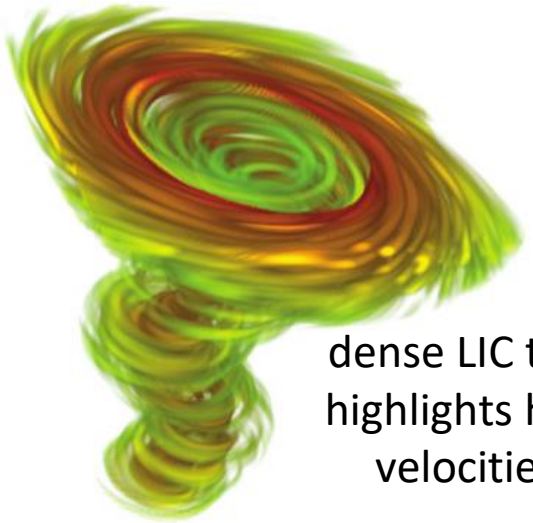




Flow around a wheel
[Rezk-Salama et al. 99]

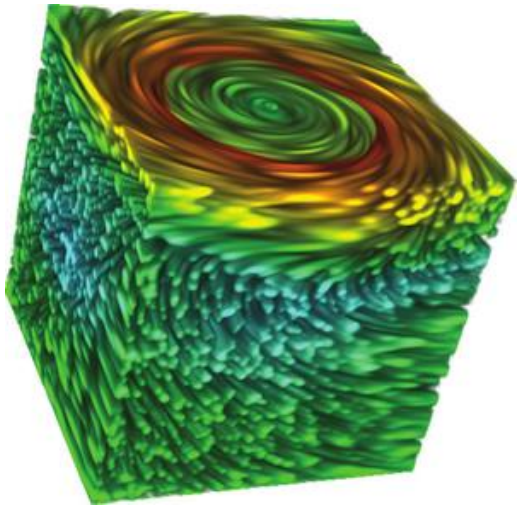
Line Integral Convolution

- 3D LIC

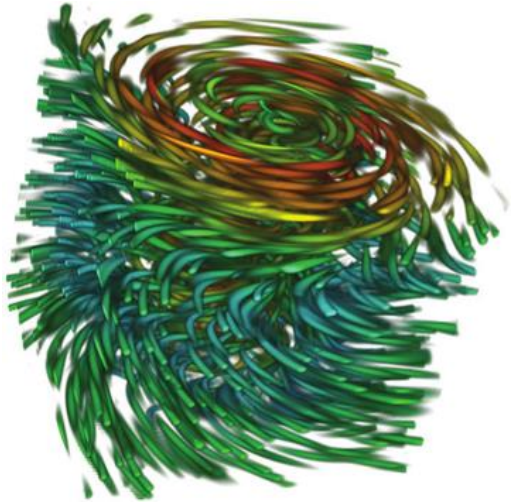


dense LIC that highlights high velocities

sparse LIC that highlights high velocities



dense LIC with white noise input



sparse LIC with spares noise

Line Integral Convolution

- Summary
  - Dense representation of flow fields
  - Convolution along characteristic lines
    → correlation along these lines
  - For 2D and (3D flows)

Questions???