

5. Vector Fields

Contents

- Introduction
- Arrows and Glyphs
- Characteristic lines in vector fields
- Particle tracing
- Line integral convolution

Introduction

- Problem: visualize a function like
 - $F: \Omega \rightarrow \mathbb{R}^2$, with F given only at certain vertices

$$F \leftrightarrow F_{ij} = \begin{pmatrix} F_{ij}^x \\ F_{ij}^y \end{pmatrix}$$

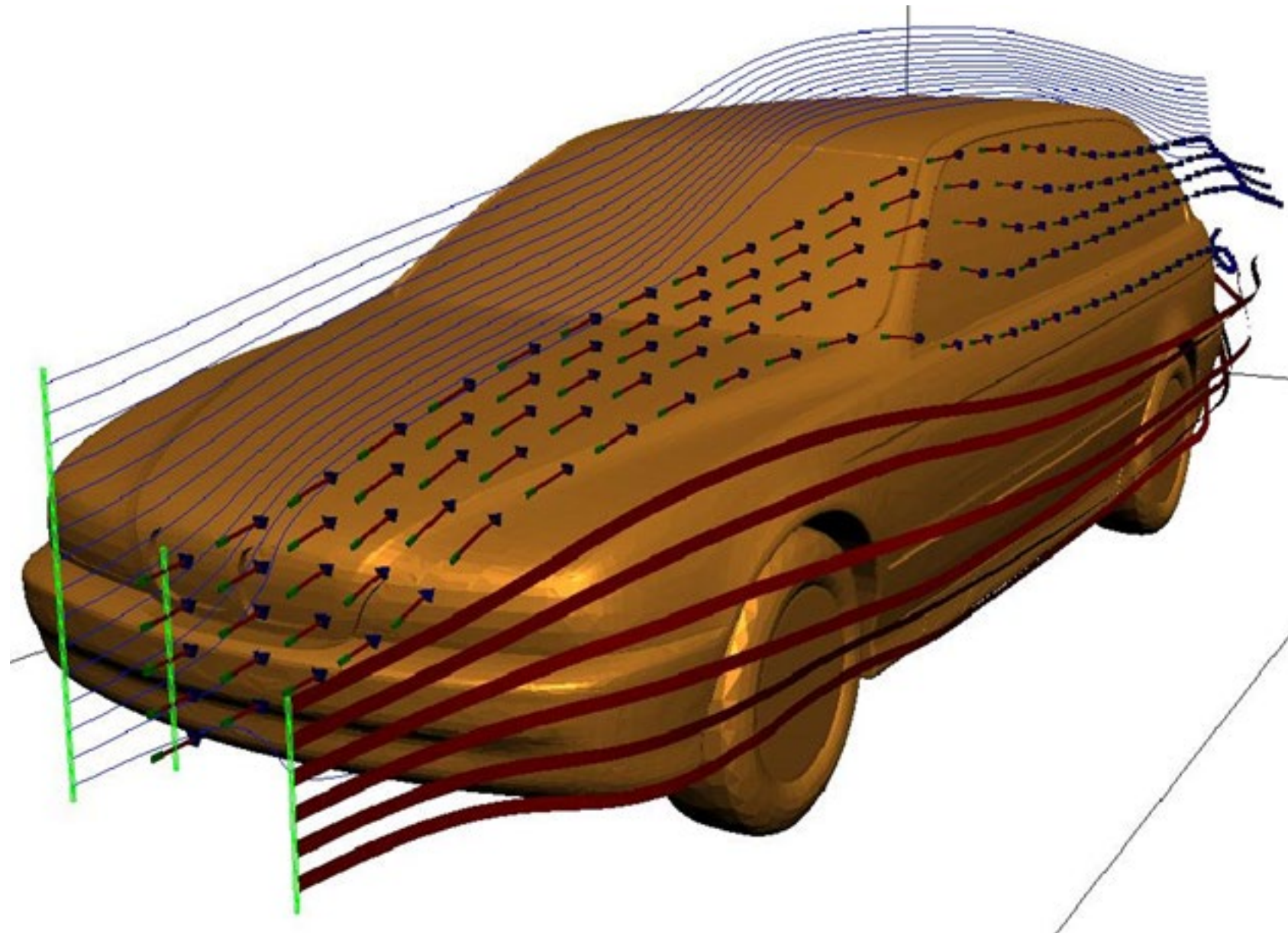
- Ideas
 - Visualize the two scalar fields F_x and F_y
 - But: components are usually not independent \rightarrow no insight!
 - If the data represents some form of velocity
 - Direction of moving particles
 - Visualize the "flow"

Introduction

- Main application area: flow visualization
 - Gas (car industry, airplanes, ...)
 - Fluids (water, reactors, blood vessels, ...)
- Important entities
 - Geometric boundary conditions
 - Velocity (flow) field $v(x, t)$
 - Pressure p , temperature T , density ρ
 - Vorticity: $\nabla \times \vec{v}$
 - Amount of circulation (total angular rate of rotation)
 - Navier-Stokes equations
 - CFD (Computational Fluid Dynamics)

Introduction

Flow visualization based on CFD data



Introduction

Classification of flow visualization

- Dimension (2D or 3D)
- Time-dependency
 - stationary (steady) vs. in-stationary (unsteady)
- Grid type
- Compressible vs. incompressible fluids
- In most cases of flow visualization
 - Numerical methods required

5.1 Arrows and Glyphs


Arrows and Glyphs

Glyphen => "Pfeile in 3D"

- Natural "vector" display
 - Arrows
 - Visualizes local features of a vector field
 - Vector itself
 - Vorticity ($\nabla \times \vec{v}$)
 - External data: pressure, temperature, etc. ...
- Important elements of a vector
 - Direction
 - Magnitude
 - Not: individual components

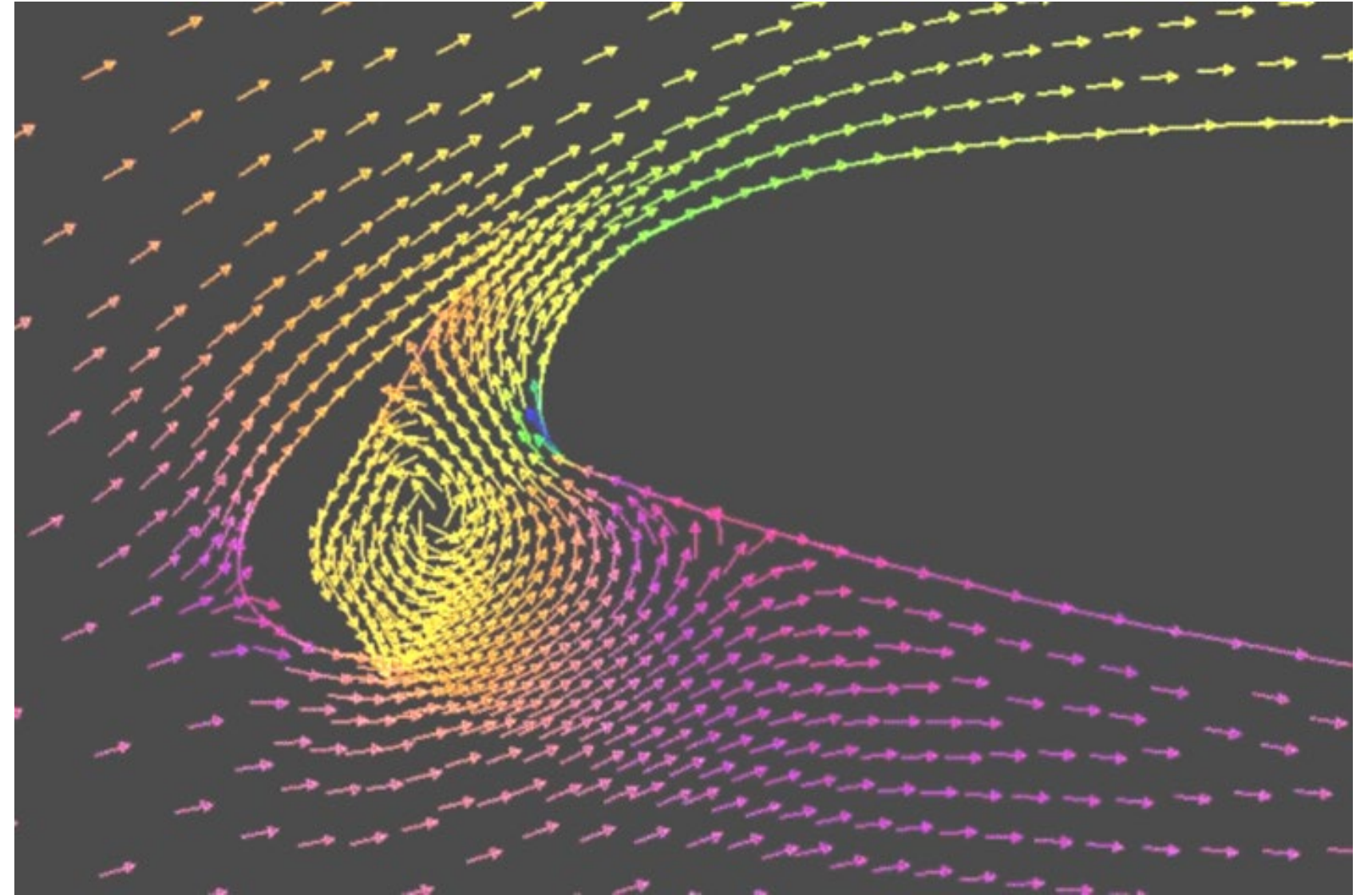
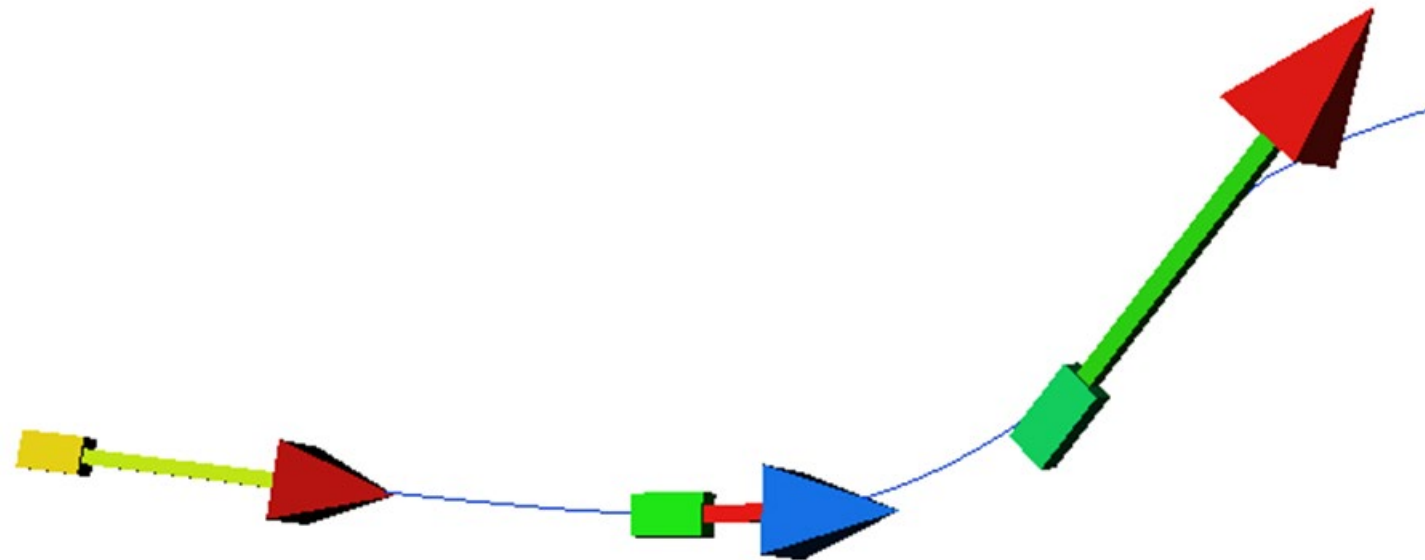
Arrows and Glyphs

Arrows

- Place arrow geometry at each (nth) grid point
 - Geometry: 
- Length
 - Represents magnitude of the vector field
- Direction
 - Coincides with direction of vector field at that vertex
- In case of very fast changing velocity
 - Results are typically not good
 - Better use arrows of constant length
 - Indicate magnitude separately (e.g. color coding)

Arrows and Glyphs

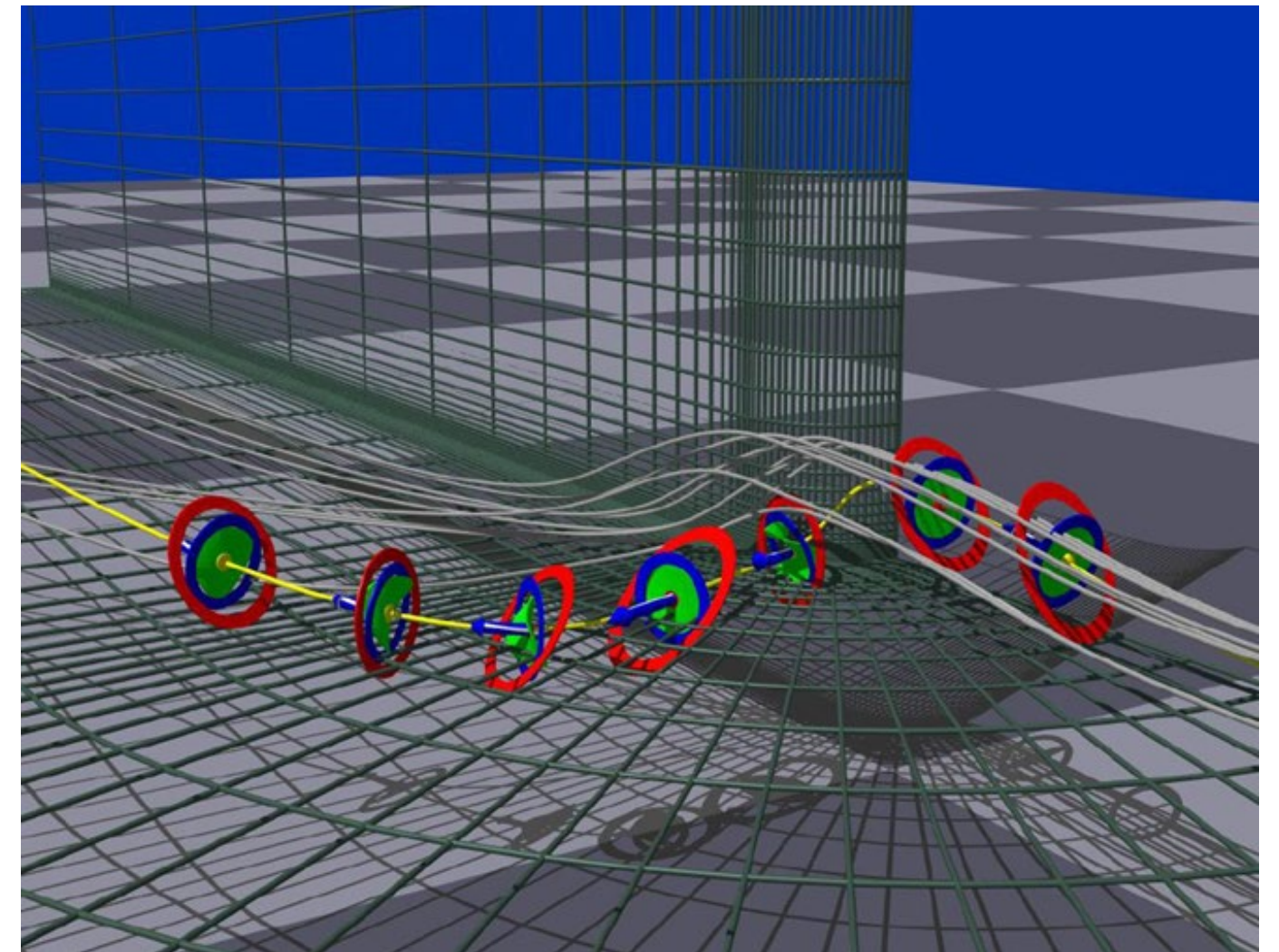
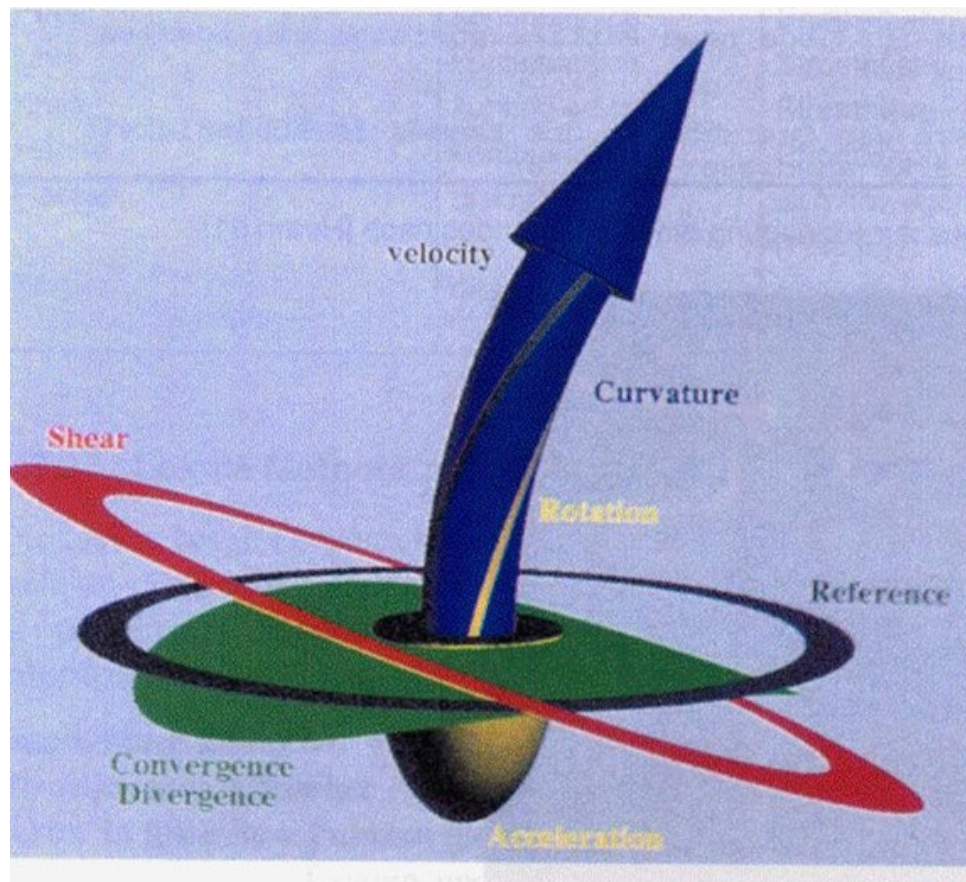
Example with arrows



Arrows and Glyphs

Glyphs

- More complex information, usually used in 3D
- Often more difficult interpretation



Arrows and Glyphs

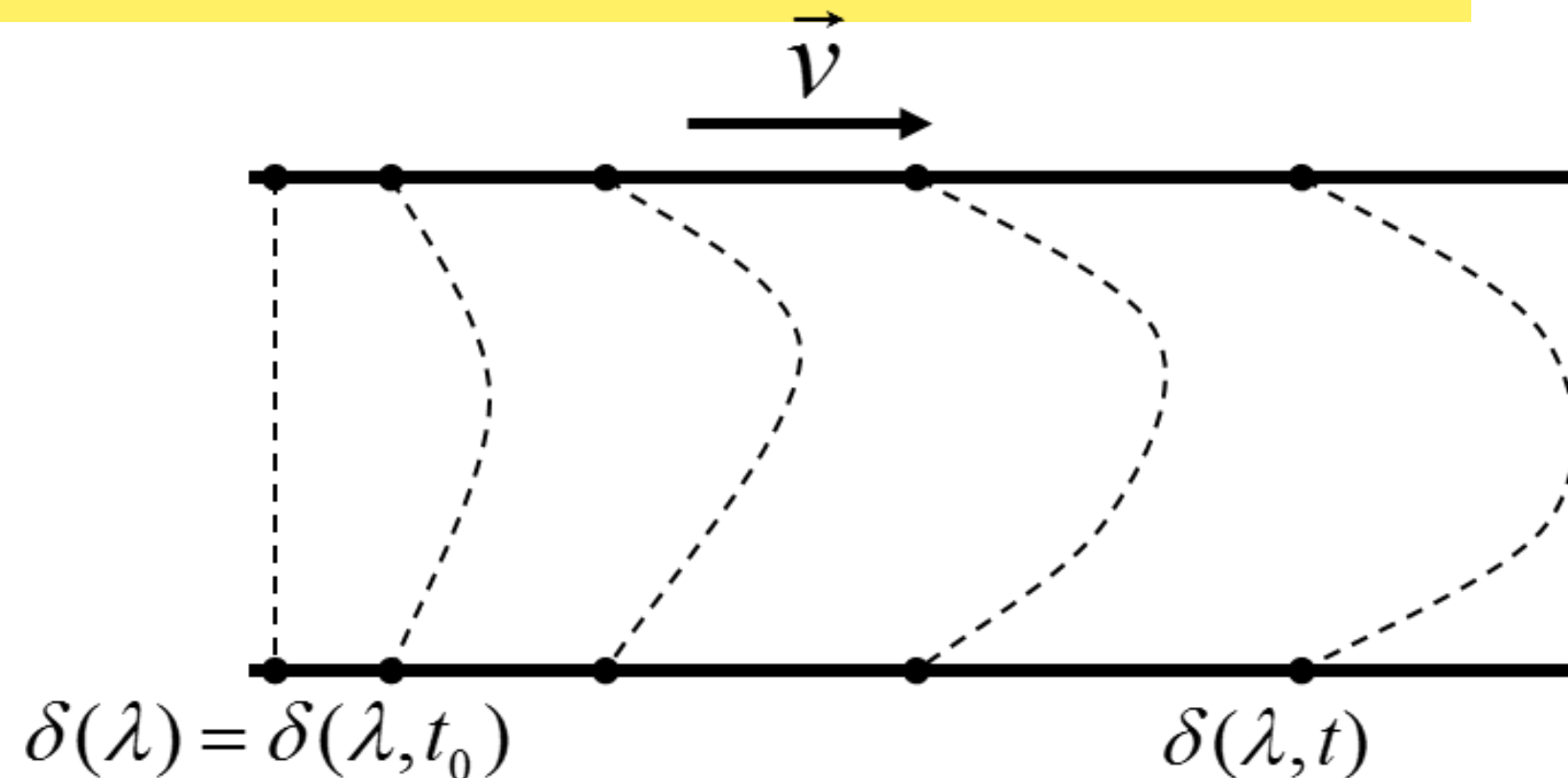
- Advantages
 - Simple
 - Absolute value directly visible
 - 2D fields without complex small structures
- Disadvantages
 - Inherent occlusion effects
 - Very small and overlapping arrows
 - Difficult for 3D and time-dependent fields
 - Poor results if magnitude of velocity changes rapidly
 - Use arrows of constant length and color code magnitude

5.2 Characteristic Lines in Vector Fields

Characteristic Lines

Time lines / time surfaces

- Line/surface of massless elements moving with flow
- Points of identical time
- Connect particles that were released simultaneously

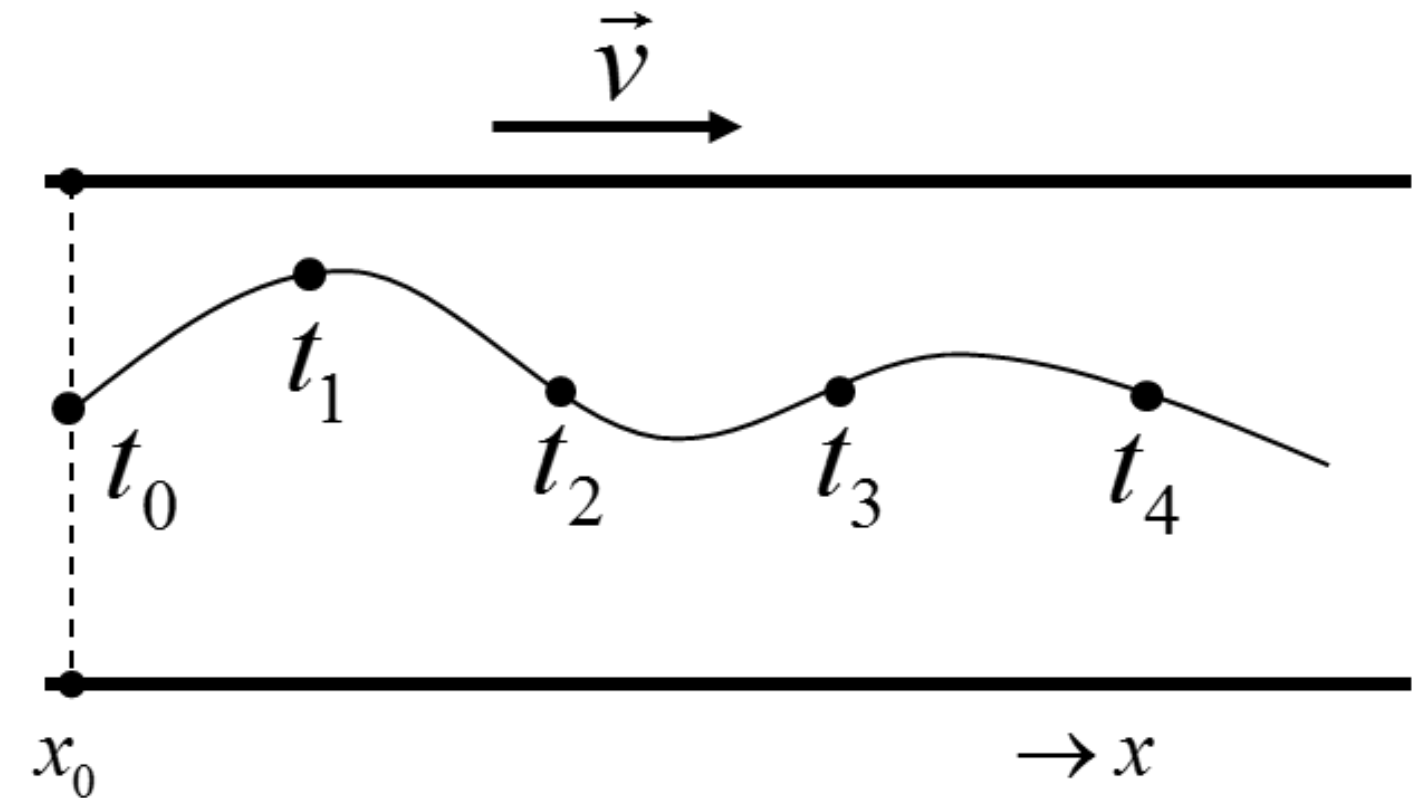


Linienbewegung durch Vektorfeld

Characteristic Lines

Path lines

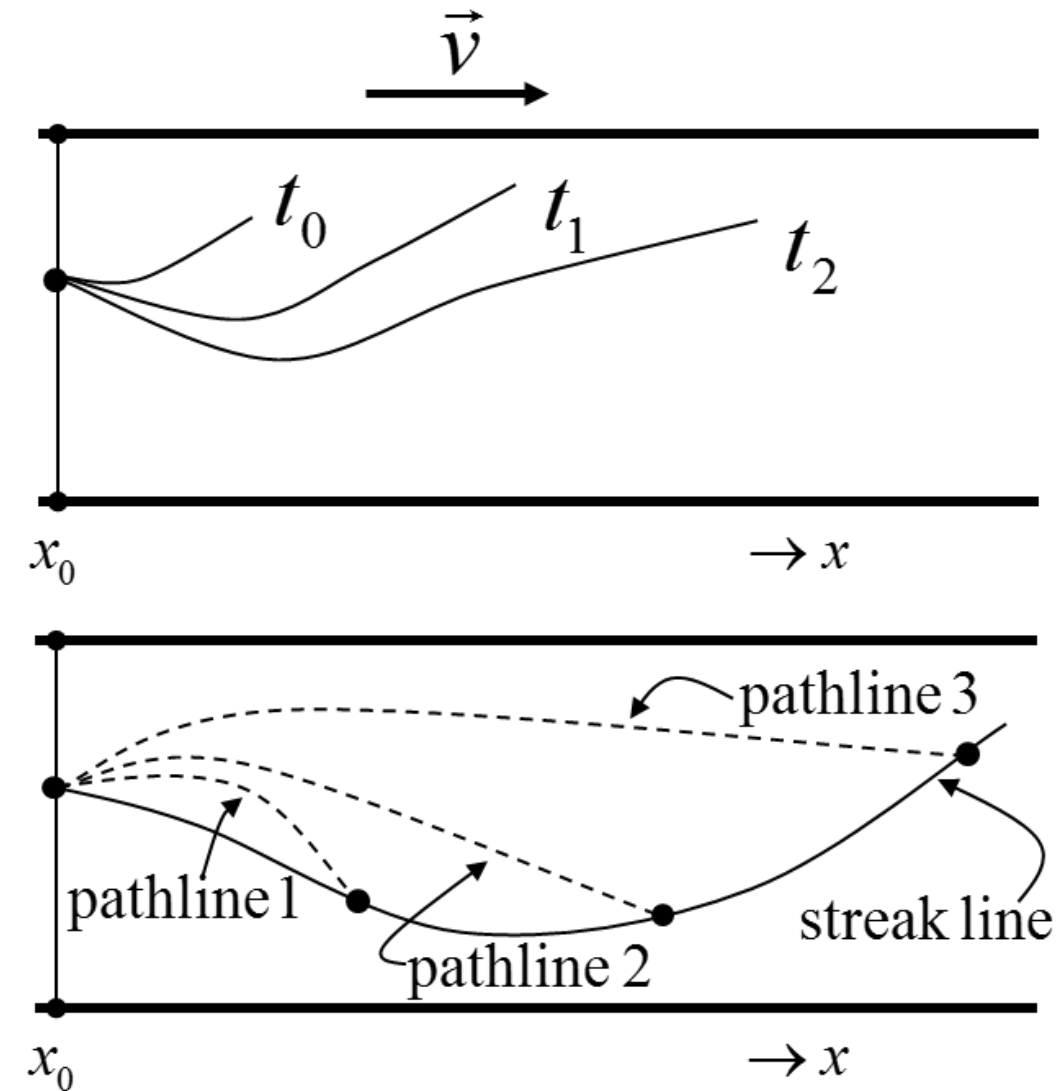
- Path of a single, massless particle
 - Inject particle at point $x(0) = x_0$
 - Long time exposure photo
- Solve $\frac{d\mathbf{x}(t)}{dt} = \mathbf{v}(\mathbf{x}(t), t)$ with $\mathbf{x}(0) = \mathbf{x}_0$



Characteristic Lines

Streak lines

- Trace of dye continuously released at x_0
- Measurement
 - No particle path, but velocity direction (on line described by dye)
- Representation
 - Connecting line of all particles starting within $0 < t_i < t_0$ at x_0
- Calculation
 - Path lines for $(x_0, x_0 + t)$

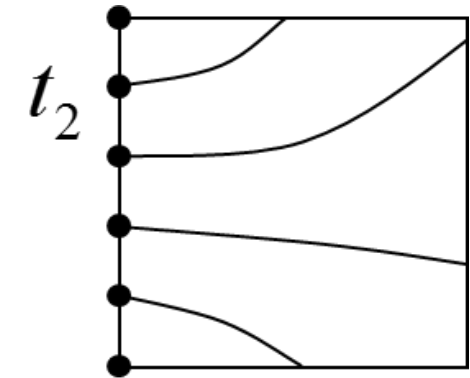
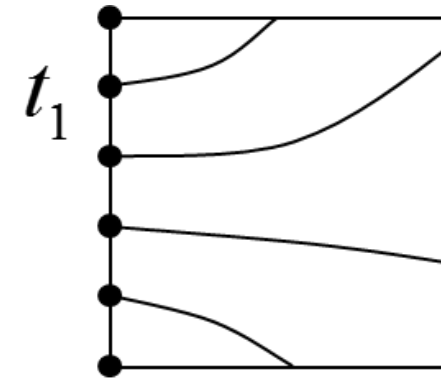
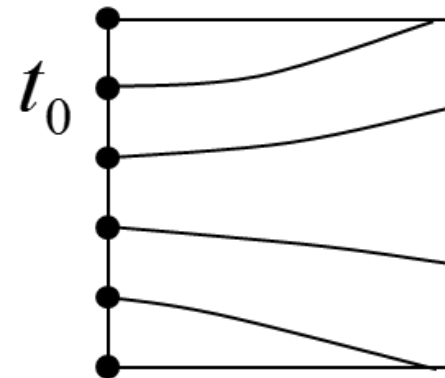


Characteristic Lines

Stream lines (or streamlines)

- Tangential to the vector field - path lines for each t_i
 - Vector field at arbitrary, yet fixed time t

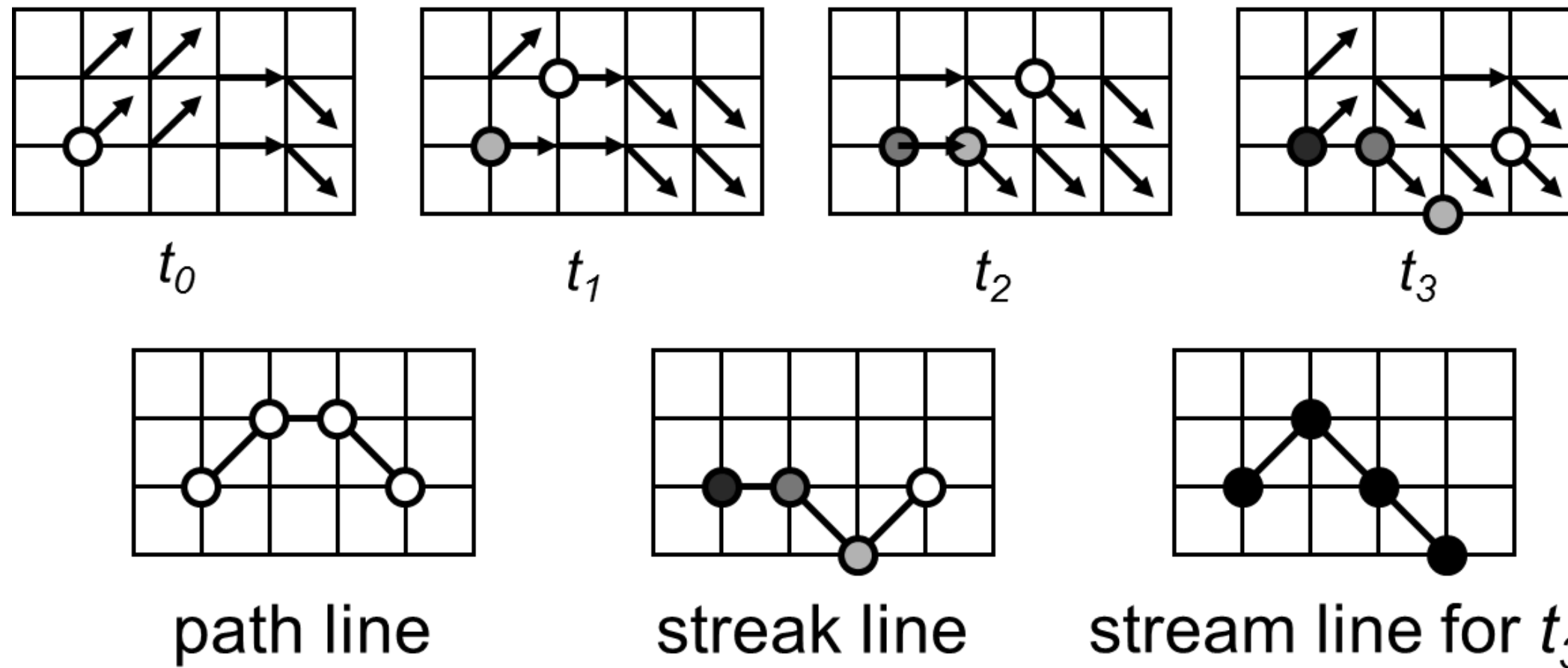
- Solve $\frac{d\mathbf{x}(s)}{ds} = \mathbf{v}(\mathbf{x}(s), t)$ with $\mathbf{x}(0) = \mathbf{x}_0$



- Photo with short time exposure of vector field \mathbf{v} at t_i
 - Freeze the velocity field and compute path lines
 - Information about whole vector field

Characteristic Lines

Comparison

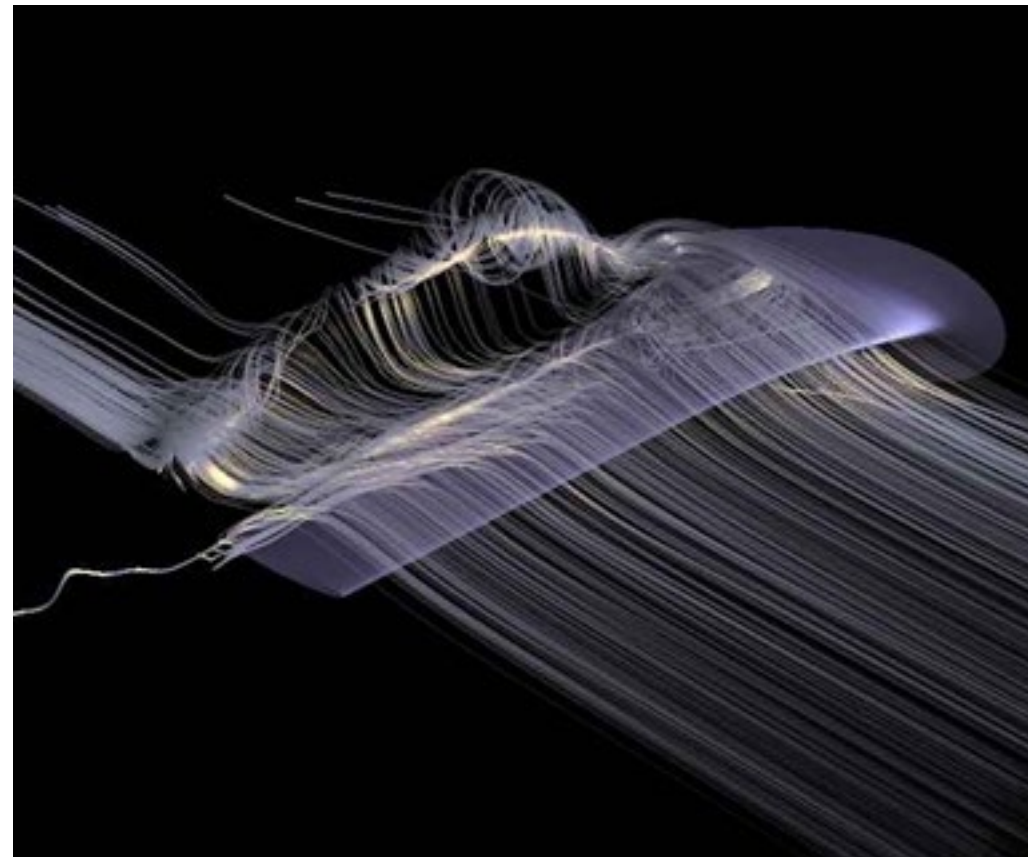


For steady flow, path lines, streak lines
and stream lines are identical!

5.3 Particle Tracing

Particle Tracing

- Vector fields usually combined with "transport"
 - Basic idea: trace particles within the field
 - Characteristic lines
- Mapping approach
 - Lines, surfaces
 - Sometimes animated



Particle Tracing

- Given

- Velocity field of particles / fluid $\vec{v}(\vec{r}, t)$, $\vec{r} = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix}$

- Task

- Find the path $\vec{r}(x(t), y(t))$ of one (or many) massless particles in this field
- Starting point requirement $\vec{r}(t_0) = \vec{r}_0$

- Solve $\frac{d\vec{r}}{dt} = \vec{v}(x(t), y(t)) = \vec{v}(\vec{r}, t) \xrightarrow{\text{Integration}} \text{thereby } \vec{r}(t_1), \vec{r}(t_2), \dots$
 $\vec{r}(t + \Delta t) = \vec{r}(t) + \int_t^{t+\Delta t} \vec{v}(\vec{r}(t), t) dt$

Particle Tracing

Integration

- This requires solving ordinary differential equations (ODE)
 - Usually of first order initial value problems
- Various methods
 - Single step: Euler
 - Multi step: Adams Bashforth
 - Multi stage: Runge-Kutta, Heun
- Literature
 - Schwarz, "Numerische Mathematik" (Kapitel 9)
 - Deuflhard, Bornemann, "Numerische Mathematik II"
 - Brauer, "Numerische Behandlung gewöhnlicher Differentialgleichungen"
 - and many more ...

Particle Tracing

Ordinary differential equation

- Initial value problem $\frac{d\vec{r}}{dt} = \vec{v}(\vec{r}, t)$, where $\vec{r}(t_0) = \vec{r}_0$
- Numerical solution
 - Discretize the "time variable": $\{t_1, t_2, \dots\}$
 - Determine r_j which approximates the solution $r(t_j)$
 - Usually, we use constant step size: $t_{j+1} = t_j + \Delta t$

konstant

Particle Tracing

Classification of methods

	neue Position aus alter Single step	mehrere Positionen werden betrachtet Multi step
Explicit	$\vec{\mathbf{r}}_{i+1} = \phi_i(\Delta t, t_i, \vec{\mathbf{r}}_i)$	$\vec{\mathbf{r}}_{i+1} = \phi_i(\Delta t, t_i, \vec{\mathbf{r}}_i, \vec{\mathbf{r}}_{i-1}, \vec{\mathbf{r}}_{i-2}, \dots)$
Implicit	$\psi_i(\Delta t, t_i, \vec{\mathbf{r}}_i, \vec{\mathbf{r}}_{i+1}) = 0$	$\psi_i(\Delta t, t_i, \vec{\mathbf{r}}_{i+1}, \vec{\mathbf{r}}_i, \vec{\mathbf{r}}_{i-1}, \vec{\mathbf{r}}_{i-2}, \dots) = 0$

Particle Tracing

- Single step methods (e.g. Euler)
 - Position r_{i+1} depends only on r_i
 - Good for adaptive strategy since no recalculation is required
- Multi step methods (e.g. Adams Bashforth)
 - Uses results of a fixed number of previous time steps
 - Cost intensive for adaptivity
- Multi stage (e.g. Heun, Runge-Kutta)
 - Single step approach with multiple evaluations of the function per time step
 - Due to cost effectiveness applied for refinement

Particle Tracing

Euler method (Runge-Kutta 1st order - RK1)

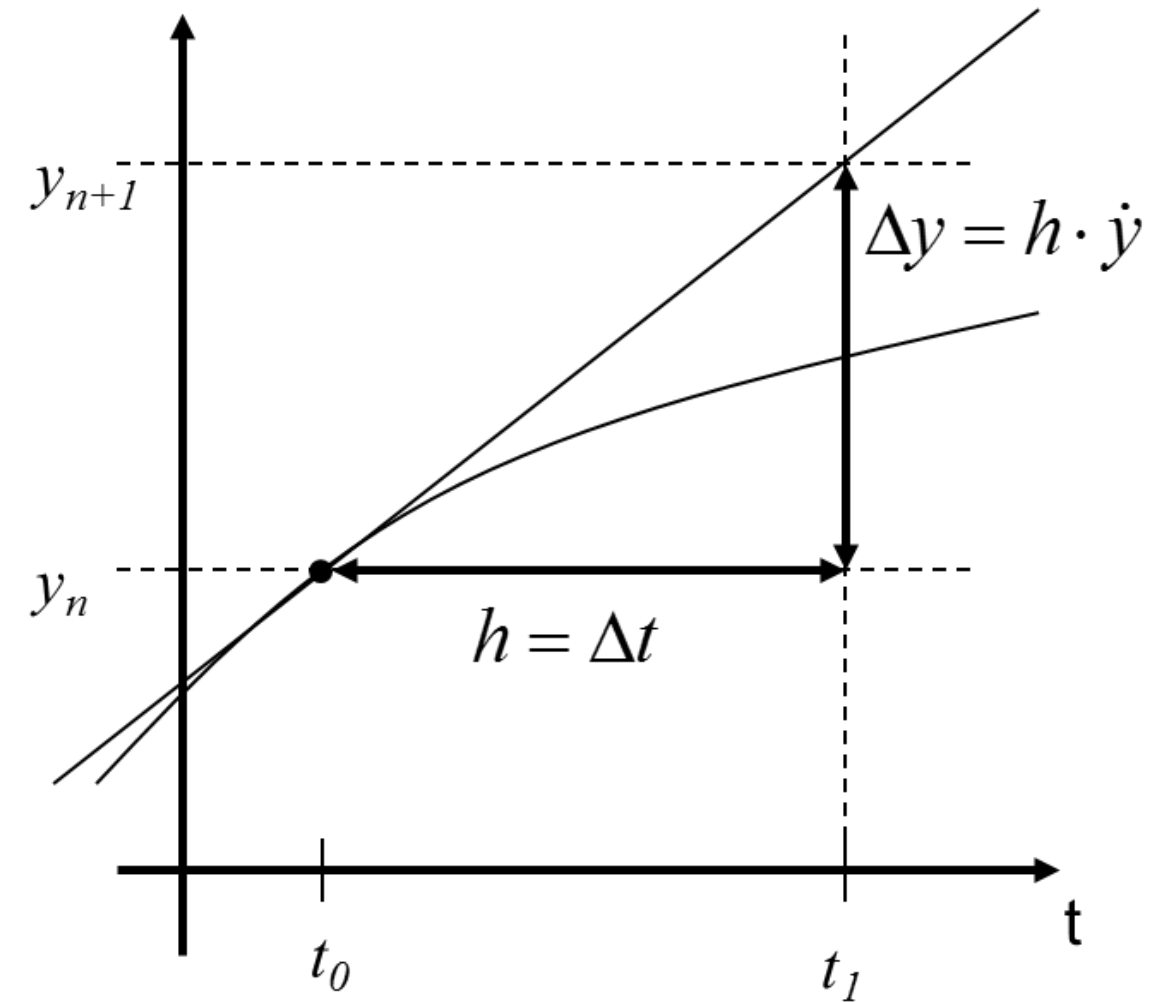
- Explicit $\vec{\mathbf{r}}_{j+1} = \vec{\mathbf{r}}_j + \Delta t \vec{\mathbf{v}}(\vec{\mathbf{r}}_j, t_j)$ $t_{j+1} = t_j + \Delta t$
- Implicit $\vec{\mathbf{r}}_{j+1} = \vec{\mathbf{r}}_j + \Delta t \vec{\mathbf{v}}(t_j + \Delta t, \vec{\mathbf{r}}_{j+1})$
- Local discretization error $O(\Delta t^2)$

Particle Tracing

Explanation in 1D

- Explicit $y_{n+1} \cong y_n + h f(y(n), t)$
 $\cong y_n + h \dot{y}(y(n), t)$
 $\cong y_n + h \dot{y}_n$

- Implicit $y_{n+1} \cong y_n + h \dot{y}_{n+1}$



Particle Tracing

Example with Euler

- Curl field $\vec{v}(t, x, y) = \vec{v}(x, y) = \begin{pmatrix} y \\ -x \end{pmatrix}$

- Exact solution is $\begin{pmatrix} y \\ x \end{pmatrix} = C \begin{pmatrix} \sin t \\ \cos t \end{pmatrix}$

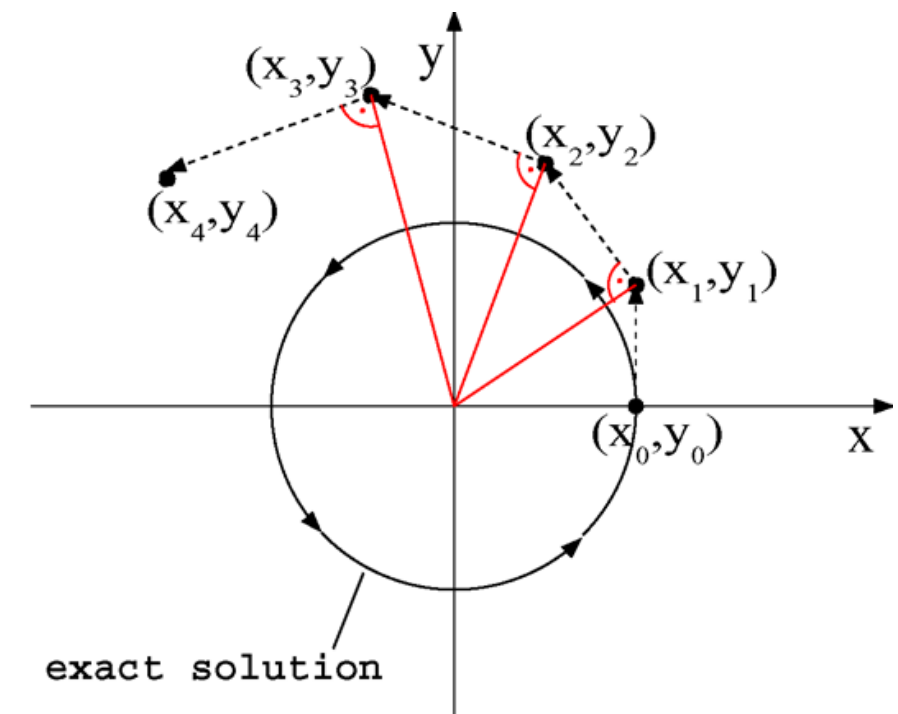
- This is a circle centered at the origin (radius depends on the starting point)

Particle Tracing

Explicit Euler

- Obtain velocity at current (x_i, y_i) with $v(x_i, y_i)$
 - Consider tangent at "starting point" (x_i, y_i)
- Assuming velocity is constant over the next time step leads to →
 - Form of a spiral due to discretization error!

$$(x_{i+1}, y_{i+1}) = (x_i, y_i) + \Delta t \vec{v}(x_i, y_i)$$

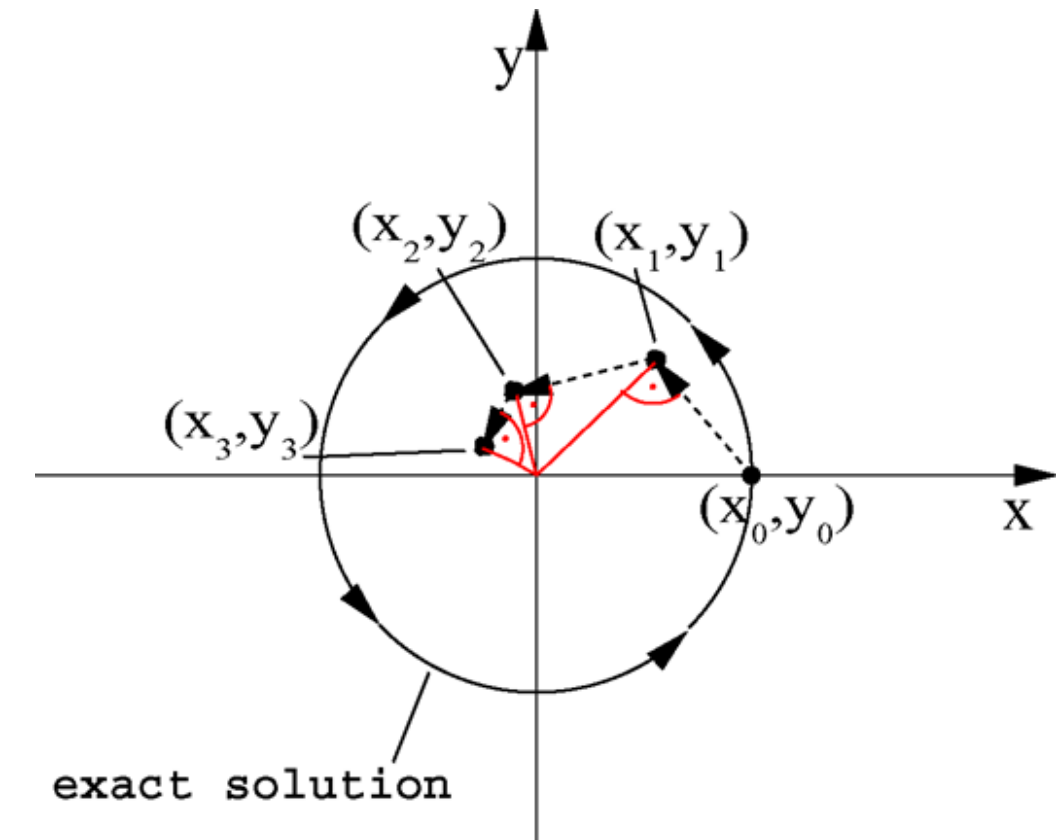


Particle Tracing

Implicit Euler

- Search (x_{i+1}, y_{i+1}) with velocity $v(x_{i+1}, y_{i+1})$
- Consider tangent at "end-point" (x_{i+1}, y_{i+1})
- This leads to \rightarrow
- Also form of a spiral
'due to discretization error!'

$$(x_{i+1}, y_{i+1}) = (x_i, y_i) + \Delta t \vec{v}(x_{i+1}, y_{i+1})$$



Particle Tracing

Euler summary

- Explicit
 - Local error $O(\Delta t^2)$
 - Global error $O(\Delta t)$
 - We need $1/\Delta t$ steps
- Implicit
 - Local error $O(\Delta t^2)$, global error $O(\Delta t)$
 - More stable than explicit
 - If v is known analytically, we have to solve for x_{i+1}, y_{i+1}
 - Otherwise: iteration is required
 - Predictor-corrector variant (not covered here)
- Bottom line: avoid explicit Euler methods!

Particle Tracing

Heun method (Runge-Kutta 2nd order - RK2)

- Explicit

$$\vec{\mathbf{r}}_{i+1} = \vec{\mathbf{r}}_i + \frac{\Delta t}{2} (\vec{\mathbf{k}}_1 + \vec{\mathbf{k}}_2) \quad \begin{aligned} \vec{\mathbf{k}}_1 &= \vec{\mathbf{v}}(t_j, y_j) \\ \vec{\mathbf{k}}_2 &= \vec{\mathbf{v}}(t_j + \Delta t, \vec{\mathbf{r}}_j + \Delta t \vec{\mathbf{k}}_1) \end{aligned}$$

- Some kind of "average" between implicit and explicit Euler
- Implicit

$$\vec{\mathbf{r}}_{j+1} = \vec{\mathbf{r}}_j + \frac{\Delta t}{2} (\vec{\mathbf{v}}(t_j, \vec{\mathbf{r}}_j) + \vec{\mathbf{v}}(t_j + \Delta t, \vec{\mathbf{r}}_{j+1}))$$

- Local discretization error $O(\Delta t^3)$

Particle Tracing

Classical Runge-Kutta (RK4)

- Application of Simpson's rule

$$\vec{\mathbf{r}}_{j+1} = \vec{\mathbf{r}}_j + \Delta t / 6 (\vec{\mathbf{k}}_1 + 2\vec{\mathbf{k}}_2 + 2\vec{\mathbf{k}}_3 + \vec{\mathbf{k}}_4)$$

where

$$\vec{\mathbf{k}}_1 = \vec{\mathbf{v}}(t_j, \vec{\mathbf{r}}_j)$$

$$\vec{\mathbf{k}}_2 = \vec{\mathbf{v}}(t_j + \Delta t / 2, \vec{\mathbf{r}}_j + \Delta t / 2 \vec{\mathbf{k}}_1)$$

$$\vec{\mathbf{k}}_3 = \vec{\mathbf{v}}(t_j + \Delta t / 2, \vec{\mathbf{r}}_j + \Delta t / 2 \vec{\mathbf{k}}_2)$$

$$\vec{\mathbf{k}}_4 = \vec{\mathbf{v}}(t_j + \Delta t, \vec{\mathbf{r}}_j + \Delta t \vec{\mathbf{k}}_3)$$

- Local discretization error $O(\Delta t^5)$ relativ kleiner Diskretisierungsfehler

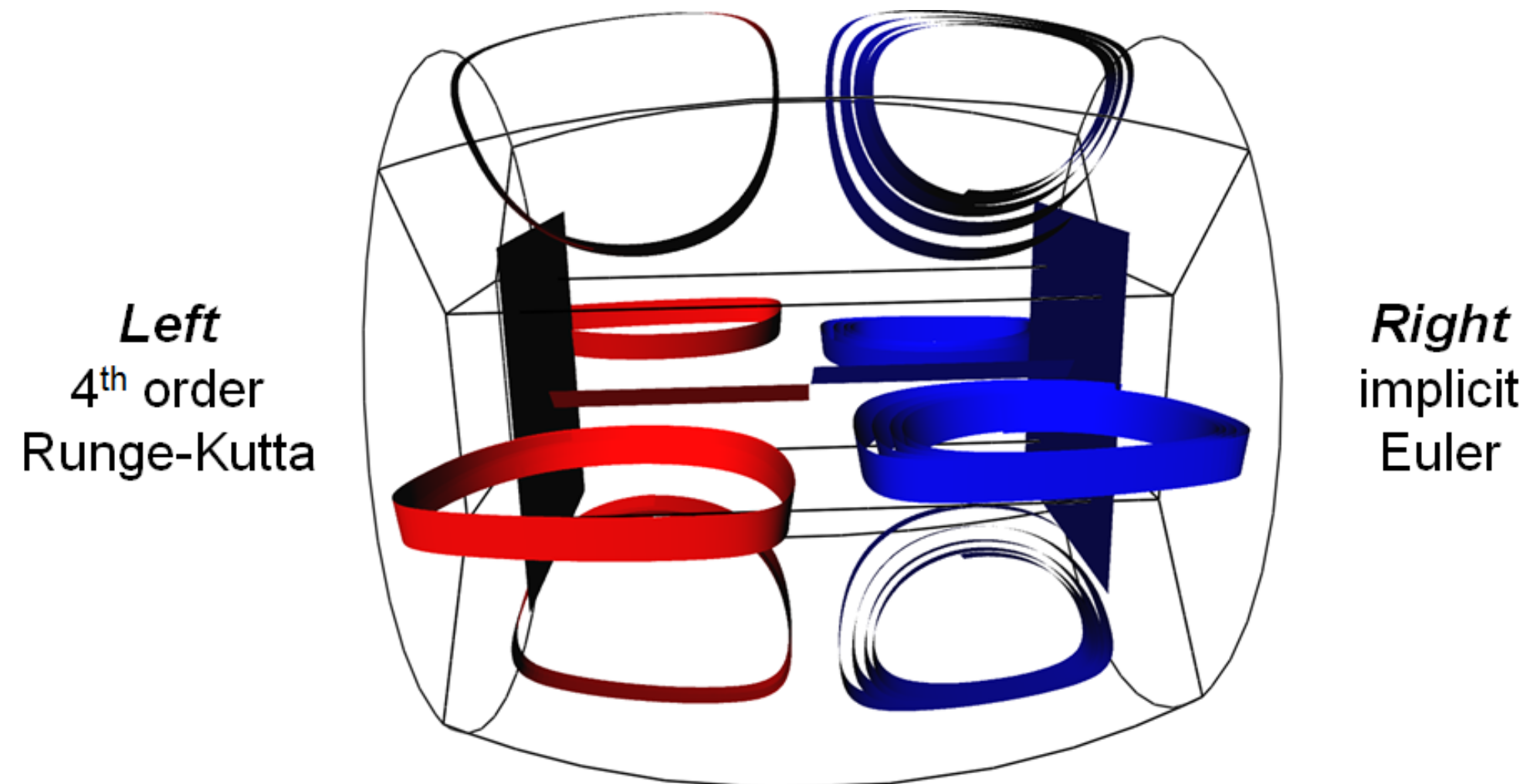
NOTE:

- There is also implicit RK4
- Note that stability is achieved either by multiple function evaluations or with implicit techniques.
- Here: explicit RK4 has multiple function evaluations \Rightarrow ok!

Particle Tracing

The right integration technique can make a difference

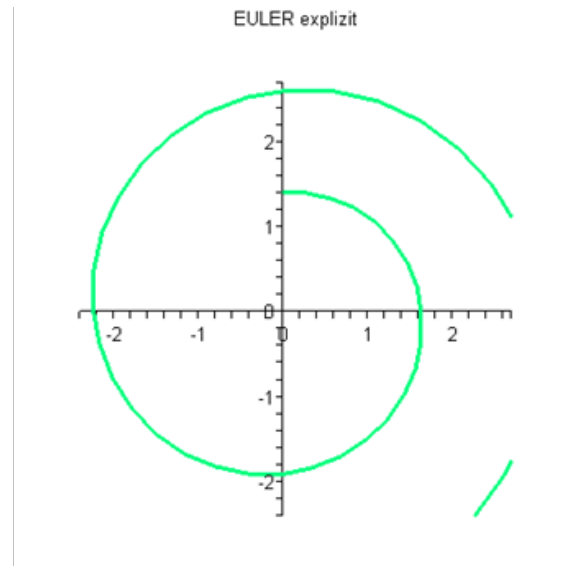
- Example: flow in a floating zone furnace for crystal growing



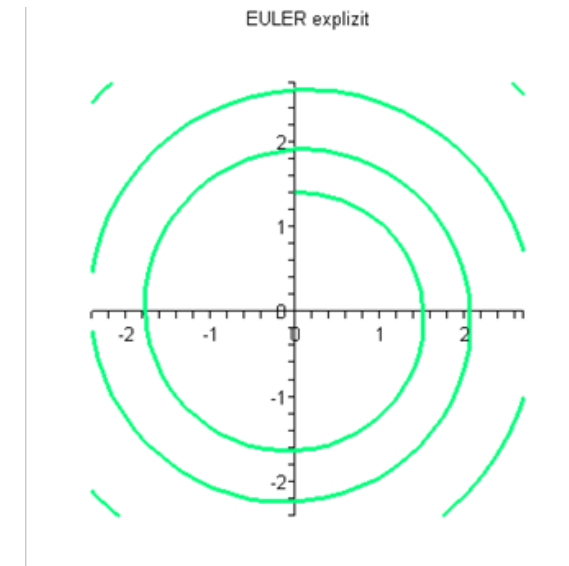
Particle Tracing

Example: circular flow with Euler

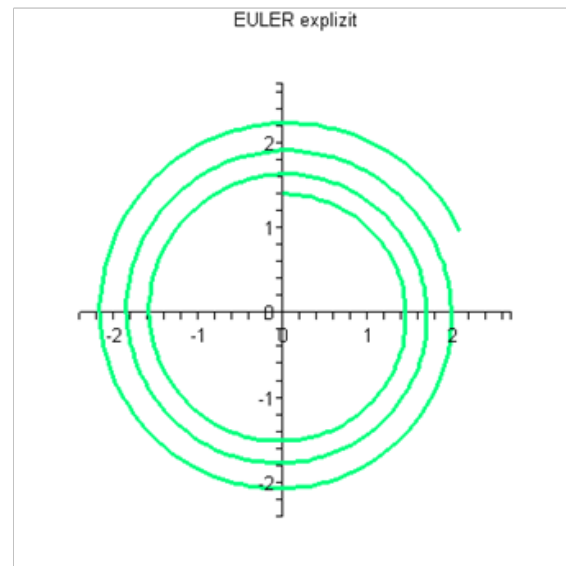
$\Delta t = 0.200$



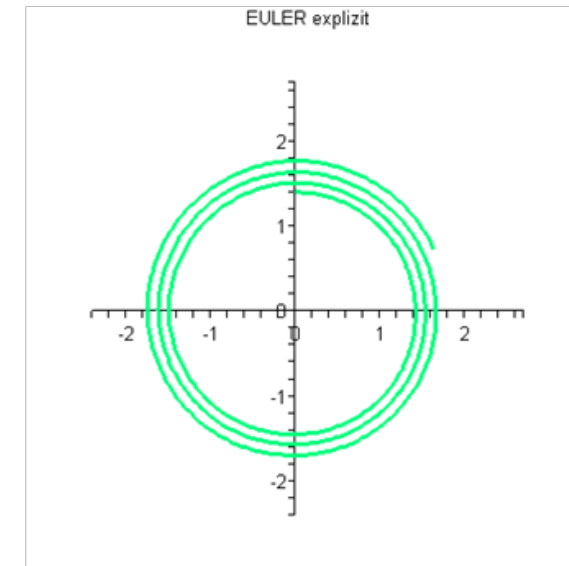
$\Delta t = 0.100$



$\Delta t = 0.050$



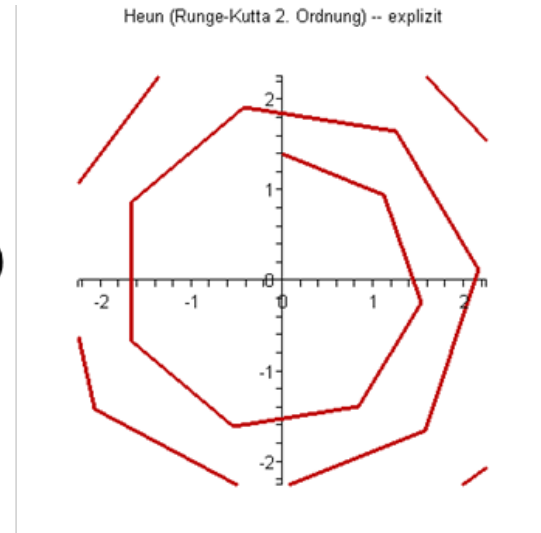
$\Delta t = 0.025$



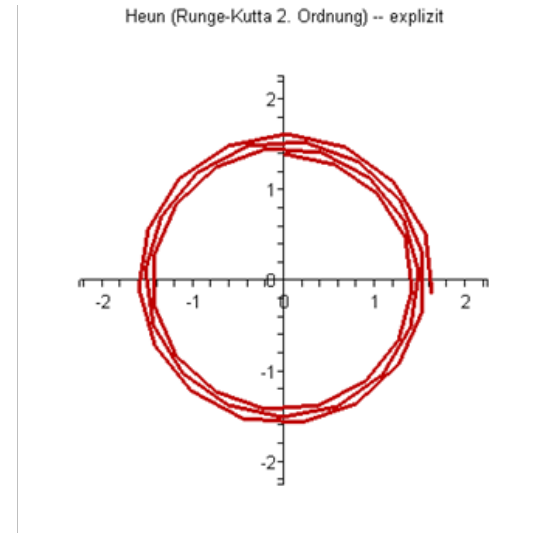
Particle Tracing

Example: circular flow with Heun

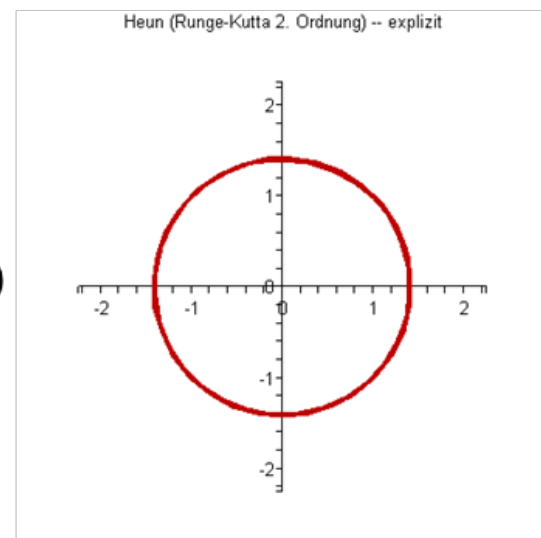
$\Delta t = 0.800$



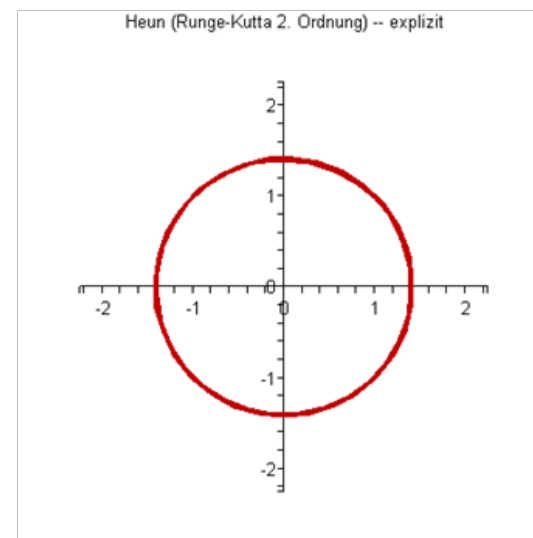
$\Delta t = 0.400$



$\Delta t = 0.200$



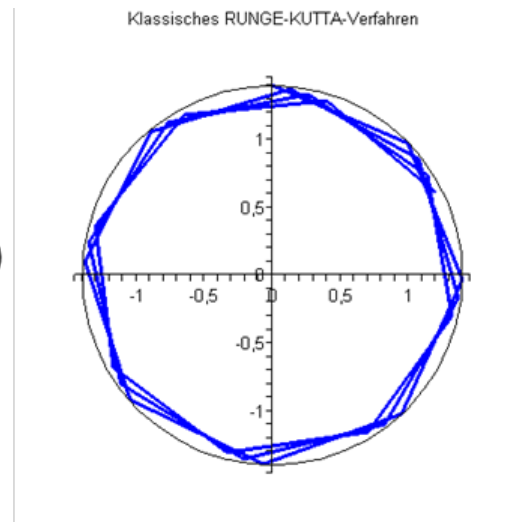
$\Delta t = 0.100$



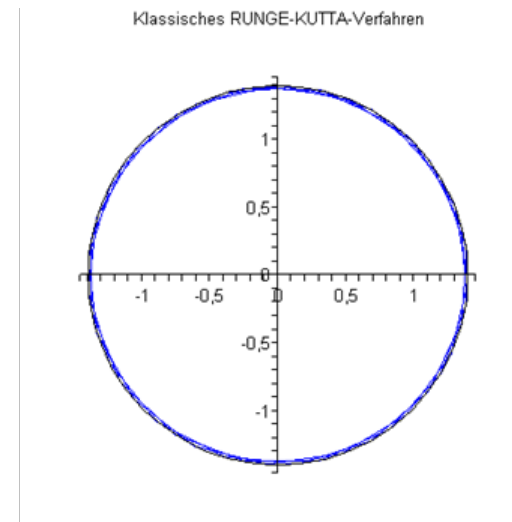
Particle Tracing

Example: circular flow with Runge-Kutta

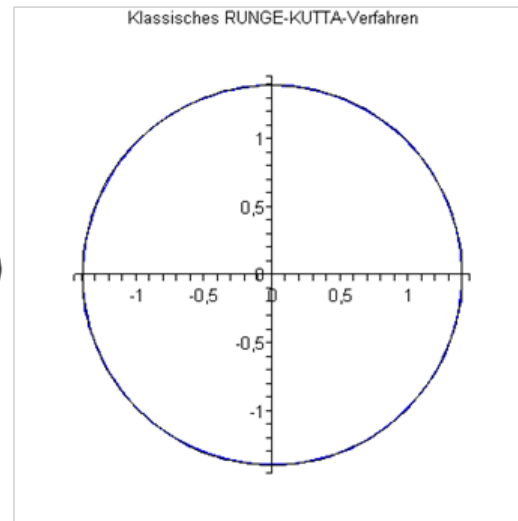
$\Delta t = 0.800$



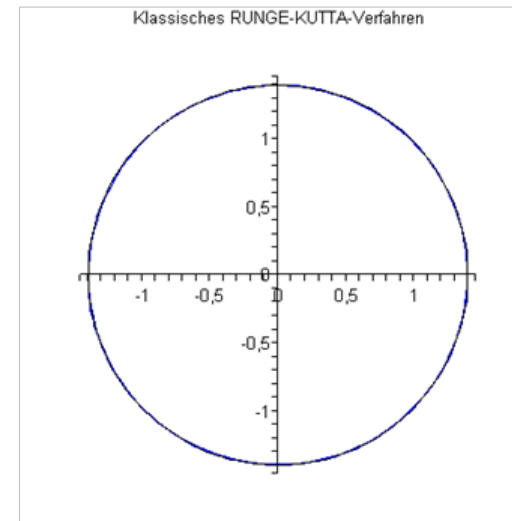
$\Delta t = 0.400$



$\Delta t = 0.200$

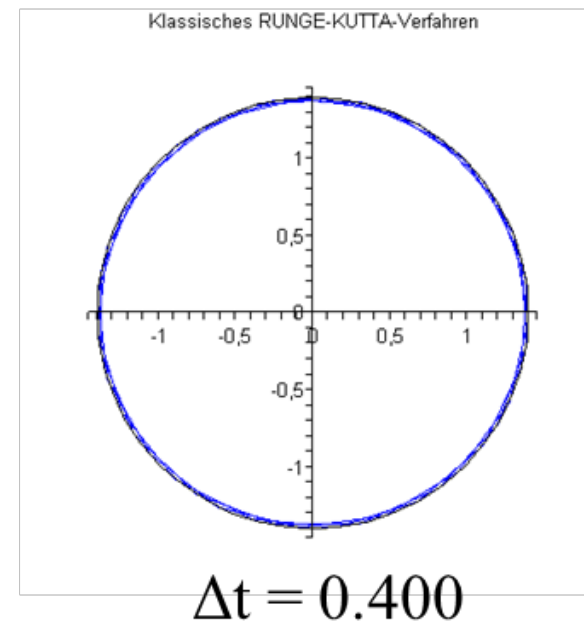
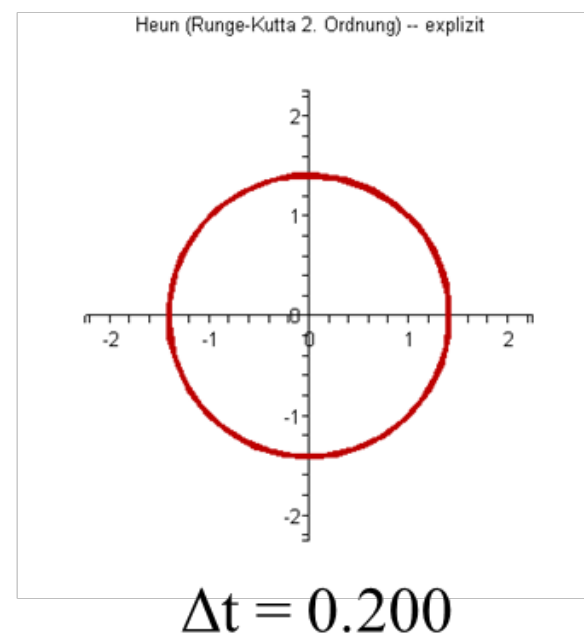
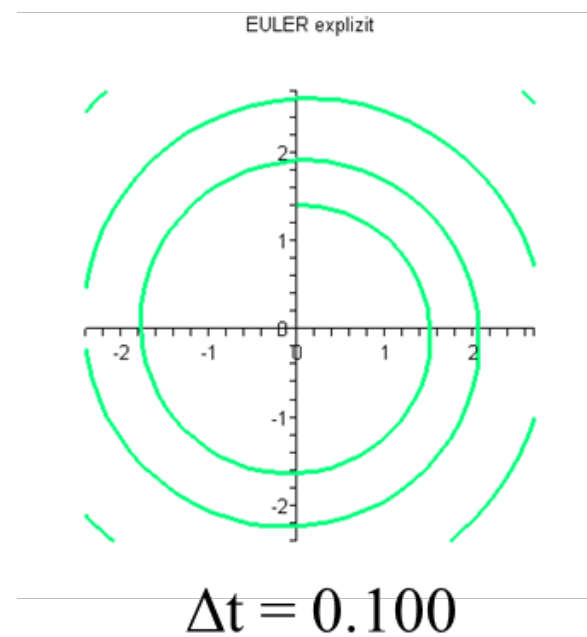
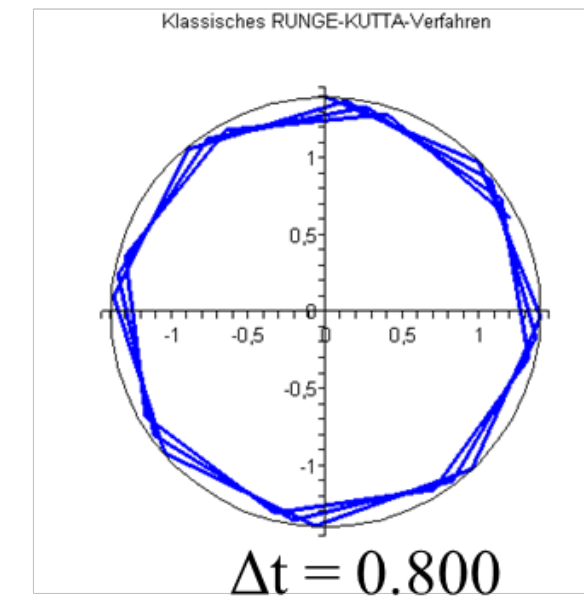
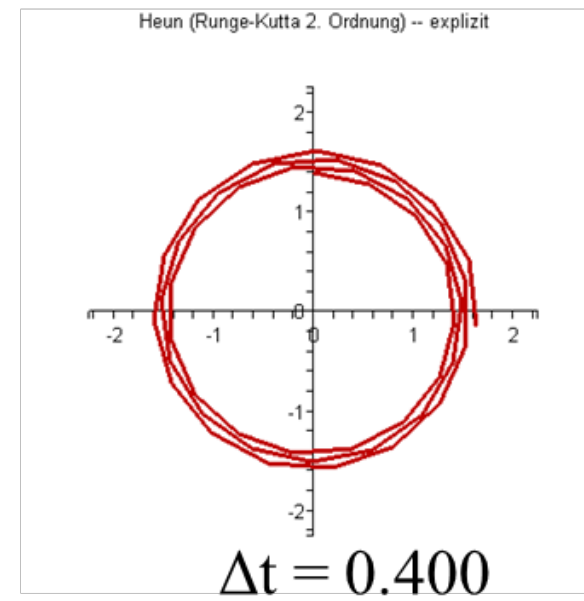
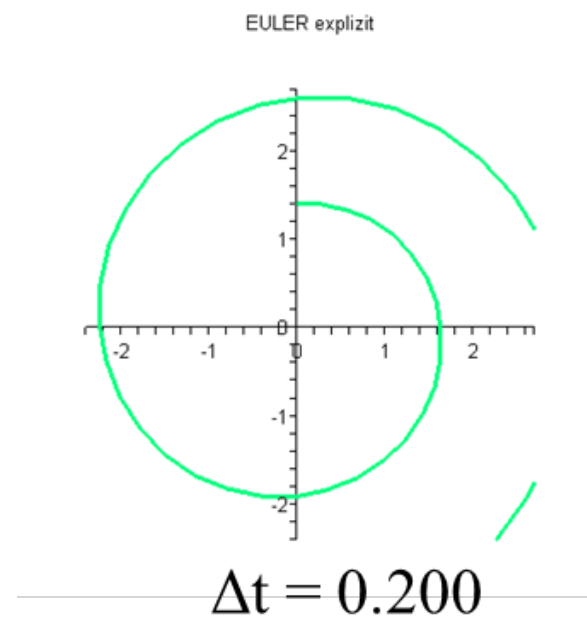


$\Delta t = 0.100$



Particle Tracing

Comparison & dependency on Δt

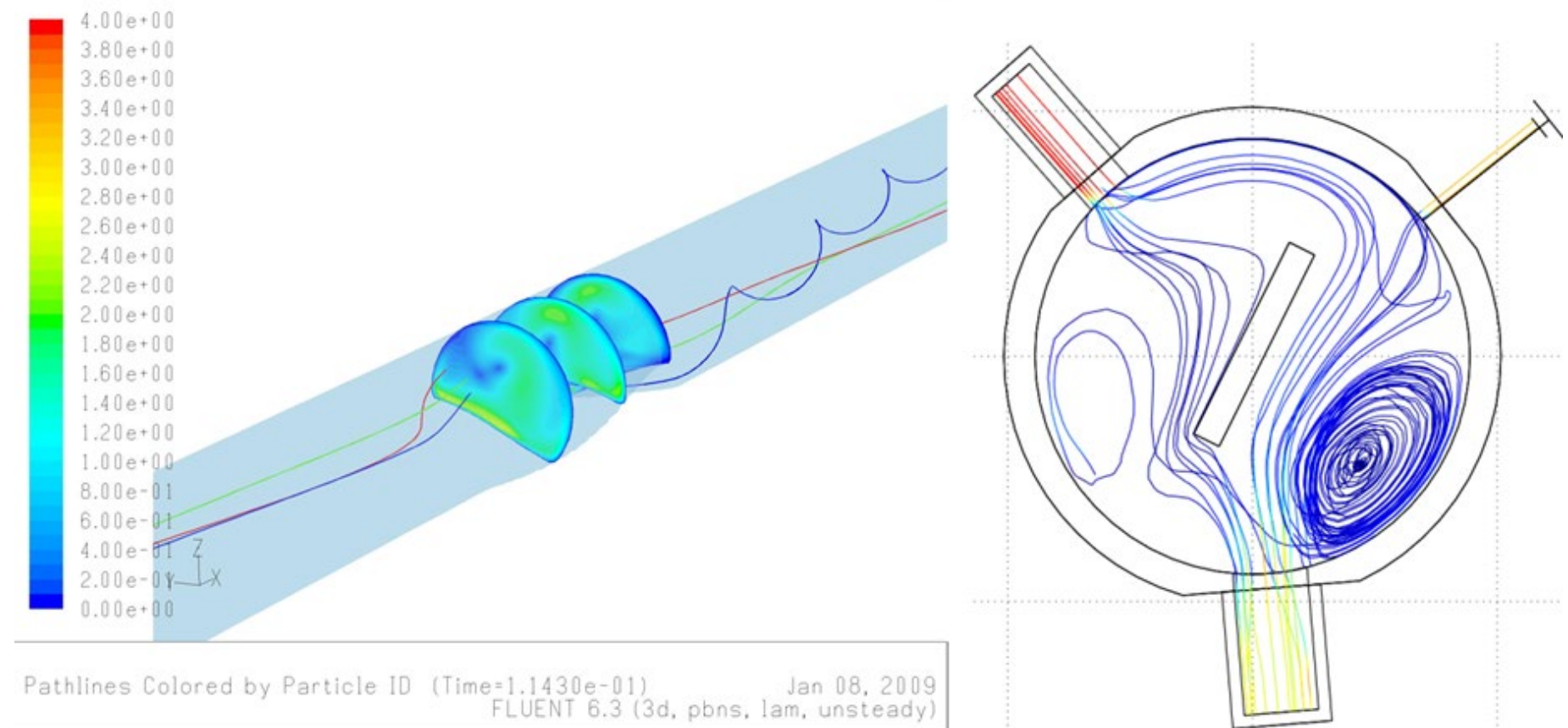


Particle Tracing

- Issues with numerical integration
 - Stability
 - Order of error
 - Step size control
 - Computational effort

Particle Tracing

- Fundamental algorithm
 - Pick a point (= particle) in the data region
 - Determine its path (= integral curve) emanating at this point
- Interpretation
 - Trace the particle without mass in the vector field flow

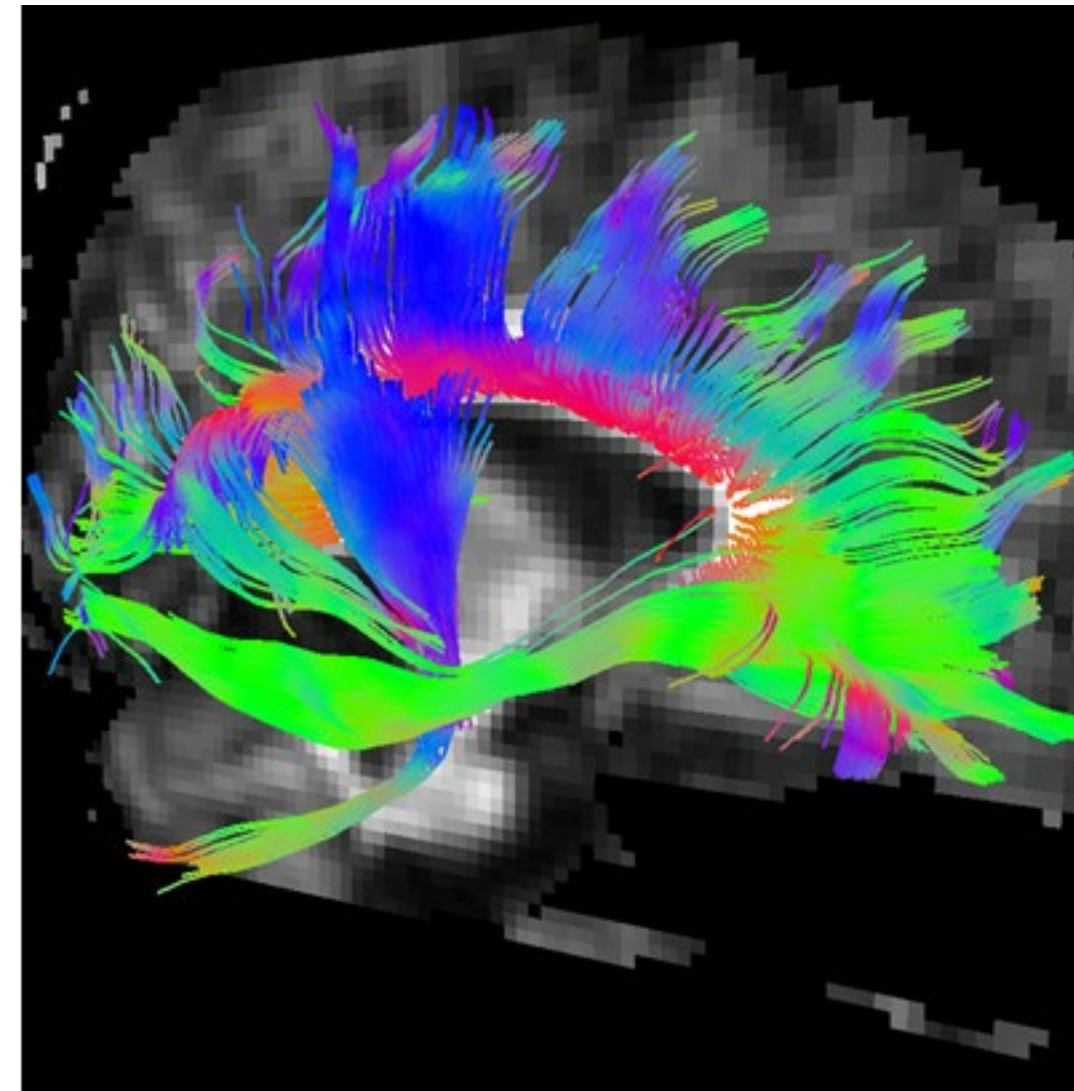
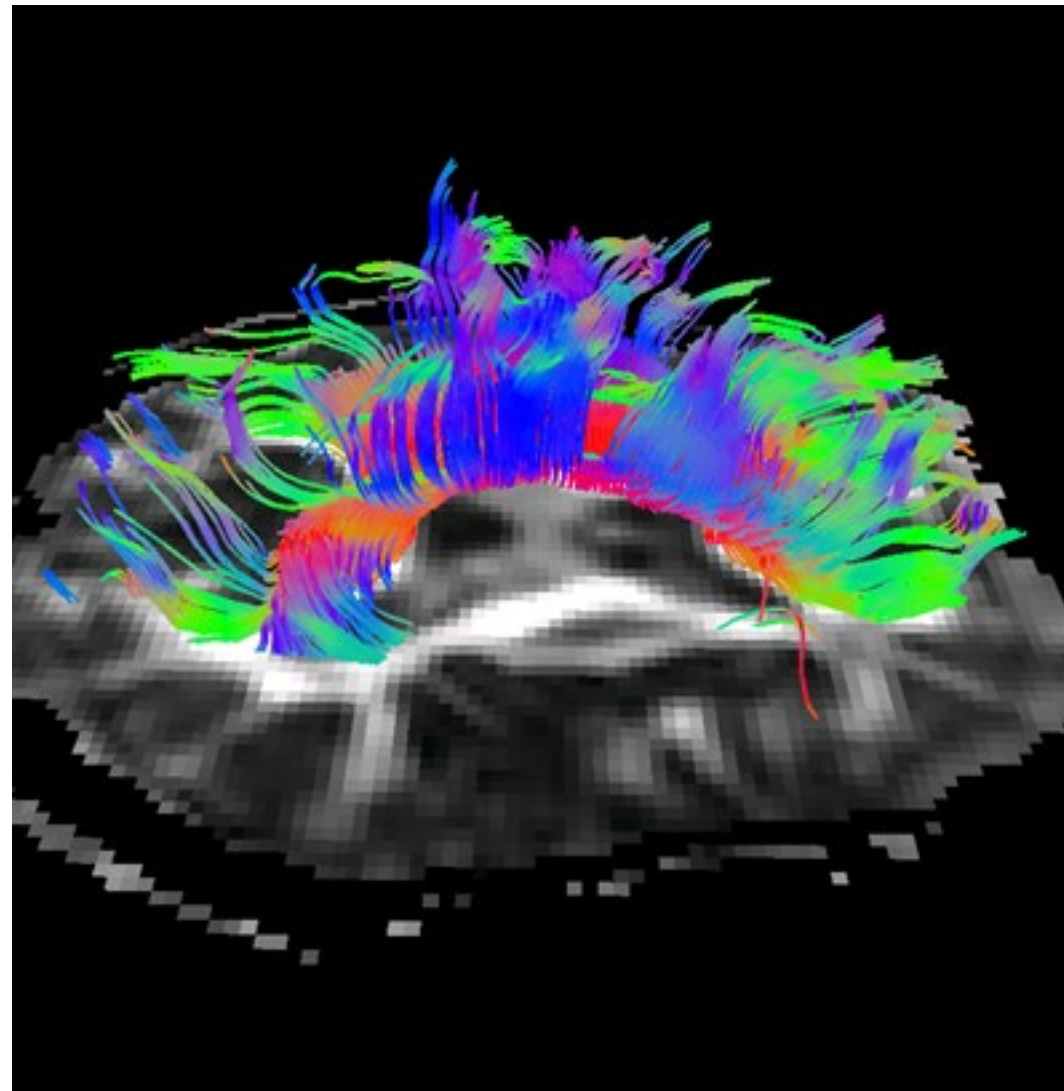


Particle Tracing

- Assumptions
 - Consider only stationary fields $v = v(r)$
 - Time-dependent fields not covered in this lecture
- The vector field
 - Is not given as an analytic function
 - It is defined on a discrete mesh (rectilinear, structured, unstructured, ...)
 - Values "in-between" have to be interpolated!

Particle Tracing

Example: Fiber tracking in a human brain



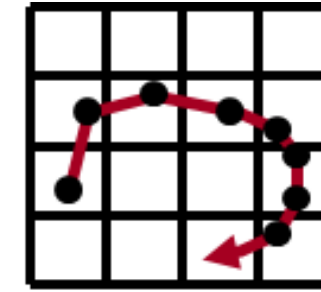
Particle Tracing

Fundamental algorithm

```

Select start point  $r_{old} = r_0$ 
Find cell that contains  $r_{old}$ 
Set step size:  $\Delta t = \dots \leq$ 
while (particle in domain) do
    Determine vector field at  $r_{old}$ 
    Integrate to new position  $r_{new}$ 
    Perform step size control
        (i.e.:  $\Delta t = \Delta t/2$  or  $\Delta t = 2\Delta t$ )
    Draw line segment  $[r_{old}, r_{new}]$ 
     $r_{old} = r_{new}$ 
    Find cell that contains  $r_{old}$ 
endwhile

```



// point location

// $\frac{1}{2}$ (average cell size)

// interpolation

// integration

// optional

// point location

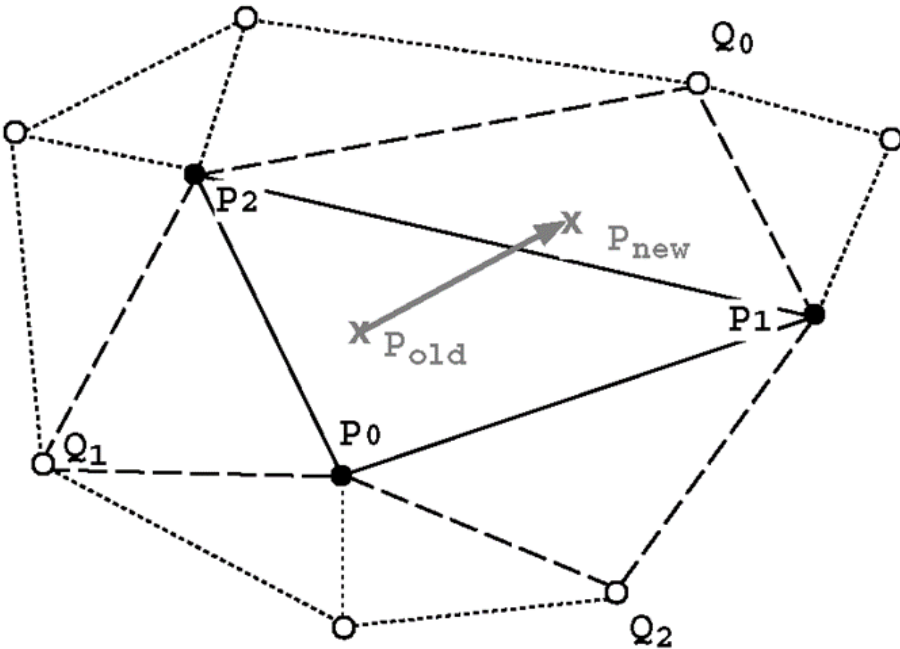
Particle Tracing

- How to do cell search?
 - Depends on the type of mesh!
 - Global search only at the very beginning
 - Later on, only local search
 - Because we know the cell of a point that is close
- Overview
 - Rectilinear meshes
 - No problem
 - Unstructured triangular meshes
 - Barycentric coordinates
 - Curvilinear grids
 - P-Space / C-Space algorithm

Particle Tracing

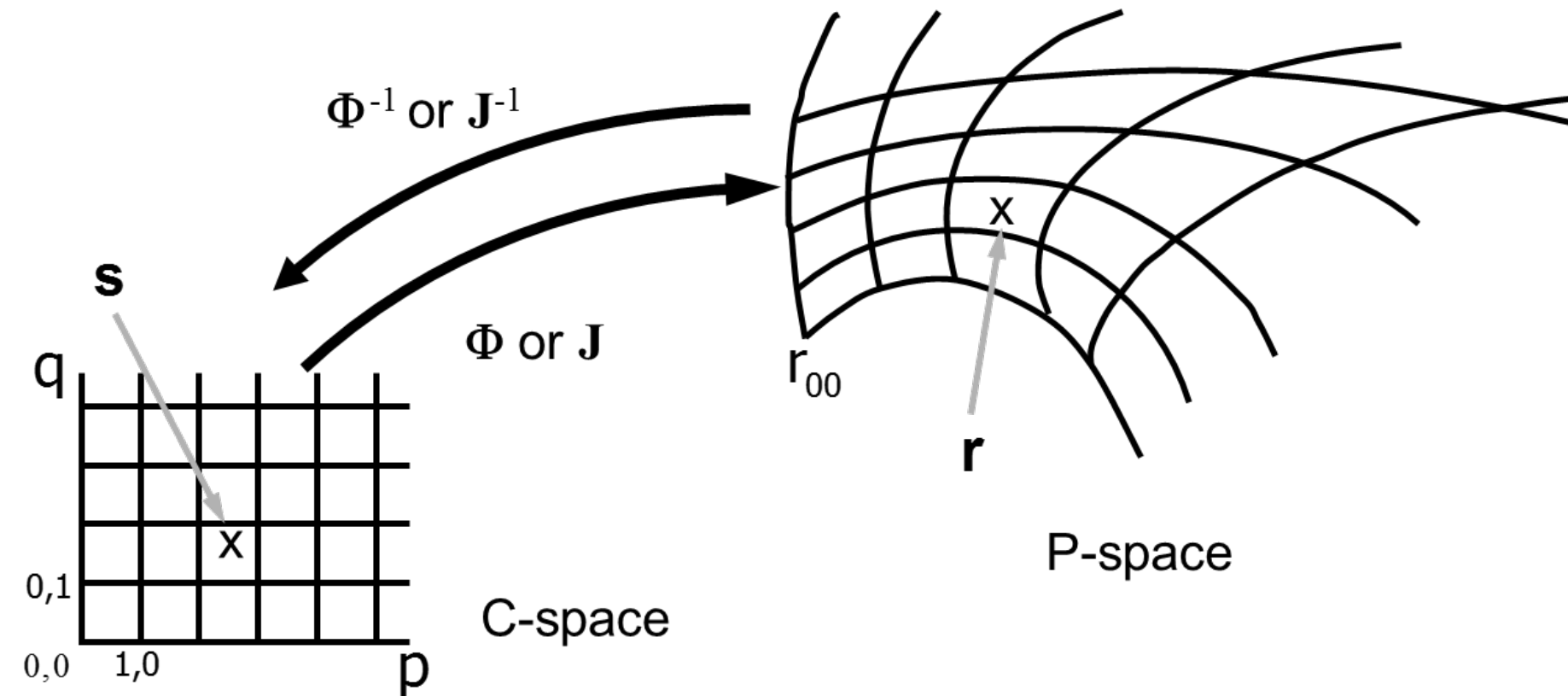
- Cell search in rectilinear meshes
 - Index of cell directly from position (x,y,z)
 - e.g. $\Delta x = \frac{x_{\max} - x_{\min}}{n}$ $i = \left\lfloor \frac{x_p}{\Delta x} \right\rfloor$
- Unstructured triangular meshes
 - Barycentric coordinates of P_{new} : d_0, d_1, d_2

$d_0 < 0$	$d_1 < 0$	$d_2 < 0$	
0	0	0	$P_{\text{new}} \in \Delta(P_0, P_1, P_2)$
0	0	1	check $P_{\text{new}} \in \Delta(Q_2, P_1, P_0)$ recursively
0	1	0	check $P_{\text{new}} \in \Delta(Q_1, P_0, P_2)$ recursively
1	0	0	check $P_{\text{new}} \in \Delta(Q_0, P_2, P_1)$ recursively
0	1	1	process fan centered at P_0
1	0	1	process fan centered at P_1
1	1	0	process fan centered at P_2
1	1	1	error



Particle Tracing

- Curvilinear grids

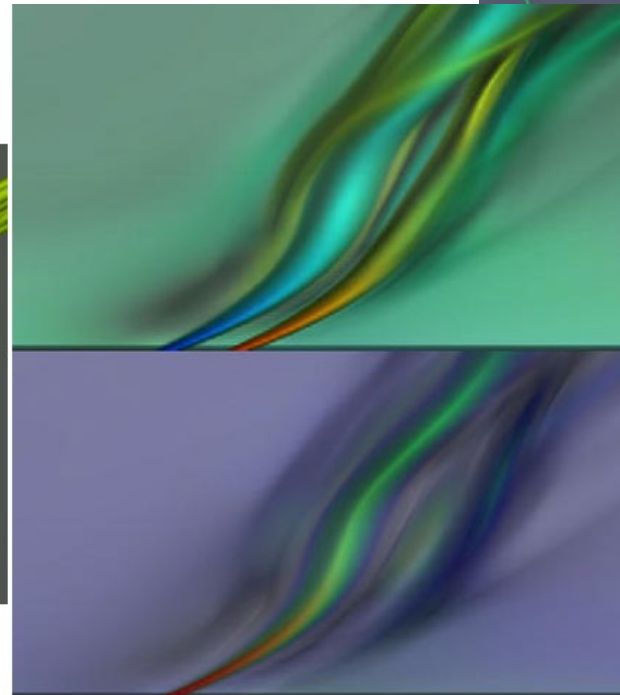
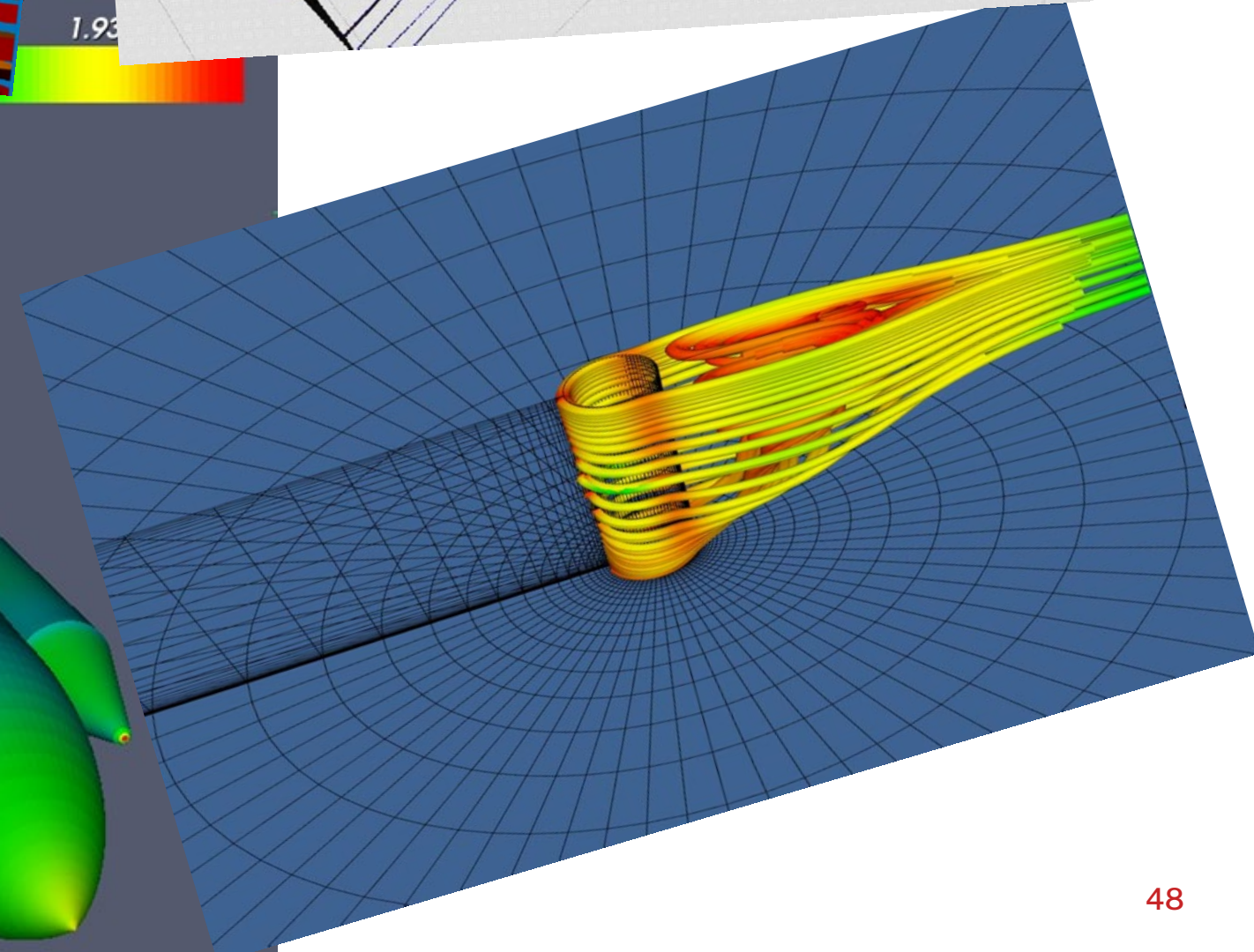
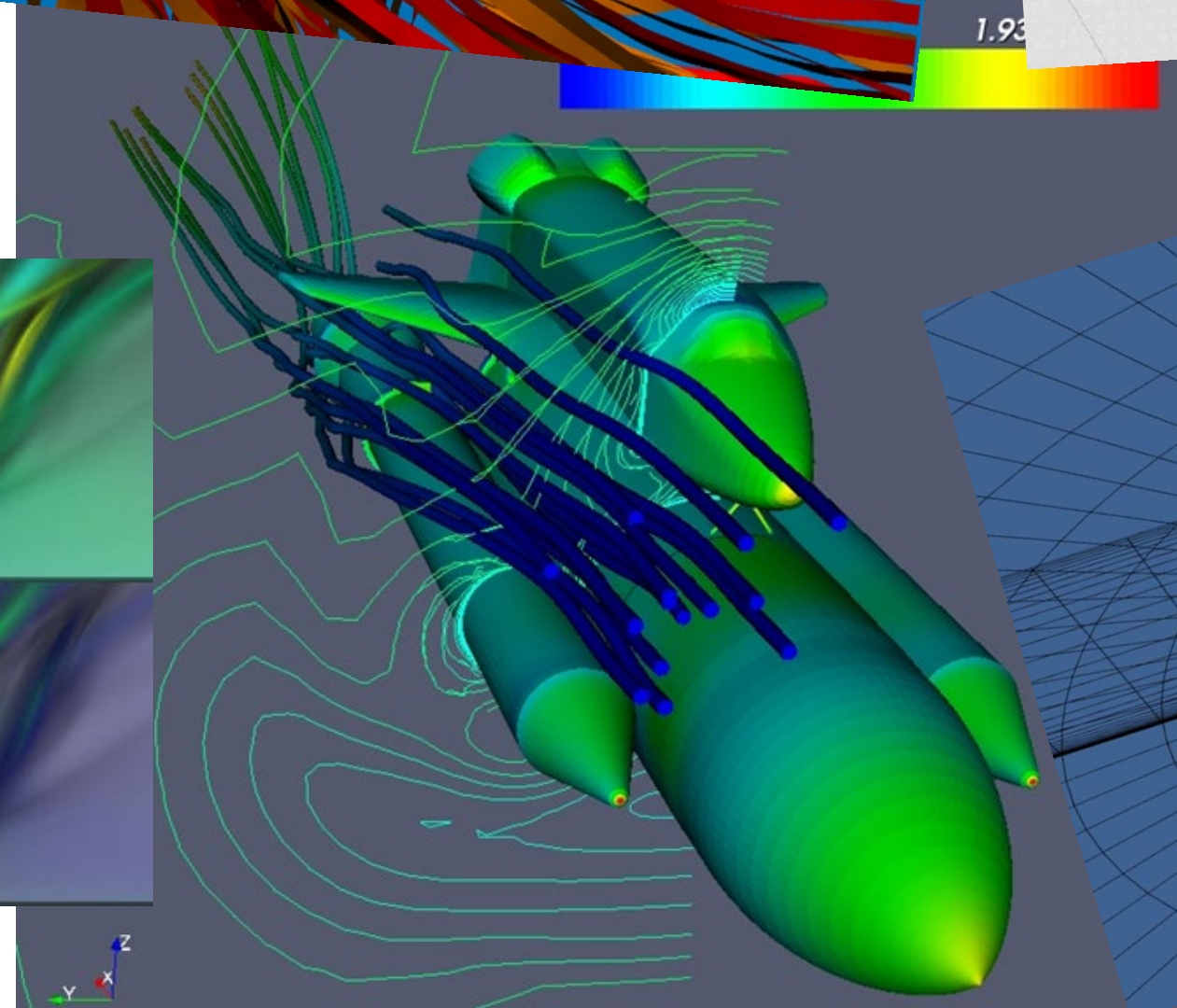
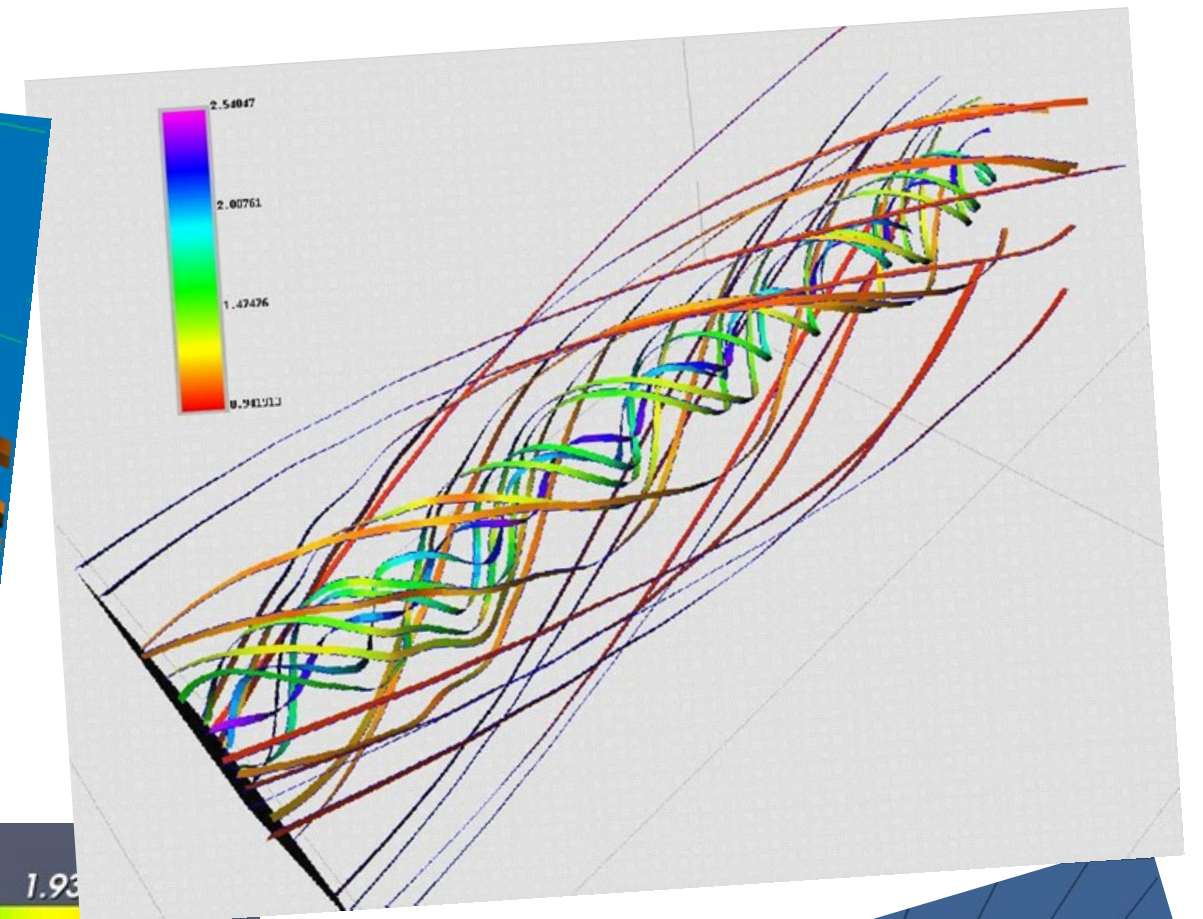
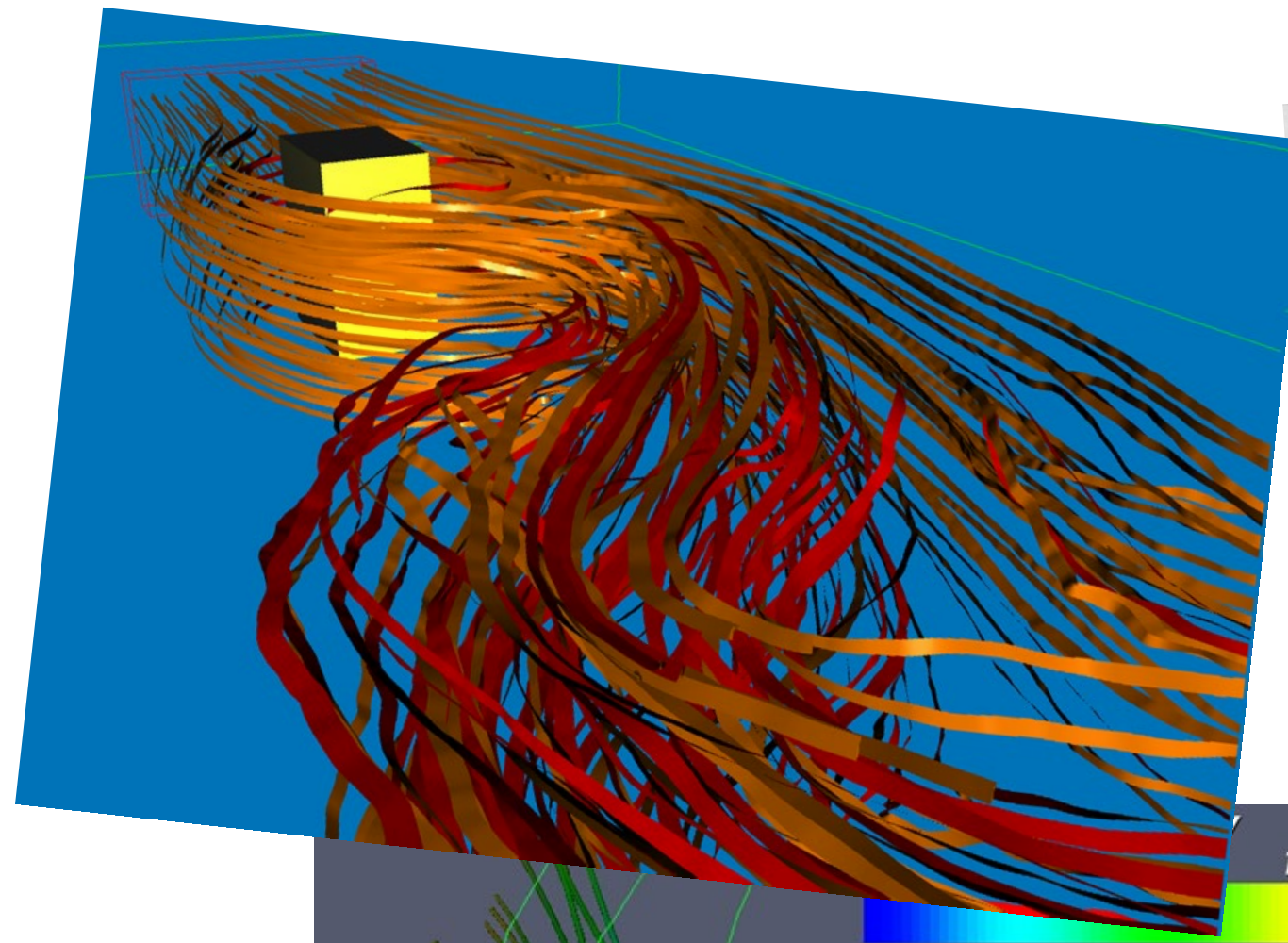
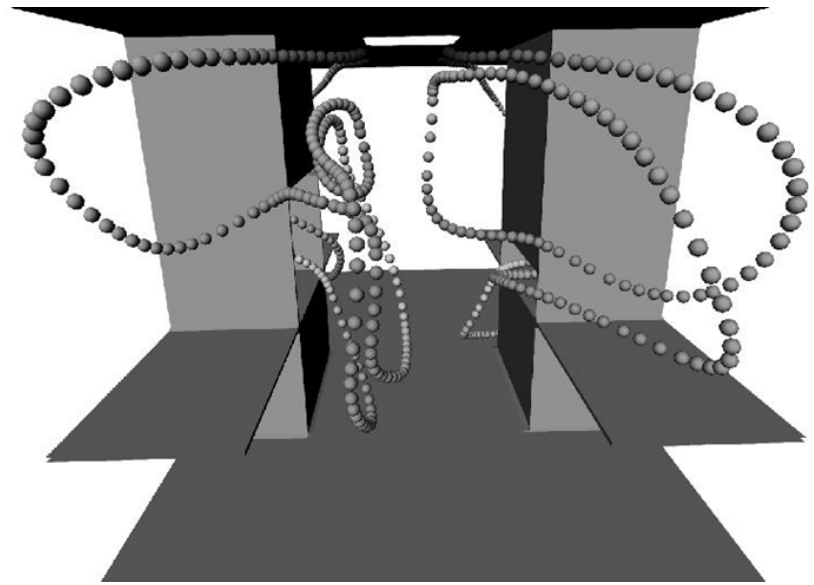


- Transformation of spaces
- This is considerably more involved than processing structured grids!

Particle Tracing

- Based on the fundamental particle tracing algorithm
 - Path lines / surfaces
 - Time lines / surfaces
 - Stream lines / surfaces
 - Ribbons
 - Tubes
 - and many more ...

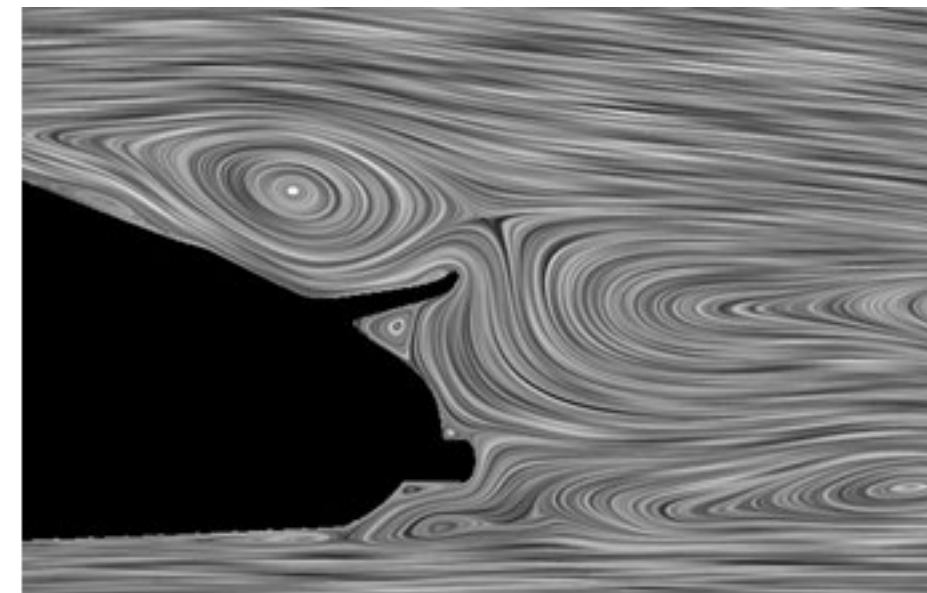
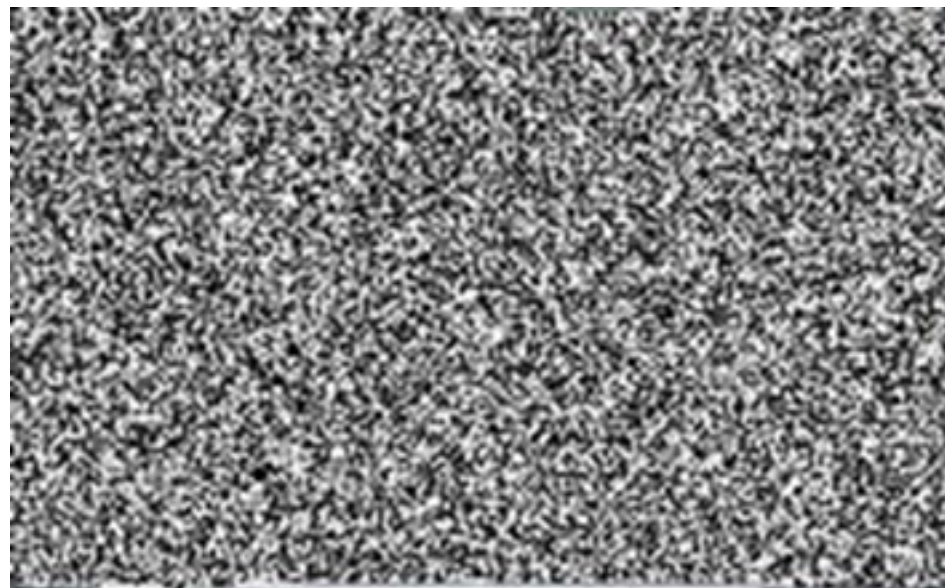
Examples



5.4 Line Integral Convolution

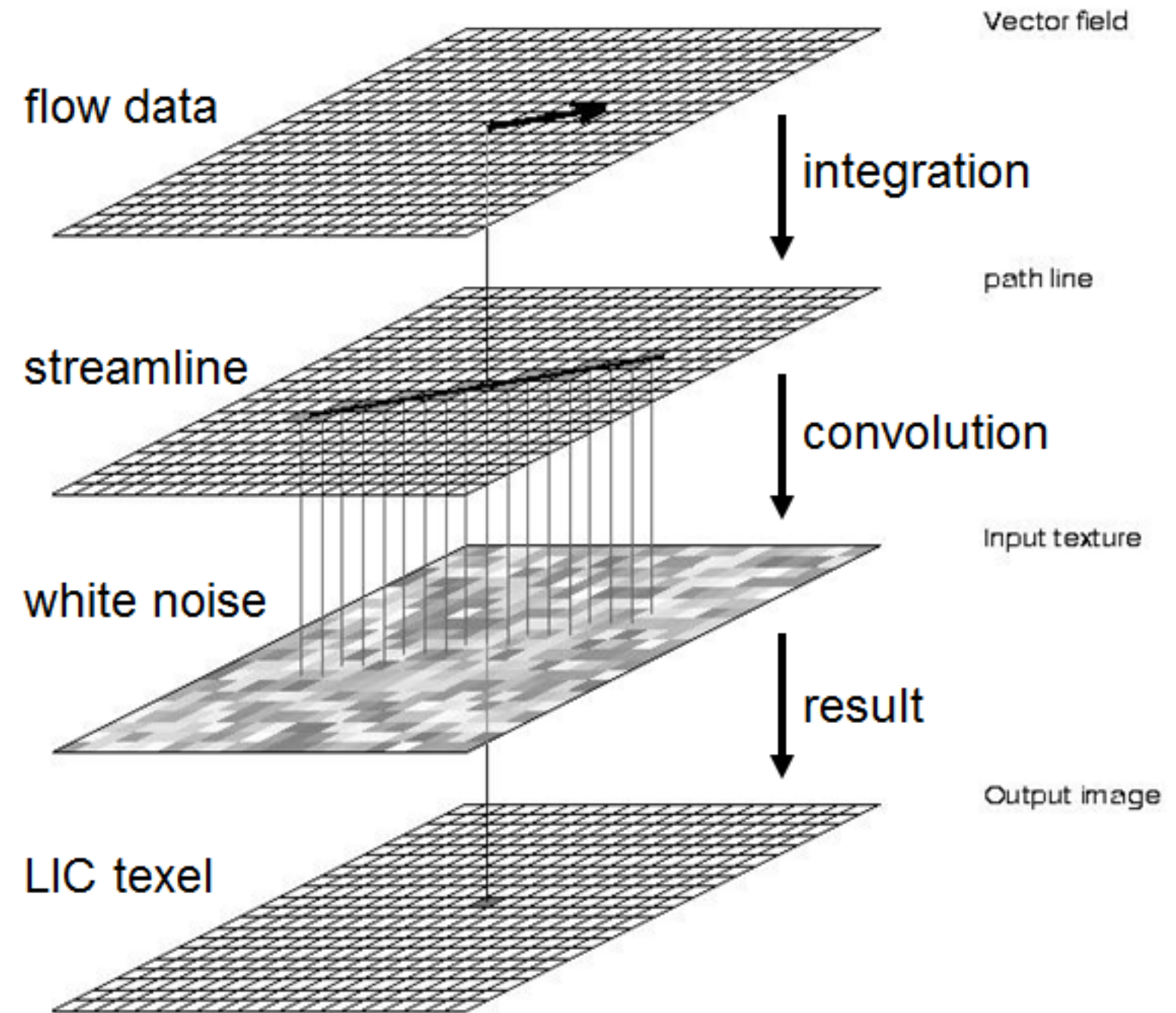
(LIC)

- Main Idea
 - Traditional approach inappropriate for dense vector fields
 - "Image" integral curves
 - Cover domain with random texture (input texture)
 - Stationary white noise
 - Convolve (blur) input texture along the path lines

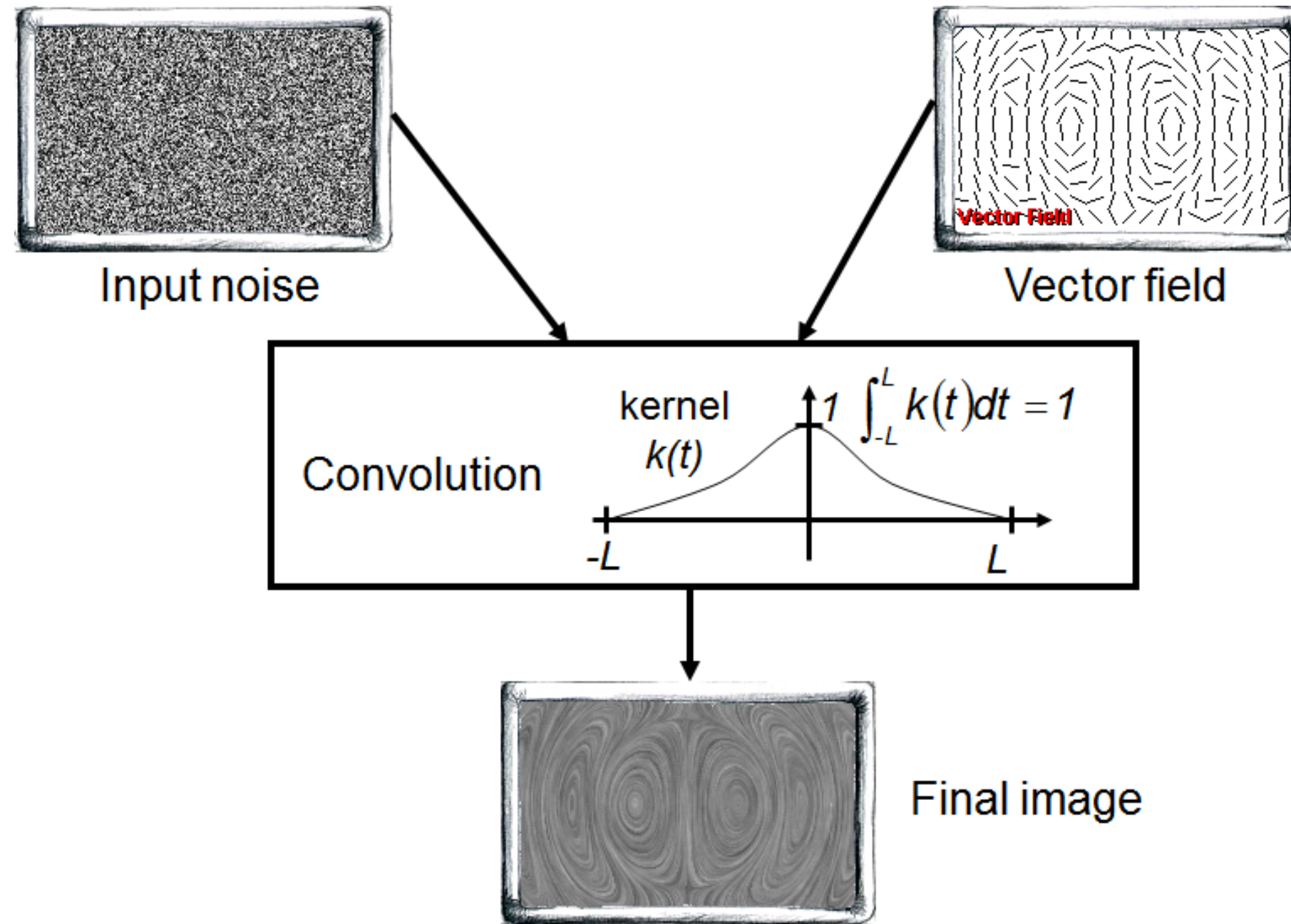


LIC

- Algorithm for 2D LIC
 - Convolve random texture along the streamlines

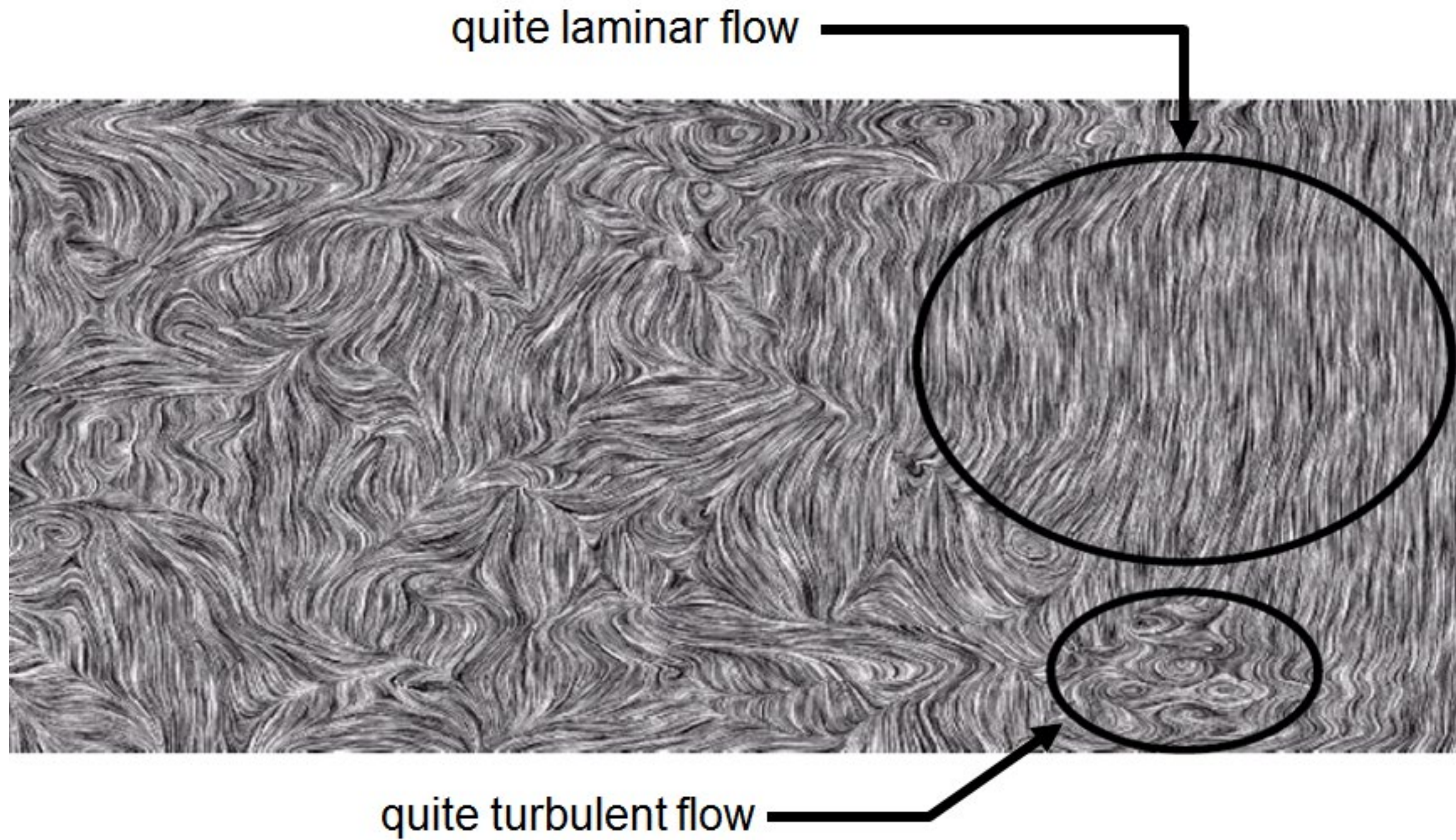


Idee: Bildfaltung

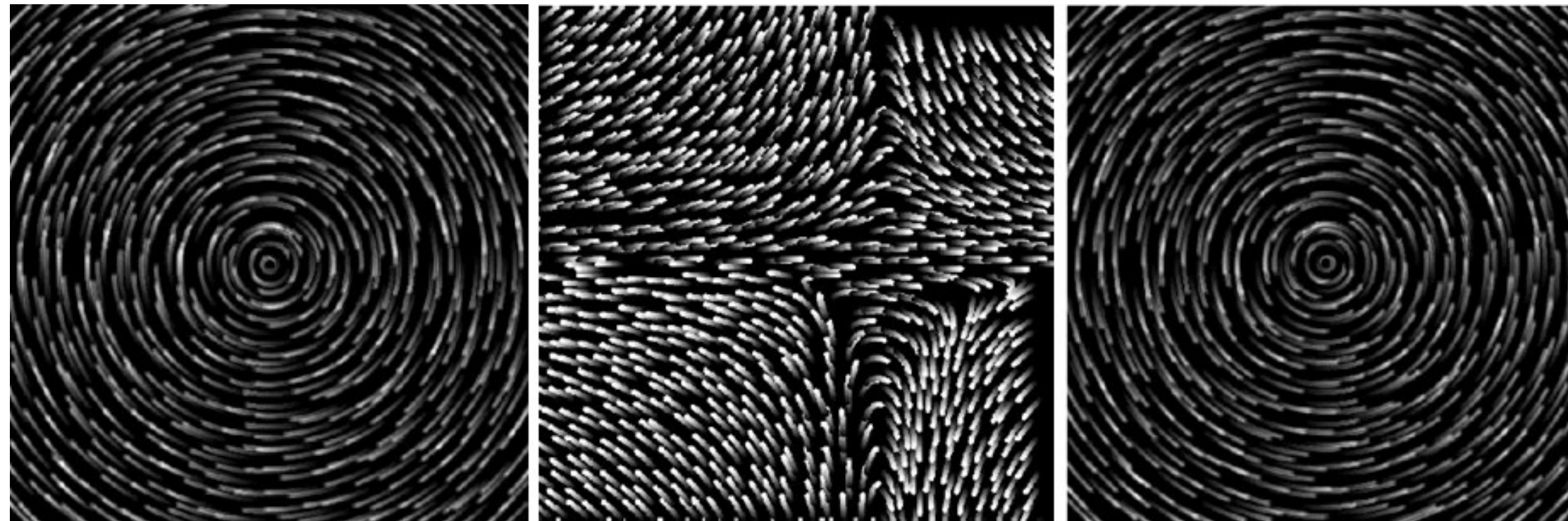
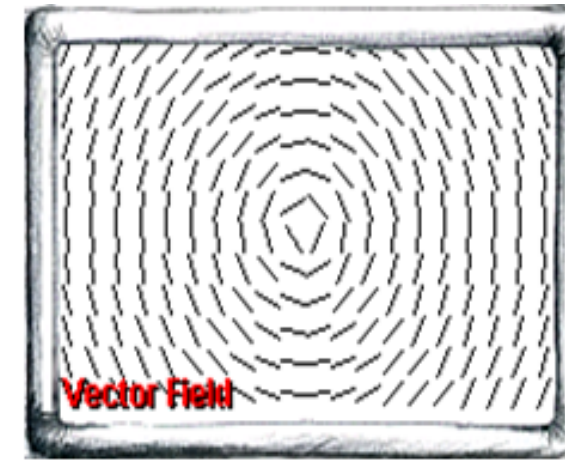
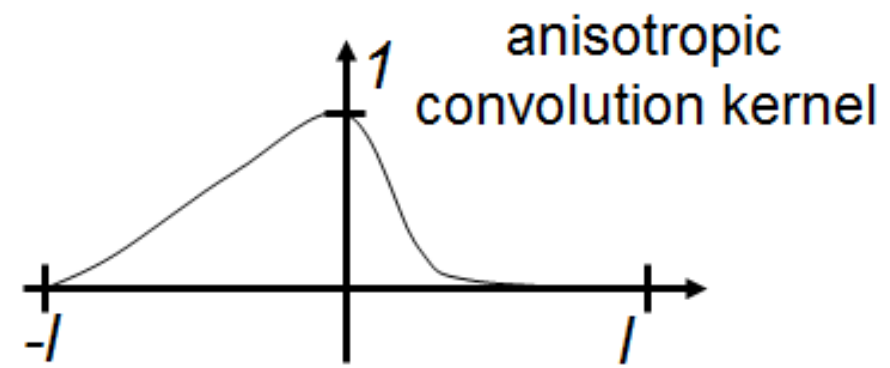
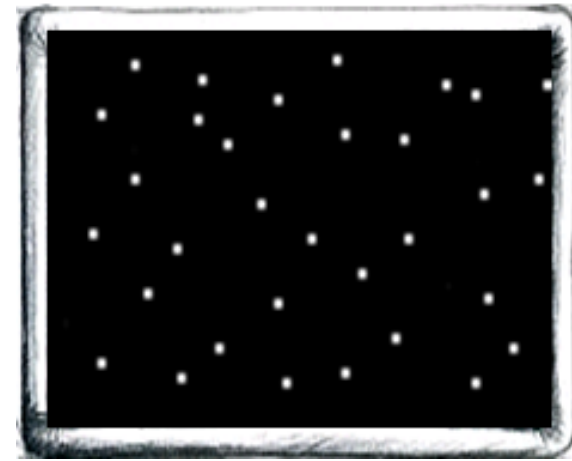


ohm

LIC

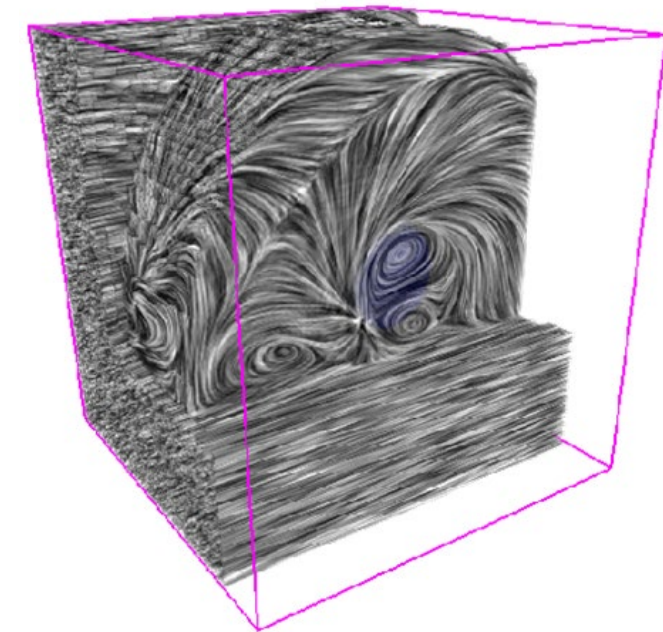
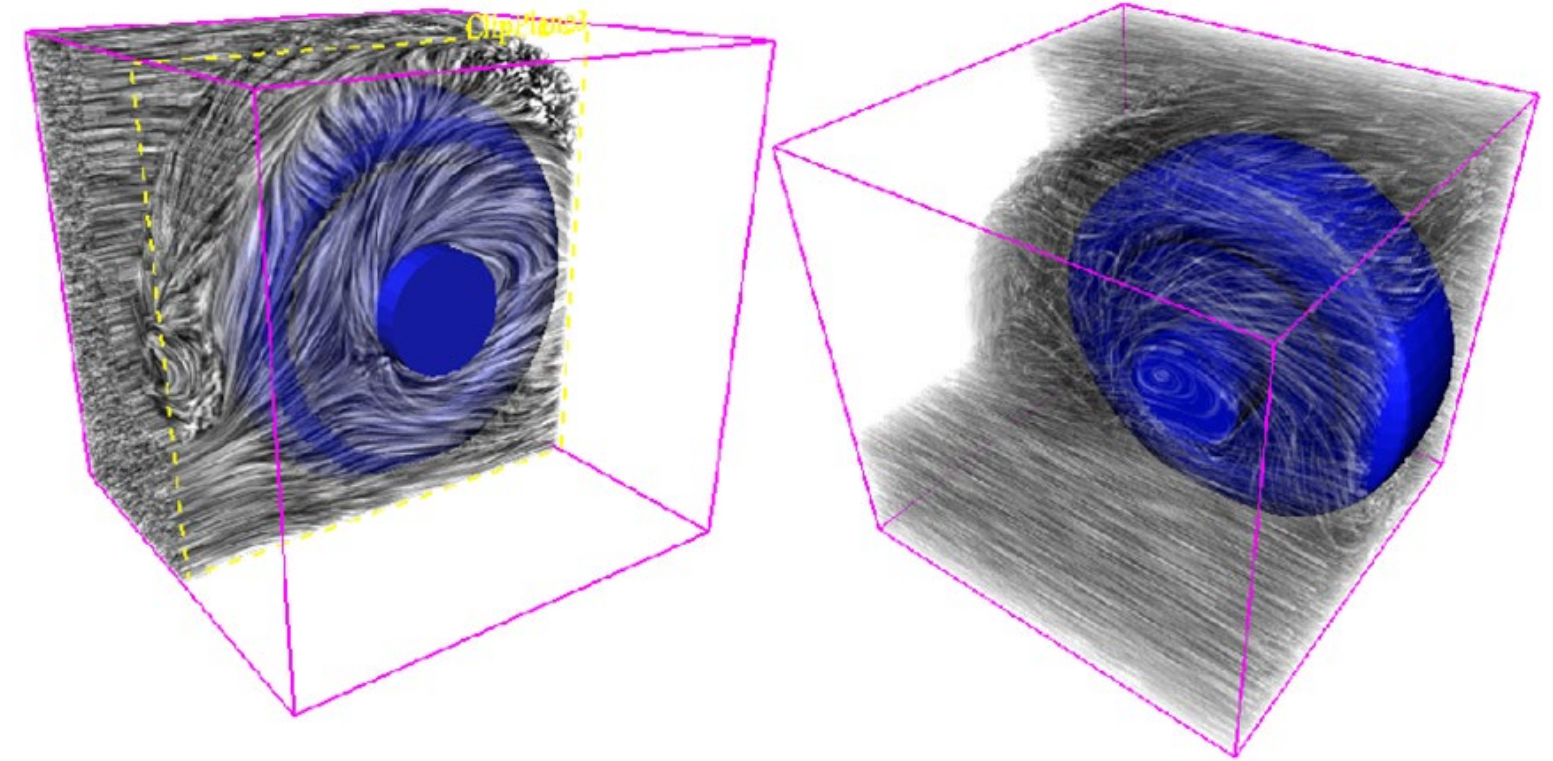
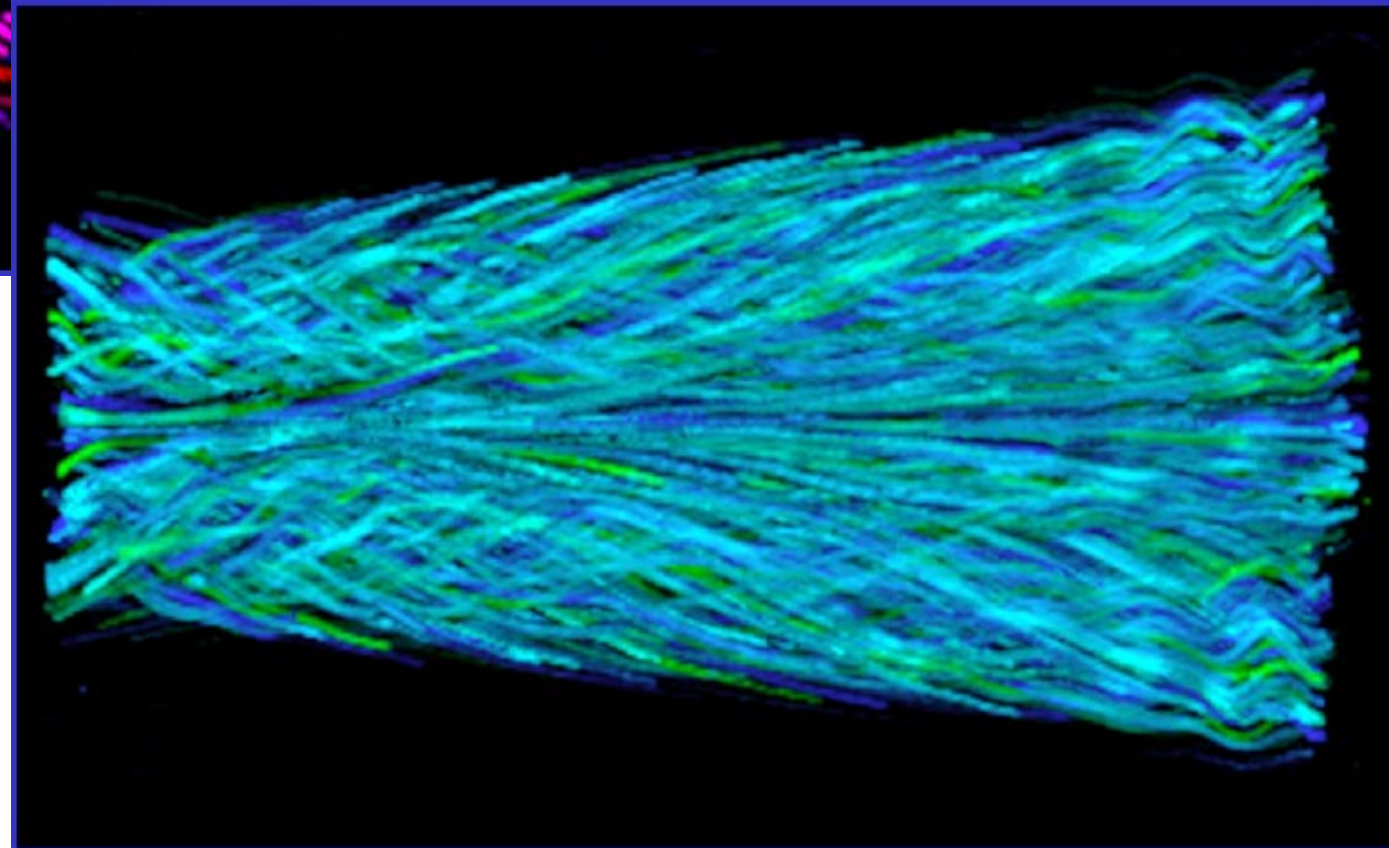
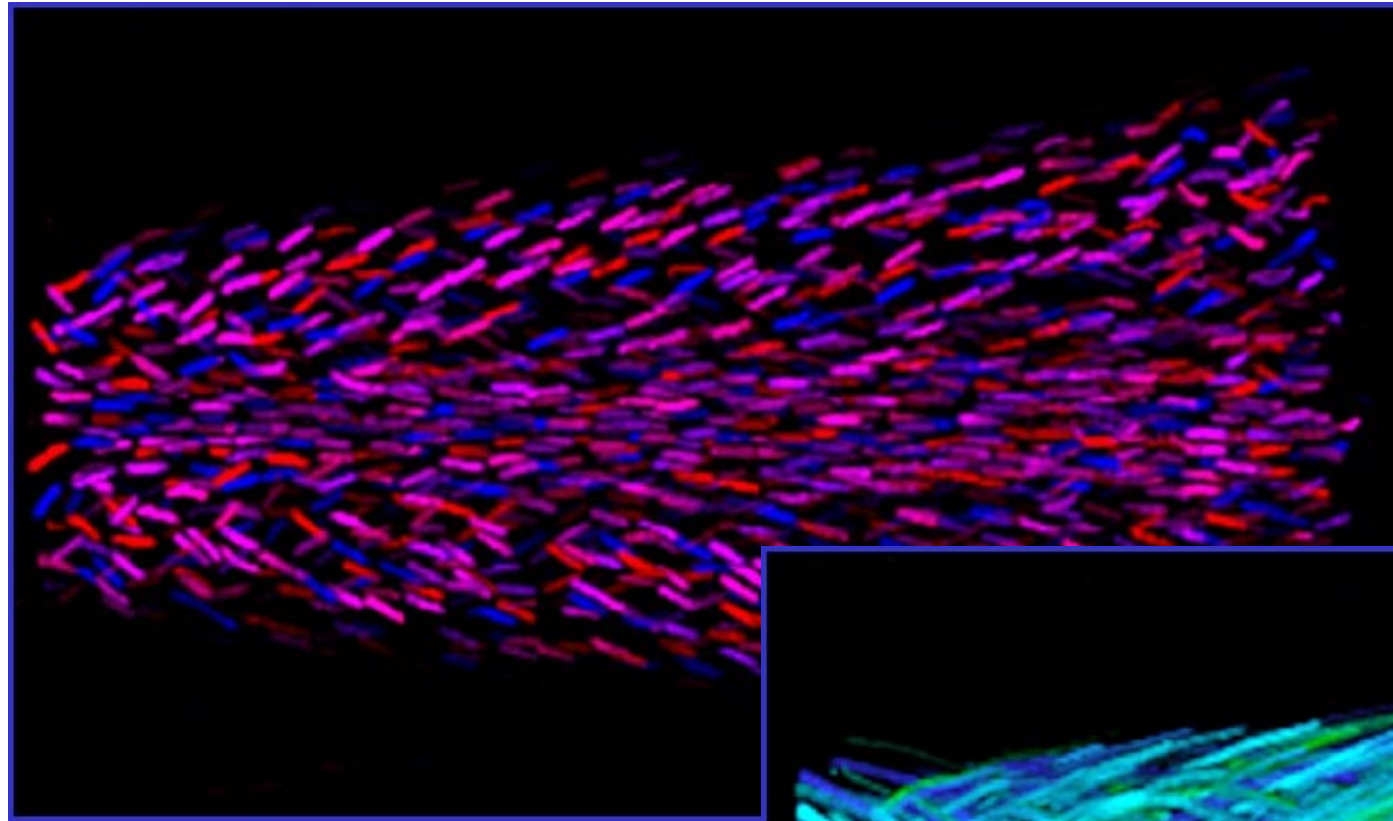


Oriented LIC (OLIC)



$\Omega \mathbf{h} \mathbf{m}$

3D LIC



3D LIC

