CS 247 – Scientific Visualization
Lecture 28: Vector / Flow Visualization, Pt. 7

Markus Hadwiger, KAUST

Reading Assignment #14 (until May 11)

Read (required):

- Data Visualization book, Chapter 6.7
- J. van Wijk: *Image-Based Flow Visualization*,
  ACM SIGGRAPH 2002

  `http://www.win.tue.nl/~vanwijk/ibfv/ibfv.pdf`

Read (optional):

- T. Günther, A. Horvath, W. Bresky, J. Daniels, S. A. Buehler:
  Lagrangian Coherent Structures and Vortex Formation in High Spatiotemporal-Resolution Satellite Winds of an Atmospheric Karman Vortex Street, 2021

  `https://www.essoar.org/doi/10.1002/essoar.10506682.2`
- H. Bhatia, G. Norgard, V. Pascucci, P.-T. Bremer:
  The Helmholtz-Hodge Decomposition – A Survey, TVCG 19(8), 2013

  `https://doi.org/10.1109/TVCG.2012.316`
- Work through online tutorials of multi-variable partial derivatives, grad, div, curl, Laplacian:

  `https://www.khanacademy.org/math/multivariable-calculus/multivariable-derivatives`
  `https://www.youtube.com/watch?v=rB83DpBJQsE` (3Blue1Brown)
- Matrix exponentials:

  `https://www.youtube.com/watch?v=O85OWBJ2ayo` (3Blue1Brown)

Quiz #3: May 14?

Organization

- First 30 min of lecture
- No material (book, notes, ...) allowed

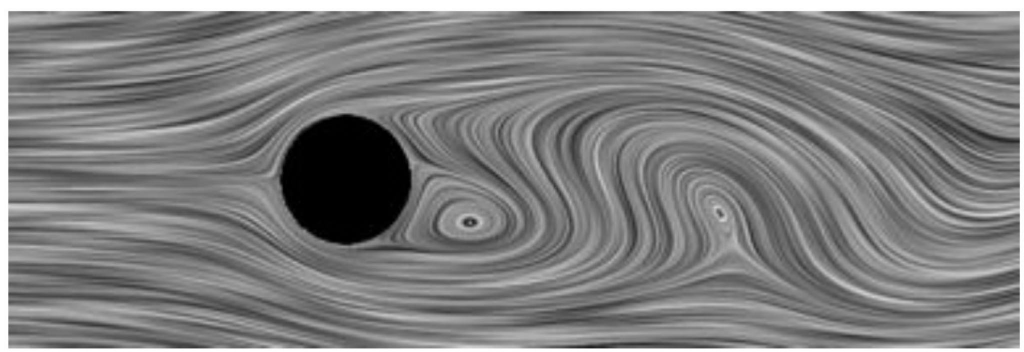
Content of questions

- Lectures (both actual lectures and slides)
- Reading assignments (except optional ones)
- Programming assignments (algorithms, methods)
- Solve short practical examples

Line Integral Convolution (LIC)

Line Integral Convolution

- Line Integral Convolution (LIC)
  - Visualize dense flow fields by imaging its integral curves
  - Cover domain with a random texture (so called ‚input texture‘, usually stationary white noise)
  - Blur (convolve) the input texture along stream lines using a specified filter kernel
- Look of 2D LIC images
  - Intensity distribution along stream lines shows high correlation
  - No correlation between neighboring stream lines

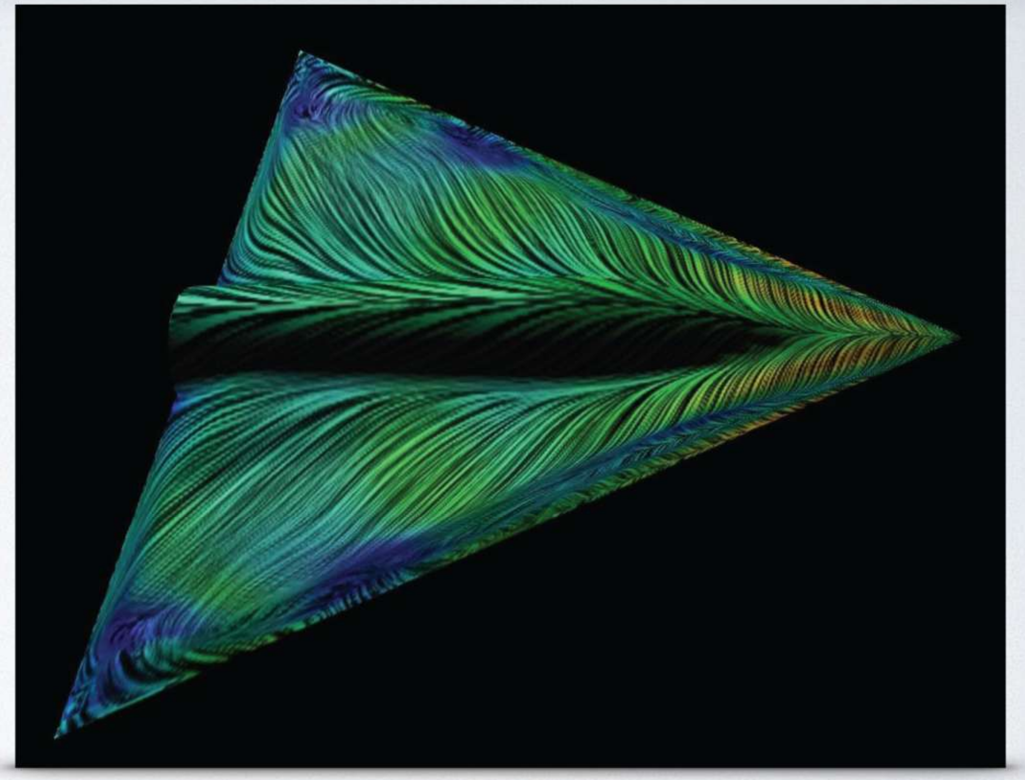


Line Integral Convolution I

- Line Integral Convolution (LIC):
  - goal: general overview of flow
  - approach: use dense textures
  - idea: flow ↔ visual correlation

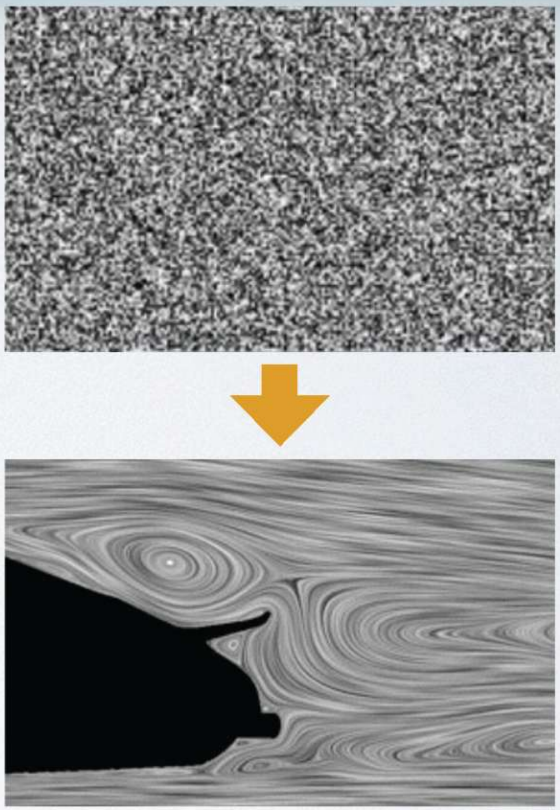
Line Integral Convolution I

- Line Integral Convolution (LIC):
  - goal: general overview of flow
  - approach: use dense textures
  - idea: flow ↔ visual correlation



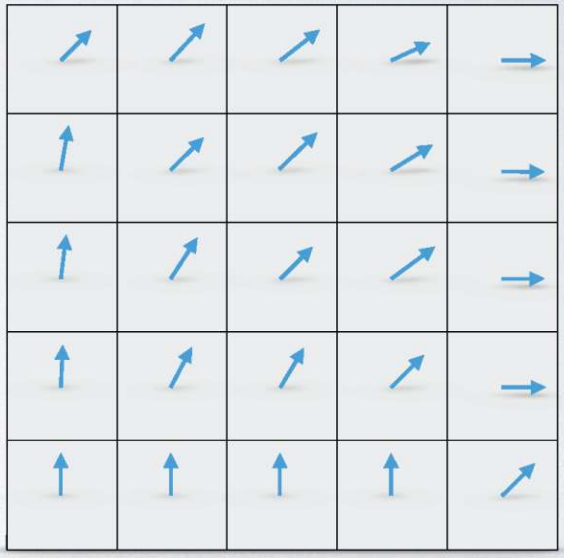
Line Integral Convolution II

- Idea
  - global visualization technique
  - dense representation
  - start with random texture
  - smear along stream lines
- Only for stream lines!
  (steady flow, i.e. time-independent fields)



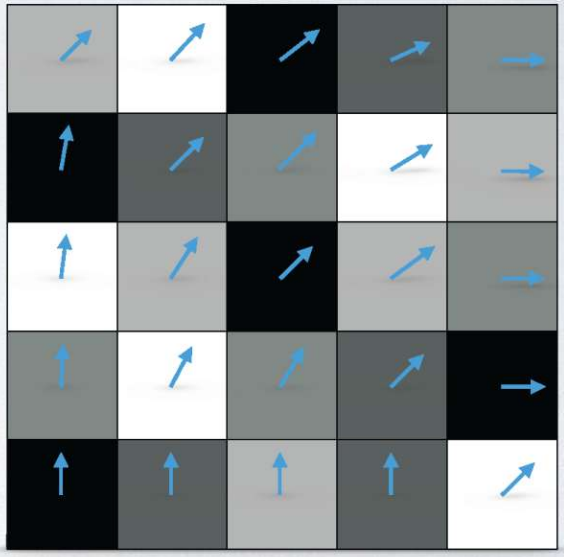
Line Integral Convolution III

- How LIC works
  - visualize dense flow fields by imaging integral curves
  - cover domain with a random texture ('input texture', usually stationary white noise)
  - blur (convolve) the input texture along stream lines



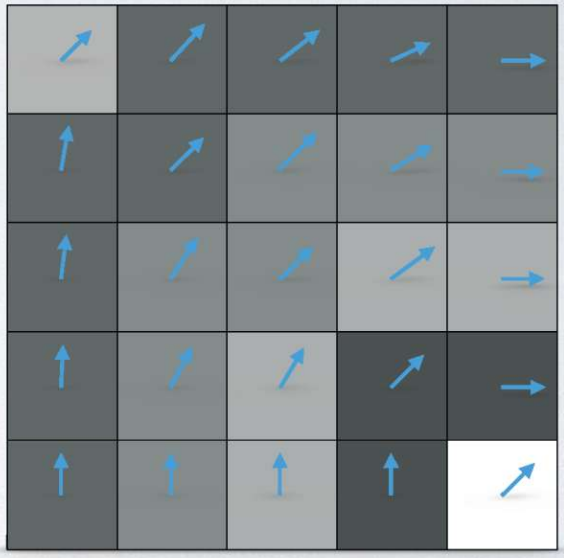
Line Integral Convolution III

- How LIC works
  - visualize dense flow fields by imaging integral curves
  - cover domain with a random texture
    ('input texture', usually stationary white noise)
  - blur (convolve) the input texture along stream lines



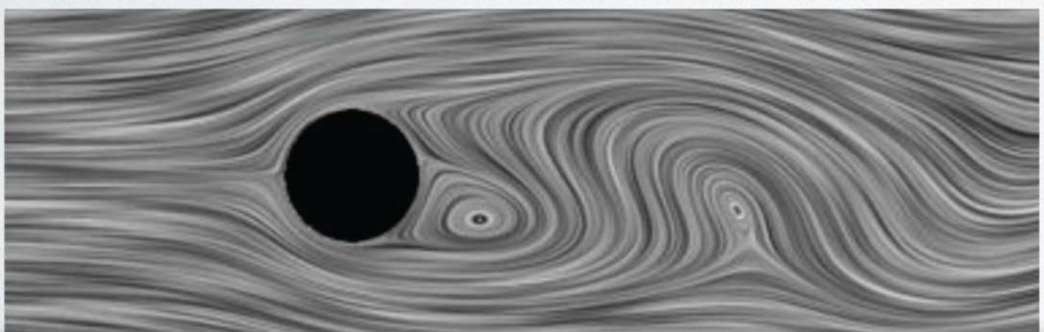
Line Integral Convolution III

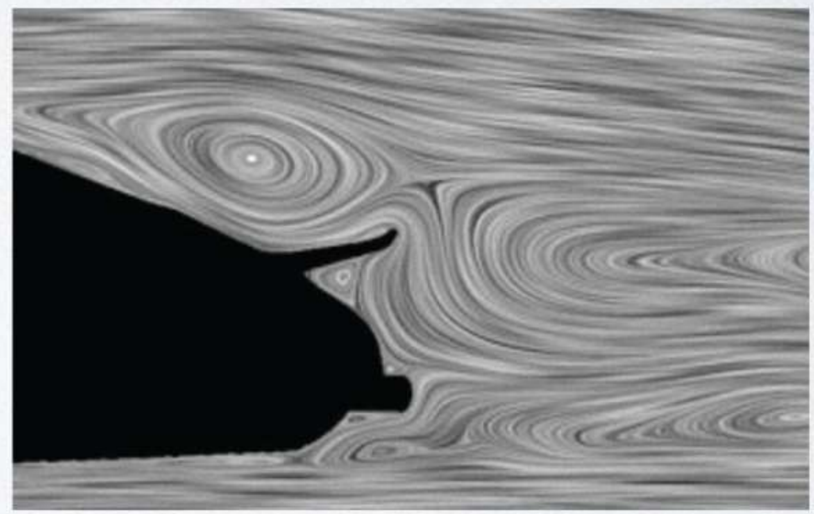
- How LIC works
  - visualize dense flow fields by imaging integral curves
  - cover domain with a random texture
    ('input texture', usually stationary white noise)
  - blur (convolve) the input texture along stream lines



Line Integral Convolution IV

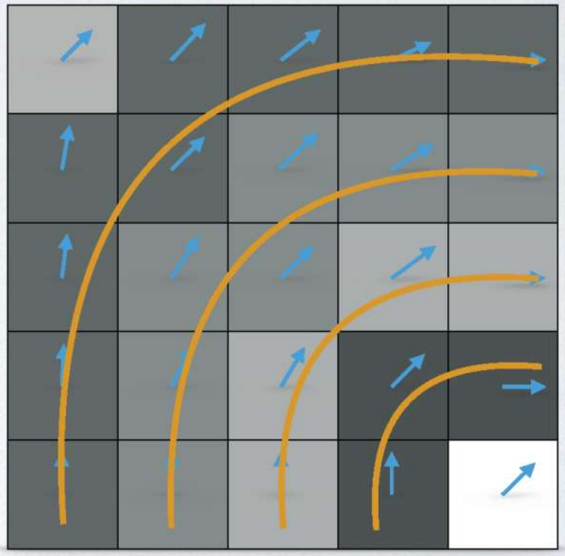
- Look of 2D LIC images
  - intensity along stream lines shows high correlation
  - no correlation between neighboring stream lines





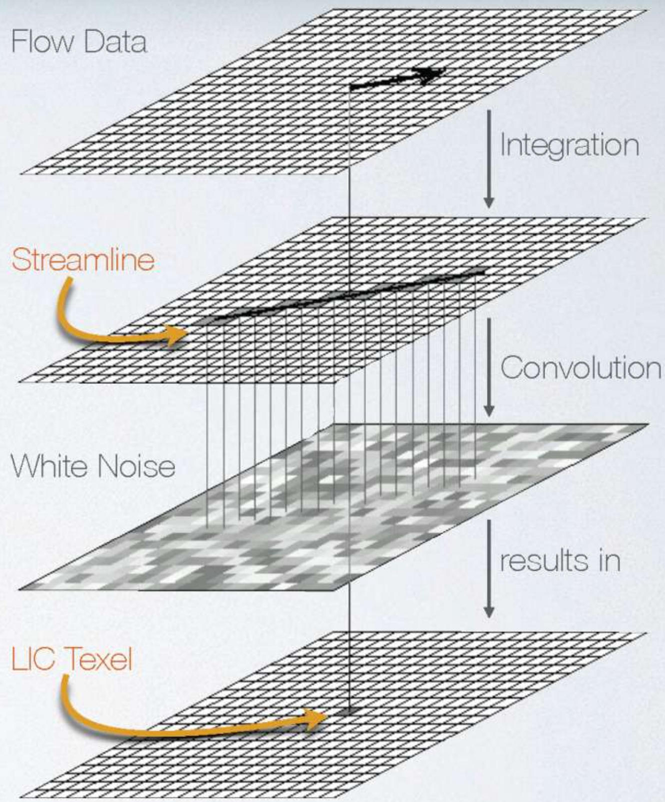
LIC Approach - Goal

- **For every texel**: let the texture value
  - correlate with neighboring texture values along the flow (in flow direction)
  - not correlate with neighboring texture values across the flow (normal to flow direction)
- **Result**: along streamlines the texture values are correlated ⇨ visually coherent!



LIC Approach - Steps

- Idea: "smear" white noise (no a priori correlations) along flow
- Calculation of a texture value:
  - follow streamline through point
  - filter white noise along streamline



Convolution Example

Gaussian Blur

en.wikipedia.org/wiki/Gaussian_blur

Cut off filter kernel after an extent of, e.g., 3*standard deviation in each direction

Example:

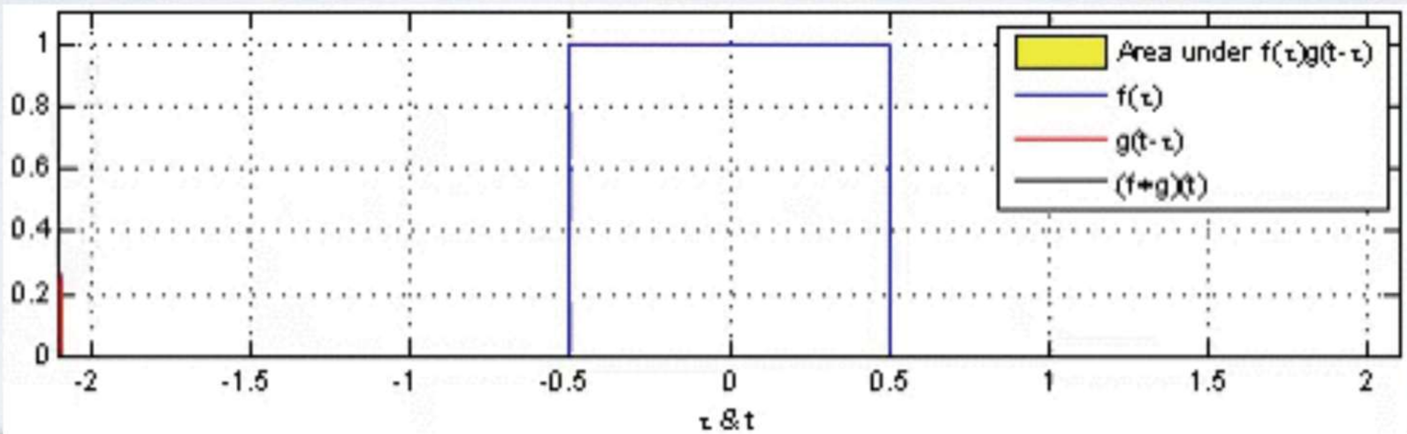
0.00000067	0.00002292	**0.00019117**	0.00038771	**0.00019117**	0.00002292	0.00000067
0.00002292	0.00078634	0.00655965	0.01330373	0.00655965	0.00078633	0.00002292
0.00019117	0.00655965	0.05472157	0.11098164	0.05472157	0.00655965	**0.00019117**
0.00038771	0.01330373	0.11098164	**0.22508352**	0.11098164	0.01330373	0.00038771
0.00019117	0.00655965	0.05472157	0.11098164	0.05472157	0.00655965	**0.00019117**
0.00002292	0.00078633	0.00655965	0.01330373	0.00655965	0.00078633	0.00002292
0.00000067	0.00002292	**0.00019117**	0.00038771	**0.00019117**	0.00002292	0.00000067

Note that 0.22508352 (the central one) is 1177 times larger than 0.00019117 which is just outside 3σ.

Can do multiple iterations to achieve larger effective filter size

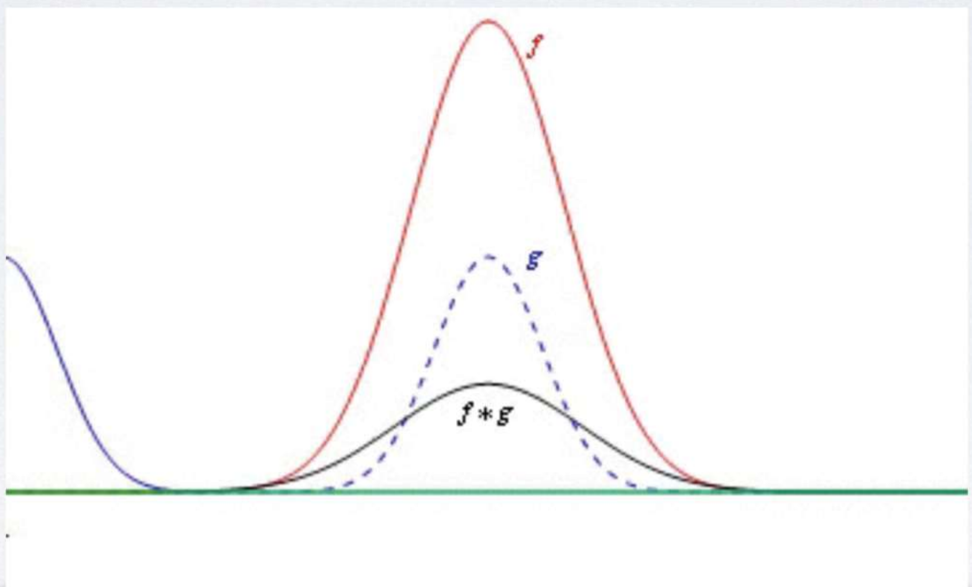
LIC Approach - 1D Convolution I

- Convolution defined as $(f * g)(x) := \int_{\mathbb{R}^n} f(\tau)g(x-\tau)d\tau$



LIC Approach - 1D Convolution II

- Convolution defined as $(f * g)(x) := \int_{\mathbb{R}^n} f(\tau) g(x - \tau) d\tau$



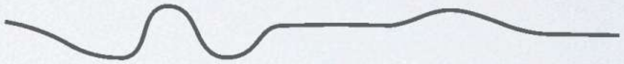
LIC Approach - 1D Convolution III

$k(x)$ convolution kernel

¼	½	¼

$f(x)$ original signal

4	2	0	8	0	4	4	4	6	4	2	2



$$(f * k)(x) = \int_{-L/2}^{L/2} f(\tau)k(x-\tau)d\tau$$

$(f * k)(x)$ smoothed signal

LIC Approach - 1D Convolution III

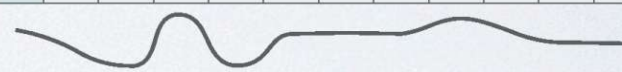
$k(x)$ convolution kernel

¼	½	¼

common section L

$f(x)$ original signal

		4	2	0	8	0	4	4	4	6	4	2	2



$(f * k)(x)$ smoothed signal

$$(f * k)(x) = \int_{-L/2}^{L/2} f(\tau)k(x-\tau)d\tau$$

LIC Approach - 1D Convolution III

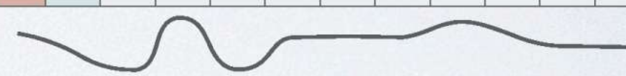
$k(x)$ convolution kernel

¼	½	¼

common section L

$f(x)$ original signal

	4	2	0	8	0	4	4	4	6	4	2	2



¼ · 0 + ½ · 4 + ¼ · 2

$(f * k)(x)$ smoothed signal

$$(f * k)(x) = \int_{-L/2}^{L/2} f(\tau)k(x-\tau)d\tau$$

LIC Approach - 1D Convolution III

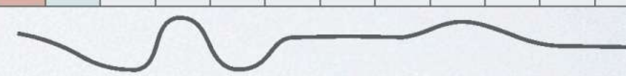
$k(x)$ convolution kernel

¼	½	¼

common section L

$f(x)$ original signal

	4	2	0	8	0	4	4	4	6	4	2	2



¼ · 0 + ½ · 4 + ¼ · 2

$(f * k)(x)$ smoothed signal

3											

$$(f * k)(x) = \int_{-L/2}^{L/2} f(\tau)k(x-\tau)d\tau$$

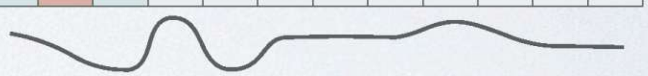
LIC Approach - 1D Convolution III

$k(x)$ convolution kernel

¼	½	¼

$f(x)$ original signal

4	2	0	8	0	4	4	4	6	4	2	2



¼ · 4 + ½ · 2 + ¼ · 0

$(f * k)(x)$ smoothed signal

3											

$$(f * k)(x) = \int_{-L/2}^{L/2} f(\tau)k(x-\tau)d\tau$$

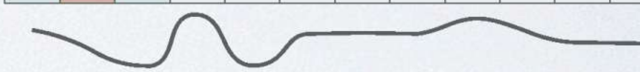
LIC Approach - 1D Convolution III

$k(x)$ convolution kernel

¼	½	¼

$f(x)$ original signal

4	2	0	8	0	4	4	4	6	4	2	2



¼ · 4 + ½ · 2 + ¼ · 0

$(f * k)(x)$ smoothed signal

3	2										

$$(f * k)(x) = \int_{-L/2}^{L/2} f(\tau)k(x-\tau)d\tau$$

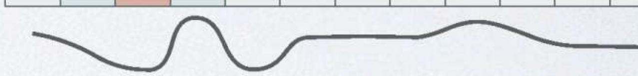
LIC Approach - 1D Convolution III

$k(x)$ convolution kernel

¼	½	¼

$f(x)$ original signal

4	2	0	8	0	4	4	4	6	4	2	2



¼ ▪ 2 + ½ ▪ 0 + ¼ ▪ 8

$(f * k)(x)$ smoothed signal

3	2										

$$(f * k)(x) = \int_{-L/2}^{L/2} f(\tau)k(x-\tau)d\tau$$

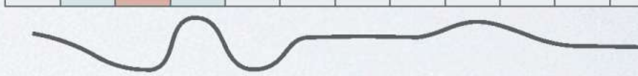
LIC Approach - 1D Convolution III

$k(x)$ convolution kernel

¼	½	¼

$f(x)$ original signal

4	2	0	8	0	4	4	4	6	4	2	2



$\frac{1}{4} \cdot 2 + \frac{1}{2} \cdot 0 + \frac{1}{4} \cdot 8$

$$(f * k)(x) = \int_{-L/2}^{L/2} f(\tau)k(x-\tau)d\tau$$

$(f * k)(x)$ smoothed signal

3	2	2½									

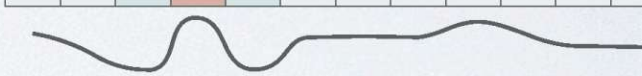
LIC Approach - 1D Convolution III

$k(x)$ convolution kernel

¼	½	¼

$f(x)$ original signal

4	2	0	8	0	4	4	4	6	4	2	2



¼ · 0 + ½ · 8 + ¼ · 0

$(f * k)(x)$ smoothed signal

3	2	2½									

$$(f * k)(x) = \int_{-L/2}^{L/2} f(\tau)k(x-\tau)d\tau$$

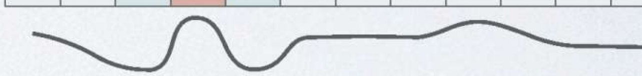
LIC Approach - 1D Convolution III

$k(x)$ convolution kernel

¼	½	¼

$f(x)$ original signal

4	2	0	8	0	4	4	4	6	4	2	2



$\frac{1}{4} \cdot 0 + \frac{1}{2} \cdot 8 + \frac{1}{4} \cdot 0$

$(f * k)(x)$ smoothed signal

3	2	2½	4								

$$(f * k)(x) = \int_{-L/2}^{L/2} f(\tau)k(x-\tau)d\tau$$

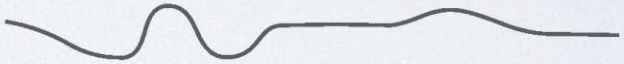
LIC Approach - 1D Convolution III

$k(x)$ convolution kernel

¼	½	¼

$f(x)$ original signal

4	2	0	8	0	4	4	4	6	4	2	2	

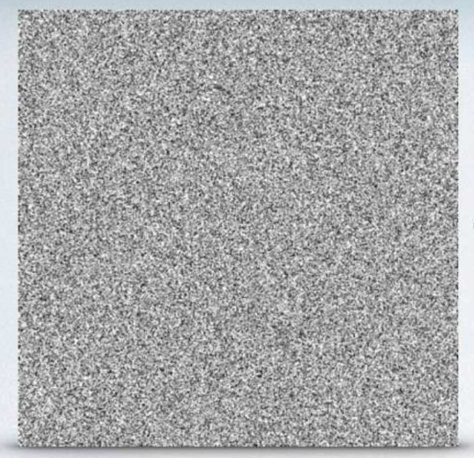


$$(f * k)(x) = \int_{-L/2}^{L/2} f(\tau)k(x-\tau)d\tau$$

$(f * k)(x)$ smoothed signal

3	2	2½	4	3	3	4	4½	5	4	2½	1½

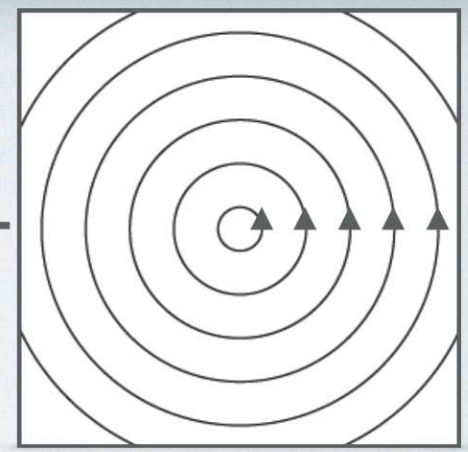
LIC Approach - 1D Convolution IV



Input Noise

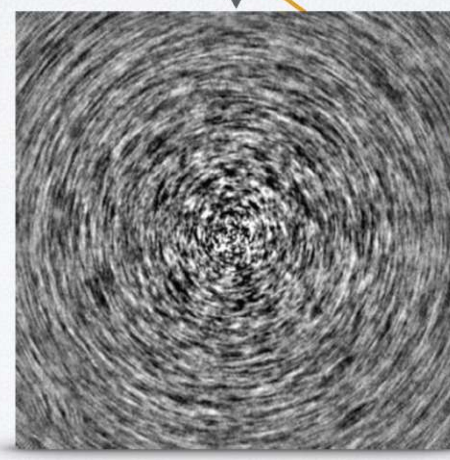
Convolution

$$\int T(\boldsymbol{x}(t+s))k(s)ds$$



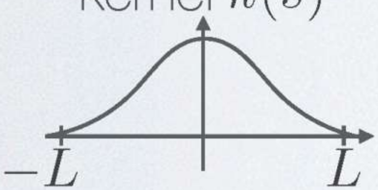
Vector Field

Streamline Tracing



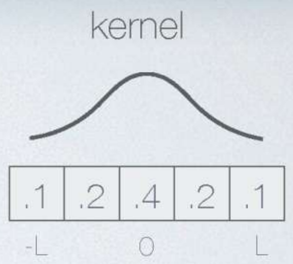
Final Image

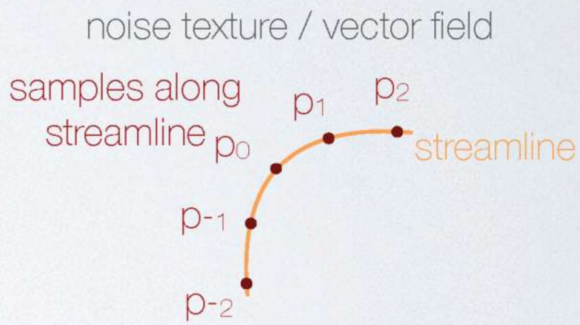
kernel $k(s)$



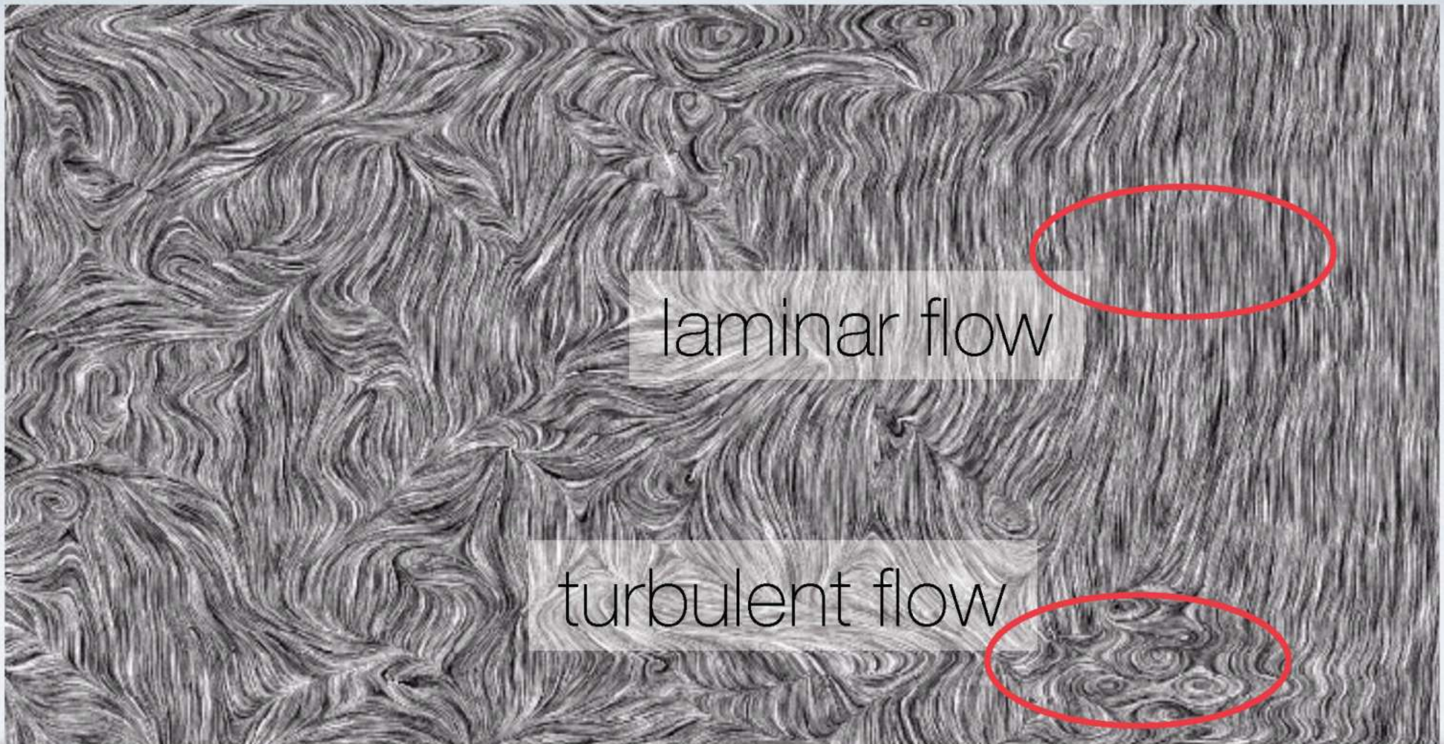
LIC - Algorithm

```
for each pixel //perfect fit for fragment shader

  t = texture( position, noise_texture );

  smoothed_value = kernel_value(center) * t;
  P+ = p- = position;

  for 1 to L // loop over kernel

    v+ = texture( p+, vector_texture );
    p+ = streamlineIntegration(p+, v+);
    smoothed_value +=
        kernel_value * texture( p+, noise_texture );

    v- = -texture( p-, vector_texture );
    p- = streamlineIntegration(p-, v-);
    smoothed_value +=
        kernel_value * texture( p-, noise_texture );
```





LIC - 2D Example



Linear Algebra Approach (1)

- Toeplitz matrix: constant diagonals

$$\mathbf{T} := \left(t_{ij}\right) \text{ with } t_{ij} := t_{i-j}$$

$$\mathbf{T}^{N\times N} := \begin{bmatrix} t_0 & t_{(-1)} & t_{(-2)} & \cdots & t_{(-(N-2))} & t_{(-(N-1))} \\ t_1 & t_0 & t_{(-1)} & \cdots & t_{(-(N-3))} & t_{(-(N-2))} \\ t_2 & t_1 & t_0 & \cdots & t_{(-(N-4))} & t_{(-(N-3))} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ t_{N-2} & t_{N-3} & t_{N-4} & \cdots & t_0 & t_{(-1)} \\ t_{N-1} & t_{N-2} & t_{N-3} & \cdots & t_1 & t_0 \end{bmatrix}$$

Linear Algebra Approach (2)

- Circulant matrix: special case of Toeplitz matrix

$$\mathbf{C} := (c_{ij}) \text{ where } c_{ij} := c_{(i-j)\, mod\, N}$$

$$\mathbf{C}^{N\times N} := \begin{bmatrix} c_0 & c_{N-1} & c_{N-2} & \cdots & c_2 & c_1 \\ c_1 & c_0 & c_{N-1} & \cdots & c_3 & c_2 \\ c_2 & c_1 & c_0 & \cdots & c_4 & c_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ c_{N-2} & c_{N-3} & c_{N-4} & \cdots & c_0 & c_{N-1} \\ c_{N-1} & c_{N-2} & c_{N-3} & \cdots & c_1 & c_0 \end{bmatrix}$$

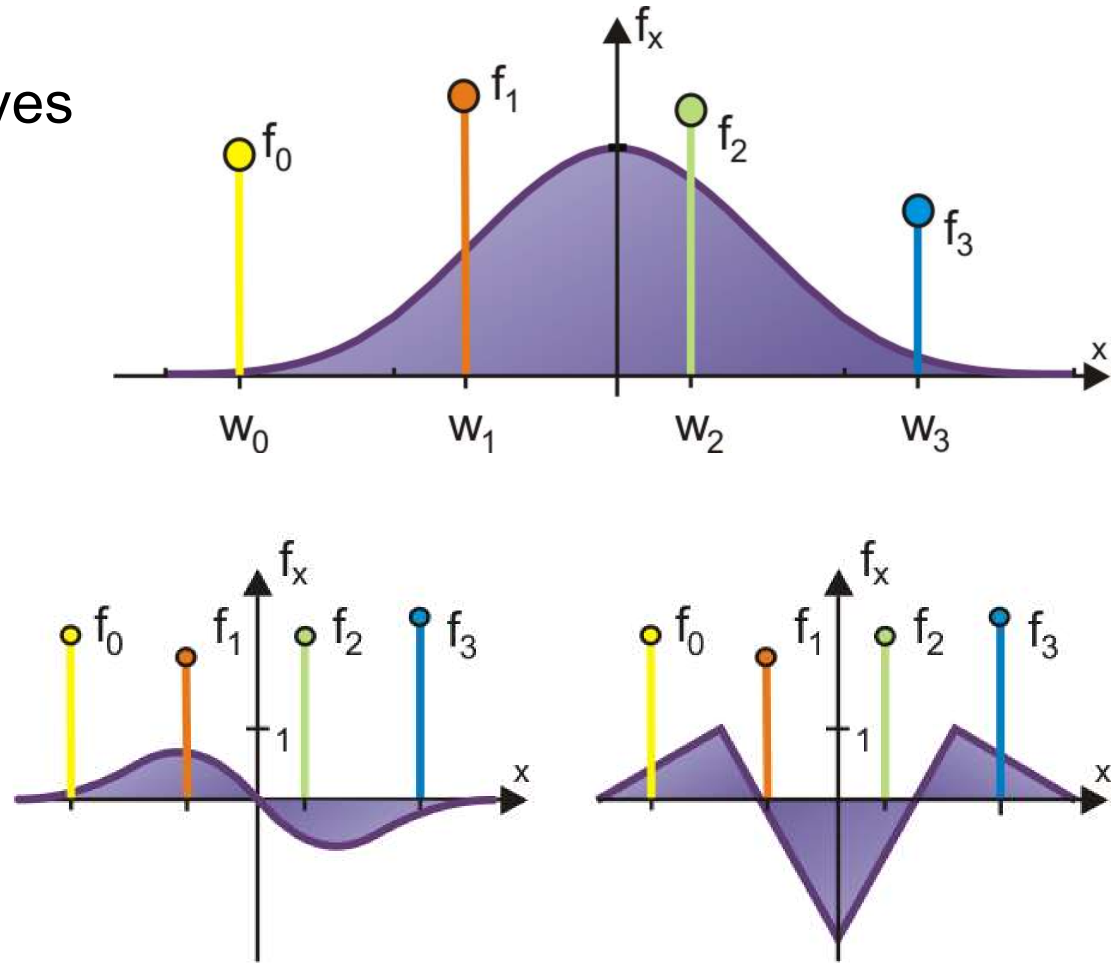
- Periodic convolution: multiply **C** with (periodic) signal in column vector
- The Fourier transform *diagonalizes* circulant matrices

Interlude:
Derivatives via Convolution

Convolve with Derivatives of Kernel

Example

- Cubic B-spline and derivatives
- Use 1D kernels and tensor product for tri-cubic
- Well-suited for curvature computation [Kindlmann et al., 2003]
- Expensive convolution?

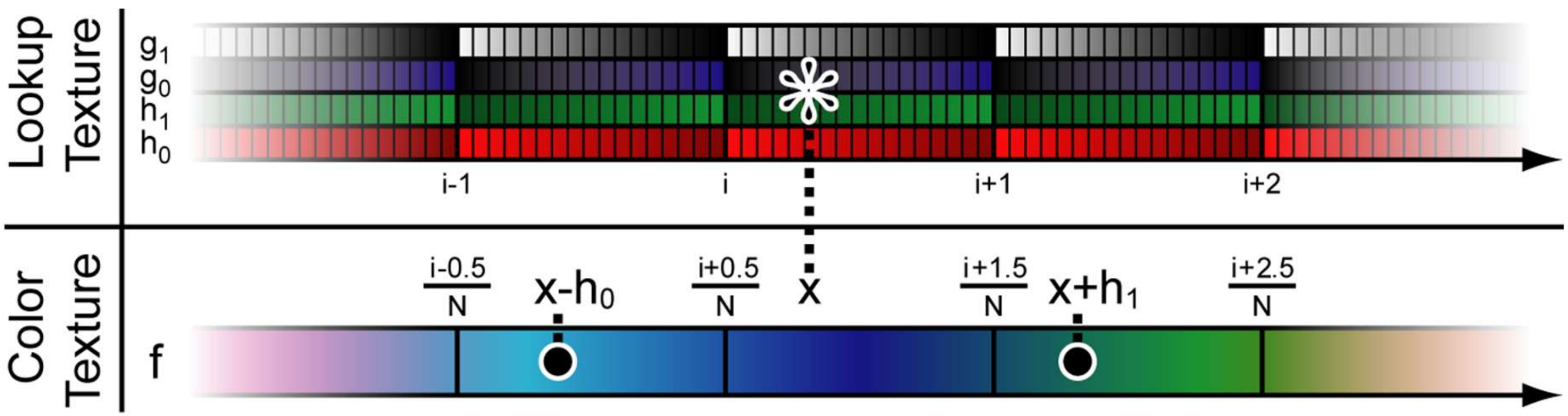


Fast Tri-Cubic Filtering on GPUs

Cubic: Need 64 neighbors; usually means 64 nearest-neighbor lookups

- But on GPUs 8 tri-linear lookups suffice for tri-cubic B-spline
- Kernels are transformed into 1D look-up textures (or simple equations)

[Sigg and Hadwiger, 2005] (GPU Gems 2)



$$g_0 \cdot f_{x-h_0} + g_1 \cdot f_{x+h_1}$$

- Newer: procedural kernel computation (see NVIDIA CUDA SDK)

Vector Fields, Vector Calculus, and Dynamical Systems

Fluid Simulation: Navier Stokes Equations

Incompressible (divergence-free) Navier Stokes equations

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \cdot \nabla)\mathbf{u} - \frac{1}{\rho}\nabla p + \nu \nabla^2 \mathbf{u} + \mathbf{F},$$

$$\nabla \cdot \mathbf{u} = 0,$$

Components:

- Self-advection of velocity (i.e., advection of velocity according to velocity)
- Pressure gradient (force due to pressure differences)
- Diffusion of velocity due to viscosity (for viscous fluids, i.e., not inviscid)
- Application of (arbitrary) external forces, e.g., gravity, user input, etc.

Some Vector Calculus (1)

Gradient (scalar field → vector field)

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

- Direction of steepest ascent; magnitude = rate
- *Conservative* vector field: gradient of some scalar (potential) function

Divergence (vector field → scalar field)

$$\nabla \cdot \mathbf{u} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}$$

- Volume density of outward flux:
  "exit rate: source? sink?"
- *Incompressible/solenoidal/divergence-free vector field*: div u = 0
  can express as curl (next slide) of some vector (potential) function

Laplacian (scalar field → scalar field)

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

- Divergence of gradient
- Measure for difference between point and its neighborhood

Some Vector Calculus (2)

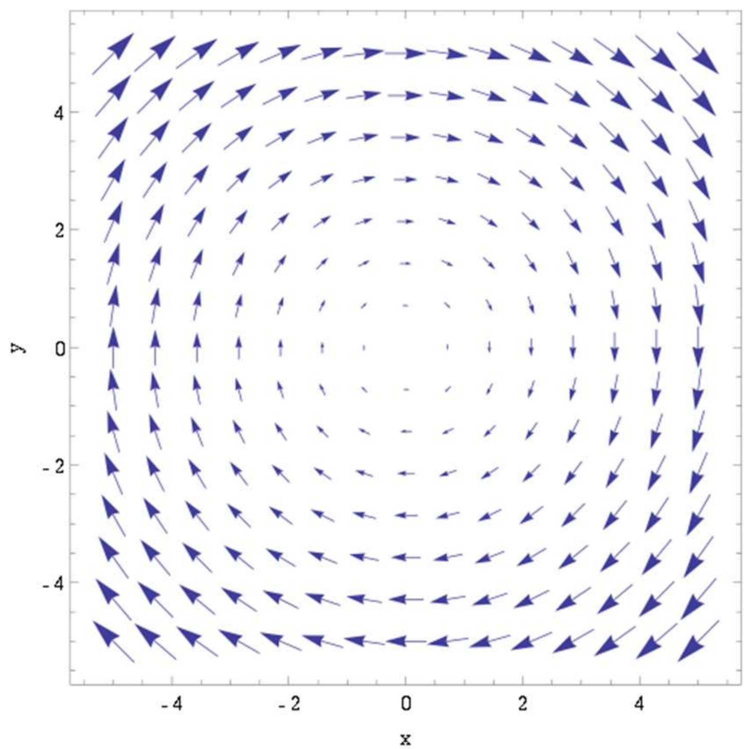
Curl (vector field → vector field)

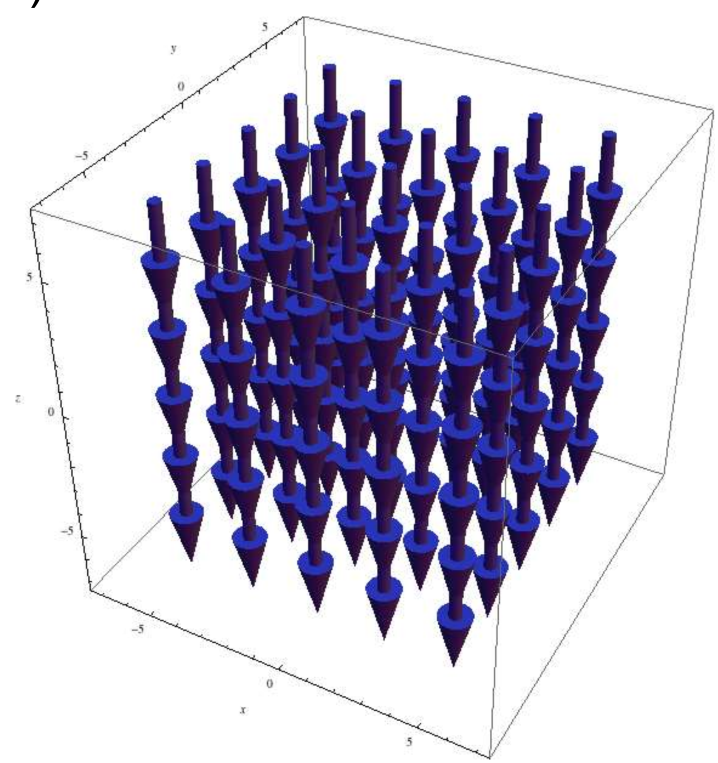
- Circulation density at a point (*vorticity*)
- If curl vanishes everywhere: irrotational/curl-free field
- Every conservative (path-independent) field is irrotational (and vice versa if domain is simply connected)

$$\nabla \times \mathbf{v} = \begin{pmatrix} w_y - v_z \\ u_z - w_x \\ v_x - u_y \end{pmatrix}$$

these are partial derivatives!

Example:
curl = const
everywhere





Some Vector Calculus (3)

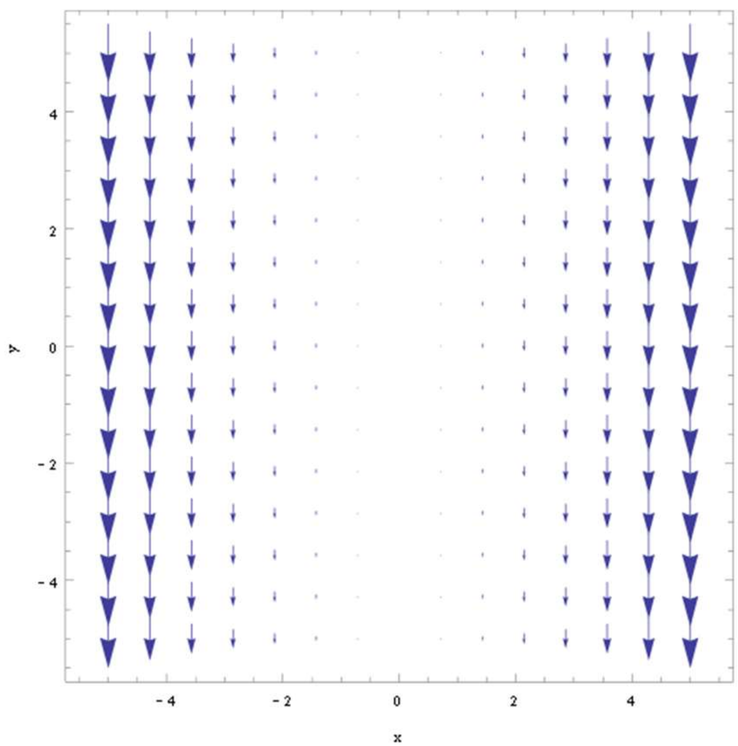
Curl (vector field → vector field)

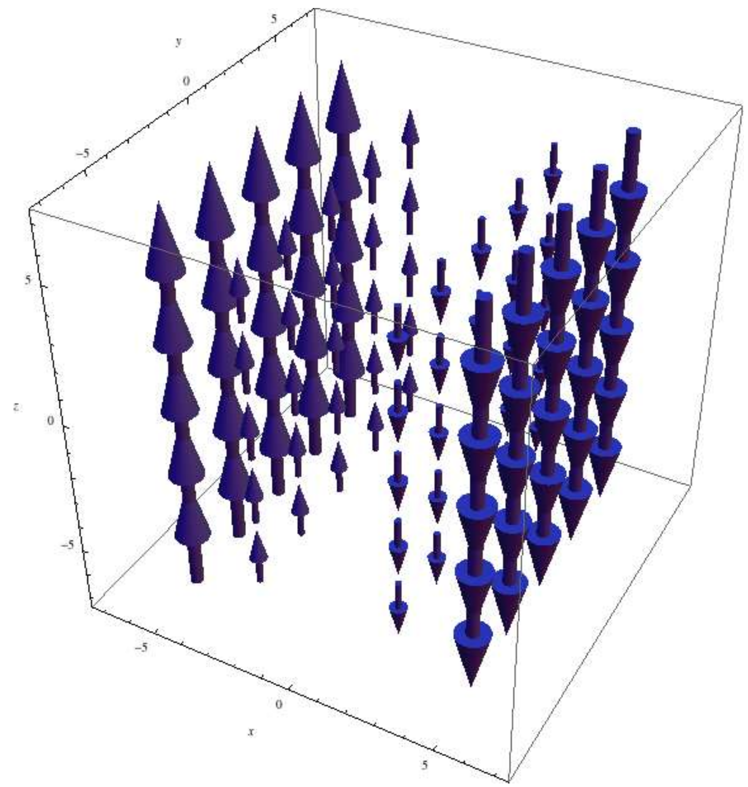
- Circulation density at a point (*vorticity*)
- If curl vanishes everywhere: irrotational/curl-free field
- Every conservative (path-independent) field is irrotational (and vice versa if domain is simply connected)

$$\nabla \times \mathbf{v} = \begin{pmatrix} w_y - v_z \\ u_z - w_x \\ v_x - u_y \end{pmatrix}$$

these are partial derivatives!

Example: curl not always “obviously rotational”





Some Vector Calculus (4)

Curl (vector field → vector field)

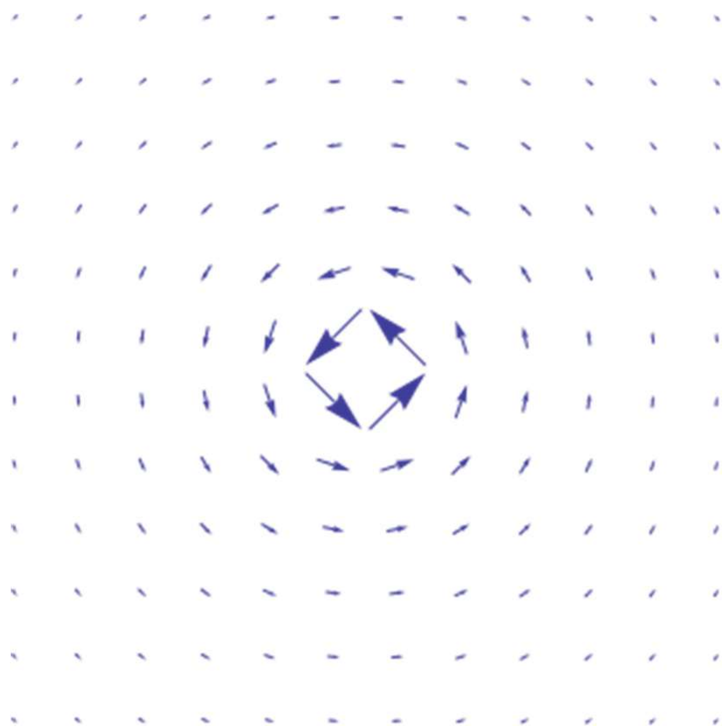
- Circulation density at a point (*vorticity*)
- If curl vanishes everywhere: irrotational/curl-free field
- Every conservative (path-independent) field is irrotational (and vice versa if domain is simply connected)

$$\nabla \times \mathbf{v} = \begin{pmatrix} w_y - v_z \\ u_z - w_x \\ v_x - u_y \end{pmatrix}$$

these are partial derivatives!

Example:
non-obvious
curl-free field

[this domain is **not** simply connected! it is the "punctured plane", i.e., the point (0,0) is not in the domain]



$$\mathbf{v}(x, y, z) = \frac{(-y, x, 0)}{x^2 + y^2}$$

not defined at (x,y) = (0,0)

$$v_x = u_y \qquad \nabla \times \mathbf{v} = 0$$

velocity gradient $\nabla \mathbf{v}$ is symmetric (see later)

Some Vector Calculus (5)

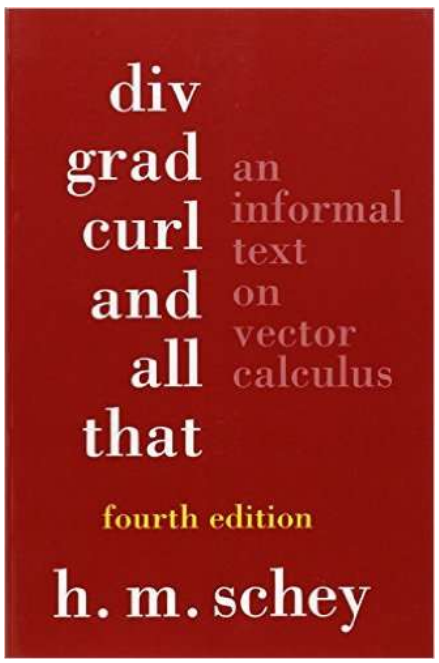
Curl (vector field → vector field)

- Circulation density at a point (*vorticity*)
- If curl vanishes everywhere: irrotational/curl-free field
- Every conservative (path-independent) field is irrotational (and vice versa if domain is simply connected)

$$\nabla \times \mathbf{v} = \begin{pmatrix} w_y - v_z \\ u_z - w_x \\ v_x - u_y \end{pmatrix}$$

these are partial derivatives!

Book:



Interactive tutorial on curl:
`http://mathinsight.org/curl_idea`

Fundamental theorem of vector calculus:
Helmholtz decomposition: Any vector field can be expressed as the sum of a solenoidal (*divergence-free*) vector field and an irrotational (*curl-free*) vector field (Helmholtz-Hodge: plus *harmonic* vector field)

Vector Fields and Dynamical Systems (1)

Velocity gradient tensor, (vector field → tensor field)

- Gradient of vector field: how does the vector field change?
- In Cartesian coordinates: *spatial partial derivatives (Jacobian matrix)*

$$\nabla \mathbf{v}(x,y,z) = \begin{pmatrix} u_x & u_y & u_z \\ v_x & v_y & v_z \\ w_x & w_y & w_z \end{pmatrix}$$

these are
partial derivatives!

- Can be decomposed into *symmetric* part + *anti-symmetric* part

$\nabla \mathbf{v} = \mathbf{D} + \mathbf{S}$ *velocity gradient tensor*

sym.: $\mathbf{D} = \frac{1}{2}\left(\nabla \mathbf{v} + (\nabla \mathbf{v})^{\mathrm{T}} \right)$ deform.: *rate-of-strain tensor*

skew-sym.: $\mathbf{S} = \frac{1}{2}\left(\nabla \mathbf{v} - (\nabla \mathbf{v})^{\mathrm{T}} \right)$ rotation: *vorticity/spin tensor*

Vector Fields and Dynamical Systems (2)

Vorticity/spin/angular velocity tensor

- Antisymmetric part of velocity gradient tensor
- Corresponds to vorticity/curl/angular velocity (beware of factor ½)

these are partial derivatives!

$$\mathbf{S} = \tfrac{1}{2} \left(\nabla \mathbf{v} - (\nabla \mathbf{v})^{\mathrm{T}} \right)$$

$$\mathbf{S} = \tfrac{1}{2} \begin{pmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{pmatrix} \qquad \omega = \begin{pmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{pmatrix} = \nabla \times \mathbf{v} = \begin{pmatrix} w_y - v_z \\ u_z - w_x \\ v_x - u_y \end{pmatrix}$$

$\mathbf{S}$ acts on vector like cross product with ω : $\mathbf{S}\, \bullet \; = \tfrac{1}{2}\, \omega \times$

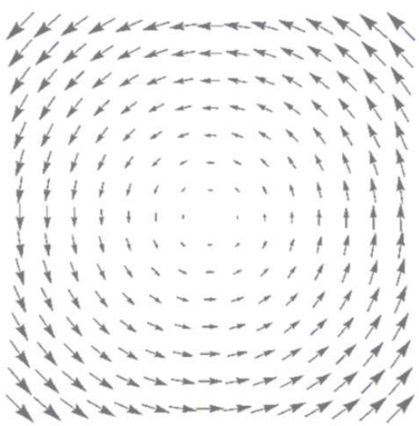
$$\mathbf{v}^{(r)} = \mathbf{S} \cdot d\mathbf{r} = \frac{1}{2} \left[\nabla \mathbf{v} - (\nabla \mathbf{v})^T \right] \cdot d\mathbf{r} = \frac{1}{2} \boldsymbol{\omega} \times d\mathbf{r}$$

Angular Velocity of Rigid Body Rotation

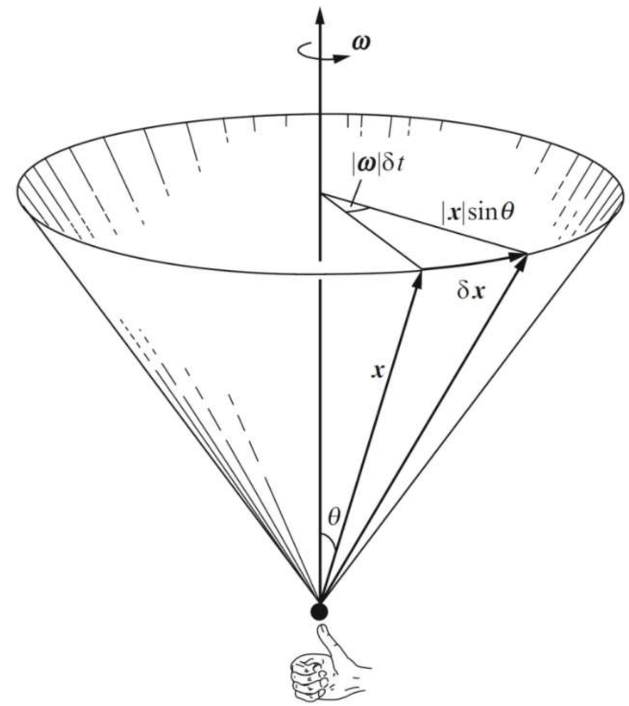
Rate of rotation

- Scalar ω: angular displacement per unit time (rad s^{-1})
  - Angle Θ at time t is $\Theta(t) = \omega t$; $\omega = 2\pi f$ where f is the frequency ($f = 1/T$; s^{-1})
- Vector $\boldsymbol{\omega}$: axis of rotation; magnitude is angular speed (if $\boldsymbol{\omega}$ is curl: speed x2)
  - Beware of different conventions that differ by a factor of ½ **!**

Cross product of $\frac{1}{2}\boldsymbol{\omega}$ with vector to center of rotation ($\mathbf{r}$) gives linear velocity vector $\mathbf{v}$ (tangent)



$$\mathbf{v}^{(r)} = \frac{1}{2}\,\boldsymbol{\omega} \times d\mathbf{r}$$



Velocity Gradient Tensor and Components (1)

Velocity gradient tensor

(here: in Cartesian coordinates)

$$\nabla \mathbf{v} = \begin{bmatrix} \frac{\partial}{\partial x} v^x & \frac{\partial}{\partial y} v^x & \frac{\partial}{\partial z} v^x \\ \frac{\partial}{\partial x} v^y & \frac{\partial}{\partial y} v^y & \frac{\partial}{\partial z} v^y \\ \frac{\partial}{\partial x} v^z & \frac{\partial}{\partial y} v^z & \frac{\partial}{\partial z} v^z \end{bmatrix}$$

these are the same partial derivatives as before!

$$\nabla \mathbf{v} = \frac{1}{2}\left(\nabla \mathbf{v} + (\nabla \mathbf{v})^T\right) + \frac{1}{2}\left(\nabla \mathbf{v} - (\nabla \mathbf{v})^T\right)$$

Velocity Gradient Tensor and Components (2)

Rate-of-strain (rate-of-deformation) tensor

(symmetric part; here: in Cartesian coordinates)

$$\mathbf{D} = \frac{1}{2}\begin{bmatrix} 2\frac{\partial}{\partial x}v^x & \frac{\partial}{\partial y}v^x + \frac{\partial}{\partial x}v^y & \frac{\partial}{\partial z}v^x + \frac{\partial}{\partial x}v^z \\ \frac{\partial}{\partial x}v^y + \frac{\partial}{\partial y}v^x & 2\frac{\partial}{\partial y}v^y & \frac{\partial}{\partial z}v^y + \frac{\partial}{\partial y}v^z \\ \frac{\partial}{\partial x}v^z + \frac{\partial}{\partial z}v^x & \frac{\partial}{\partial y}v^z + \frac{\partial}{\partial z}v^y & 2\frac{\partial}{\partial z}v^z \end{bmatrix}$$

$$tr(\mathbf{D}) = \nabla \cdot \mathbf{v}$$

Velocity Gradient Tensor and Components (3)

Vorticity tensor (spin tensor)

(skew-symmetric part; here: in Cartesian coordinates)

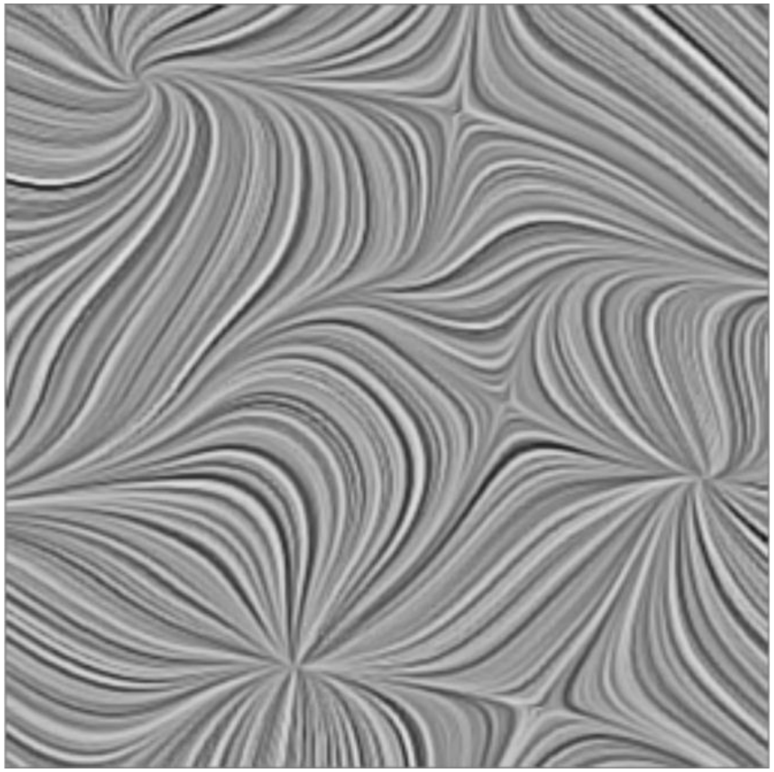
$$\mathbf{S} = \frac{1}{2}\begin{bmatrix} 0 & \frac{\partial}{\partial y}v^x - \frac{\partial}{\partial x}v^y & \frac{\partial}{\partial z}v^x - \frac{\partial}{\partial x}v^z \\ \frac{\partial}{\partial x}v^y - \frac{\partial}{\partial y}v^x & 0 & \frac{\partial}{\partial z}v^y - \frac{\partial}{\partial y}v^z \\ \frac{\partial}{\partial x}v^z - \frac{\partial}{\partial z}v^x & \frac{\partial}{\partial y}v^z - \frac{\partial}{\partial z}v^y & 0 \end{bmatrix}$$

$$\mathbf{S} = \frac{1}{2}\begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \qquad \boldsymbol{\omega} \equiv \nabla \times \mathbf{v}$$

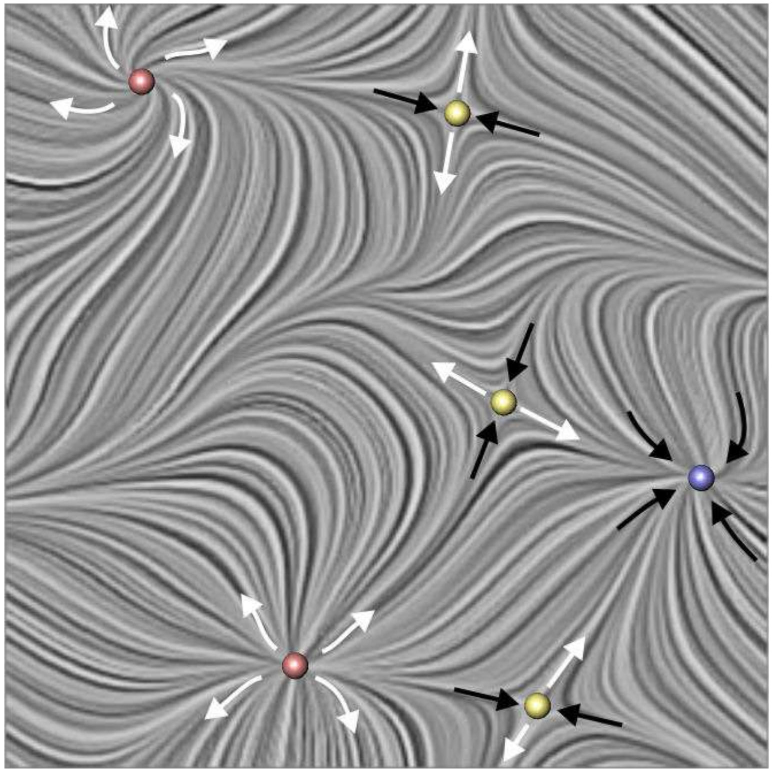
Critical Point Analysis

Critical Points (Steady Flow!)

Classify critical points according to the *eigenvalues* of the velocity gradient tensor at the critical point



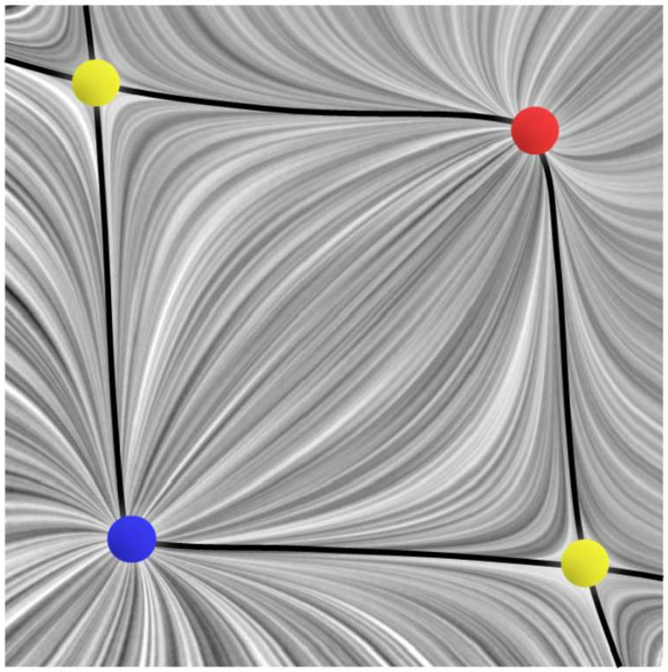
stream lines (LIC)

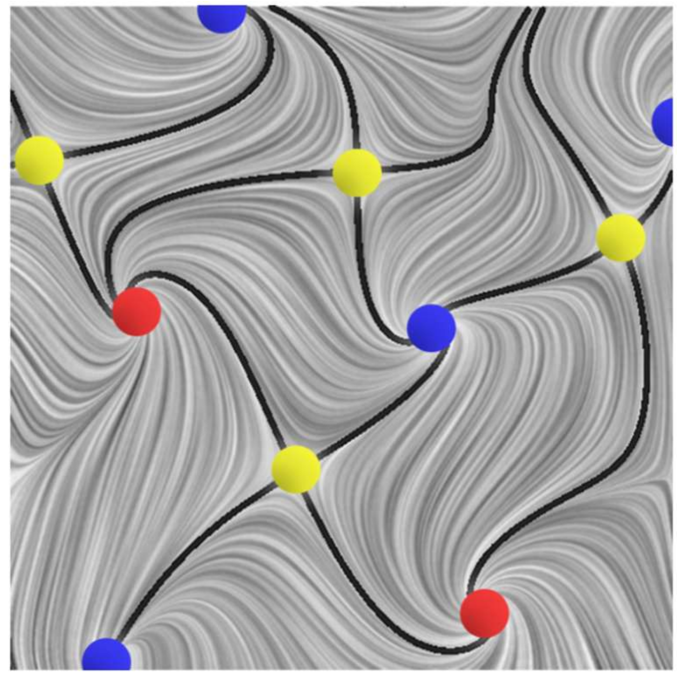


critical points (**v** = 0)

Vector Field Topology: Topological Skeleton

Connect critical points by *separatrices*





Sources (red), sinks (blue), saddles (yellow)

(Non-Linear) Dynamical Systems

Start with system of linear ODEs (with constant coefficients)

- Non-linear systems can be linearized around critical points
- Use linearization for characterization

$\dot{\mathbf{x}} = A\mathbf{x}$ $\quad$ A is an $n \times n$ matrix $\quad \longrightarrow \quad \boxed{\begin{aligned} \mathbf{v} &= A\,\mathbf{x}, \\ \nabla \mathbf{v} &= A. \end{aligned}}$

$$\dot{\mathbf{x}} = \frac{d\mathbf{x}}{dt} = \begin{bmatrix} \frac{dx_1}{dt} \\ \vdots \\ \frac{dx_n}{dt} \end{bmatrix}$$

$$\mathbf{x}(0) = \mathbf{x}_0$$

solution: $\mathbf{x}(t) = e^{At}\mathbf{x}_0$

characterize behavior
through eigenvalues of A

A Few Facts about Eigenvalues and –vectors

The matrix $\begin{bmatrix} c & -s \\ s & c \end{bmatrix}$ has eigenvalues $\lambda_1 = c + s\mathbf{i} \quad \lambda_2 = c - s\mathbf{i}$

with eigenvectors $u_1 = \begin{bmatrix} 1 \\ -\mathbf{i} \end{bmatrix} \quad u_2 = \begin{bmatrix} 1 \\ +\mathbf{i} \end{bmatrix}$ (if s non-zero)

If c = 0, this is a skew-symmetric matrix: pure imaginary eigenvalues

Skew-symmetric matrices: “infinitesimal rotations” (infinitesimal generators of rot.)

For $c = \cos\theta$ and $s = \sin\theta$: 2x2 rotation matrix with

$$\begin{aligned} \lambda_1 &= e^{\mathbf{i}\theta} &= \cos\theta + \mathbf{i}\sin\theta \\ \lambda_2 &= e^{-\mathbf{i}\theta} &= \cos\theta - \mathbf{i}\sin\theta \end{aligned}$$

Eigenvalues

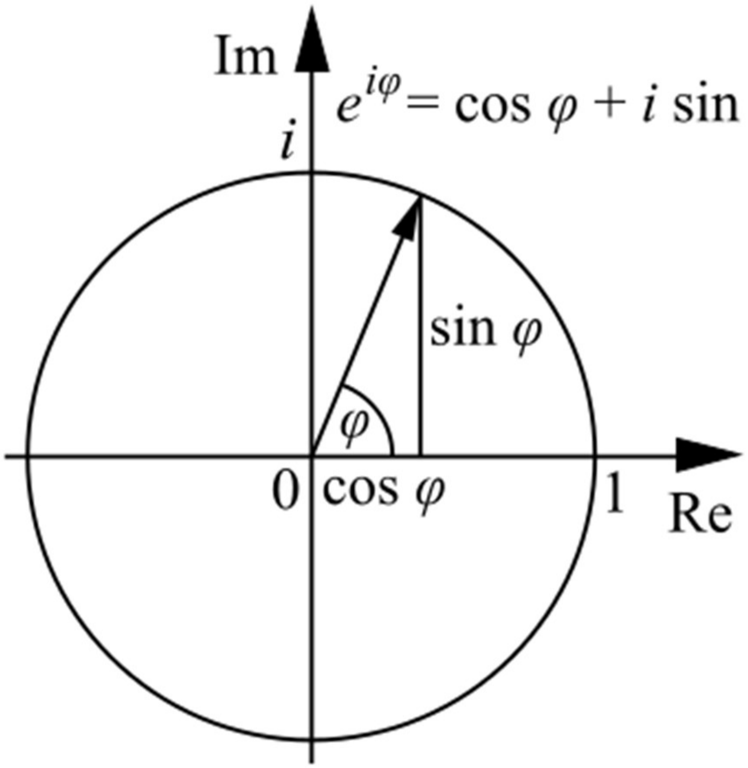
- Symmetric matrix: all eigenvalues are *real*
- Skew-symmetric matrix: all eigenvalues are *pure imaginary*

Euler’s Formula

Can be derived from the infinite power series for exp(), cos(), sin()

$$e^{ix} = \cos x + i \sin x$$

$$e^{i\pi} + 1 = 0$$



Matrix Exponentials

Defined via same power series as usual exponential

$$\exp(X) = e^X := \sum_{k=0}^{\infty} \frac{X^k}{k!} = I + X + \frac{X^2}{2!} + \frac{X^3}{3!} + \dots$$

Easy if X is diagonalizable

$$X = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} \qquad \exp(tX) = \begin{pmatrix} e^{\lambda_1 t} & 0 \\ 0 & e^{\lambda_2 t} \end{pmatrix}$$

Exponentials of anti-symmetric matrices are rotation matrices

$$X = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \qquad \exp(tX) = \begin{pmatrix} \cos t & -\sin t \\ \sin t & \cos t \end{pmatrix}$$

Matrix Exponentials

Defined via same power series as usual exponential

$$\exp(X) = e^X := \sum_{k=0}^{\infty} \frac{X^k}{k!} = I + X + \frac{X^2}{2!} + \frac{X^3}{3!} + \dots$$

Easy if X is diagonalizable

$$X = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} \qquad \exp(tX) = \begin{pmatrix} e^{\lambda_1 t} & 0 \\ 0 & e^{\lambda_2 t} \end{pmatrix}$$

Complex eigenvalues lead to rotation

$$X = \begin{pmatrix} a & -\omega \\ \omega & a \end{pmatrix} \qquad \exp(tX) = e^{at} \begin{pmatrix} \cos \omega t & -\sin \omega t \\ \sin \omega t & \cos \omega t \end{pmatrix}$$

$$\lambda_{1,2} = a \pm \mathbf{i}\omega$$

Classification of Critical Points (1)

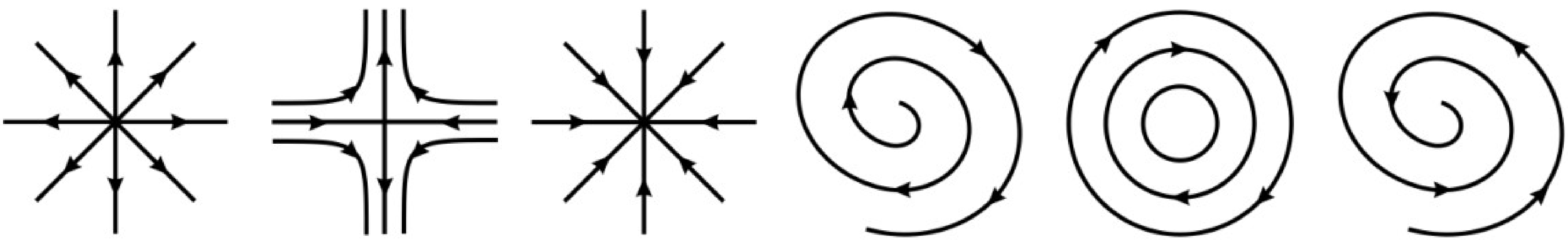
(Isolated) critical point (equilibrium point)

- Velocity vanishes (all components zero)

$$\mathbf{v}(\mathbf{x}_\mathrm{c}) = \mathbf{0} \quad \text{with} \quad \mathbf{v}(\mathbf{x}_\mathrm{c} \pm \boldsymbol{\epsilon}) \neq \mathbf{0} \qquad\qquad \det(\nabla \mathbf{v}(\mathbf{x}_\mathrm{c})) \neq 0$$

Characterize using velocity gradient $\nabla \mathbf{v}$ at critical point $\mathbf{x}_\mathrm{c}$

- Look at eigenvalues (and eigenvectors) of $\nabla \mathbf{v}$



Repelling node: $R_1, R_2 > 0$, $I_1 = I_2 = 0$

Saddle point: $R_1 < 0, R_2 > 0$, $I_1 = I_2 = 0$

Attracting node: $R_1, R_2 < 0$, $I_1 = I_2 = 0$

Repelling focus: $R_1 = R_2 > 0$, $I_1 = -I_2 \neq 0$

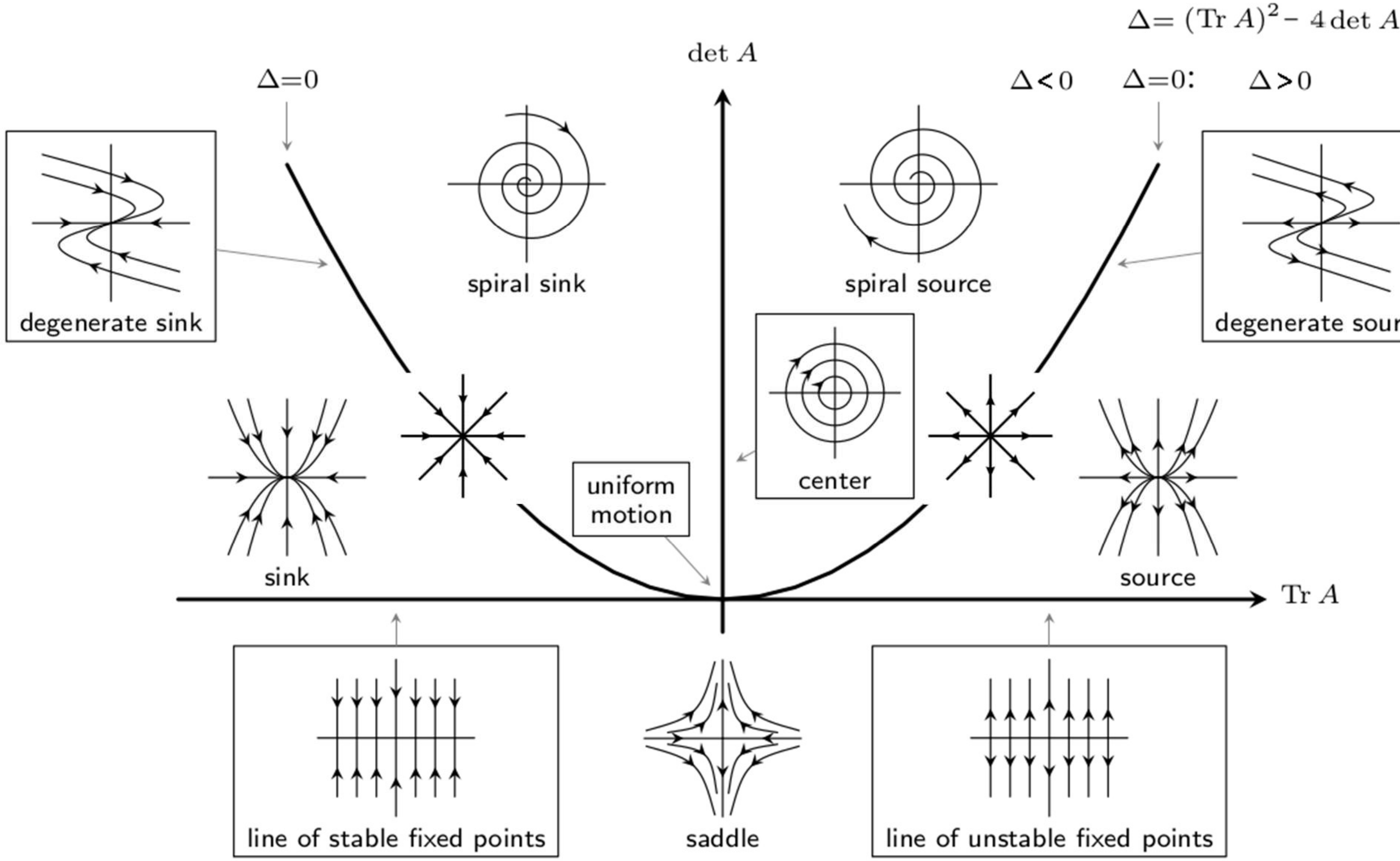
Center: $R_1 = R_2 = 0$, $I_1 = -I_2 \neq 0$

Attracting focus: $R_1 = R_2 < 0$, $I_1 = -I_2 \neq 0$

the first three phase portraits are special cases, see later slides!

Classification of Critical Points (2)

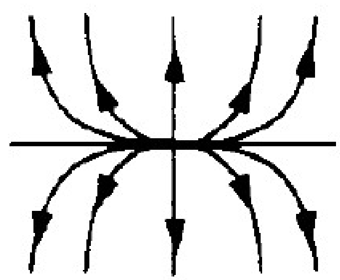
Poincaré Diagram: Classification of Phase Portaits in the $(\det A, \operatorname{Tr} A)$-plane



A Few Details (1)

Repelling/attracting nodes

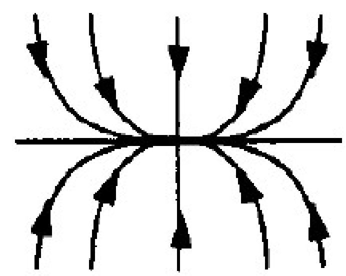
- Do not necessarily imply that streamlines are straight lines (do not confuse with the linear system of ODEs!)
- They are only straight lines when both eigenvalues are real and have the same sign, *and are also equal* (as in the phase portraits before)
- If they are not equal:



Repelling Node

R1, R2 > 0

I1, I2 = 0



Attracting Node

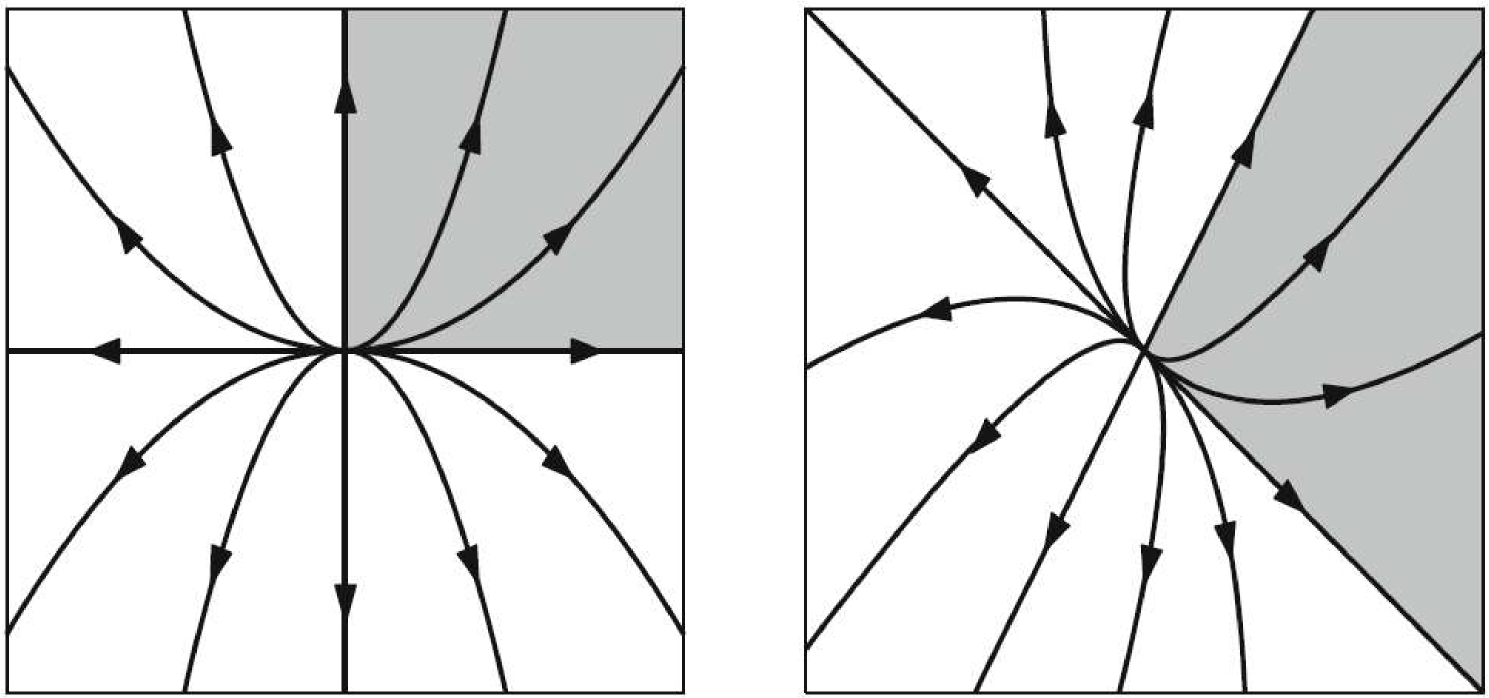
R1, R2 < 0

I1, I2 = 0

A Few Details (2)

What about skew axes?

- Both of the systems below have eigenvalues 3 and 6
- Jordan normal form (Jordan canonical form) gives details



$$\begin{bmatrix} 3 & 0 \\ 0 & 6 \end{bmatrix} \qquad \begin{bmatrix} 4 & 1 \\ 2 & 5 \end{bmatrix}$$

Jordan Normal Form (2x2 Matrix)

For every real 2x2 matrix A there is an invertible P such that

$P^{-1}AP$ is one of the following Jordan matrices (all entries are real):

$$J_1 = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \qquad J_2 = \begin{bmatrix} \lambda & 0 \\ 1 & \lambda \end{bmatrix}$$ (defective matrix)

$$J_3 = \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} \qquad J_4 = \begin{bmatrix} a & -b \\ b & a \end{bmatrix}$$

Each of these has its corresponding rule for constructing P

- Example on prev. slide (the two eigenvectors are not orthogonal):

$$P = \frac{1}{3}\begin{bmatrix} 1 & 1 \\ -1 & 2 \end{bmatrix} \qquad \frac{1}{3}\begin{bmatrix} 2 & -1 \\ 1 & 1 \end{bmatrix}\begin{bmatrix} 4 & 1 \\ 2 & 5 \end{bmatrix}\begin{bmatrix} 1 & 1 \\ -1 & 2 \end{bmatrix} = \begin{bmatrix} 3 & 0 \\ 0 & 6 \end{bmatrix}$$

See also *algebraic* and *geometric multiplicity* of eigenvalues

Jordan Normal Form (2x2 Matrix)

For every real 2x2 matrix A there is an invertible P such that

$P^{-1}AP$ is one of the following Jordan matrices (all entries are real):

$$J_1 = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \qquad J_2 = \begin{bmatrix} \lambda & 0 \\ 1 & \lambda \end{bmatrix}$$ (defective matrix)

same eigenvalues,
trace, determinant!

$$J_3 = \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} \qquad J_4 = \begin{bmatrix} a & -b \\ b & a \end{bmatrix}$$

Each of these has its corresponding rule for constructing P

- Example on prev. slide (the two eigenvectors are not orthogonal):

$$P = \frac{1}{3}\begin{bmatrix} 1 & 1 \\ -1 & 2 \end{bmatrix} \qquad \frac{1}{3}\begin{bmatrix} 2 & -1 \\ 1 & 1 \end{bmatrix}\begin{bmatrix} 4 & 1 \\ 2 & 5 \end{bmatrix}\begin{bmatrix} 1 & 1 \\ -1 & 2 \end{bmatrix} = \begin{bmatrix} 3 & 0 \\ 0 & 6 \end{bmatrix}$$

See also *algebraic* and *geometric multiplicity* of eigenvalues

Another Example

$P^{-1}AP$ has form J_1

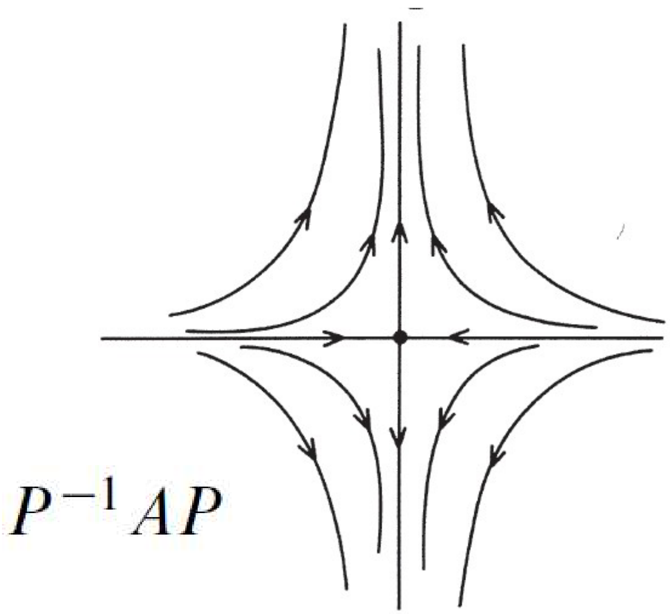
$$A = \begin{bmatrix} -1 & -3 \\ 0 & 2 \end{bmatrix}$$

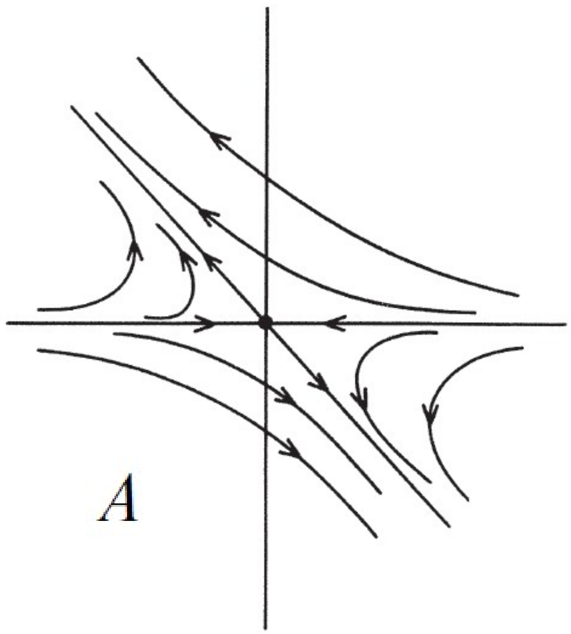
Eigenvalues:

$\lambda_1 = -1$

$\lambda_2 = 2$

$$P = \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix} \qquad P^{-1} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$$

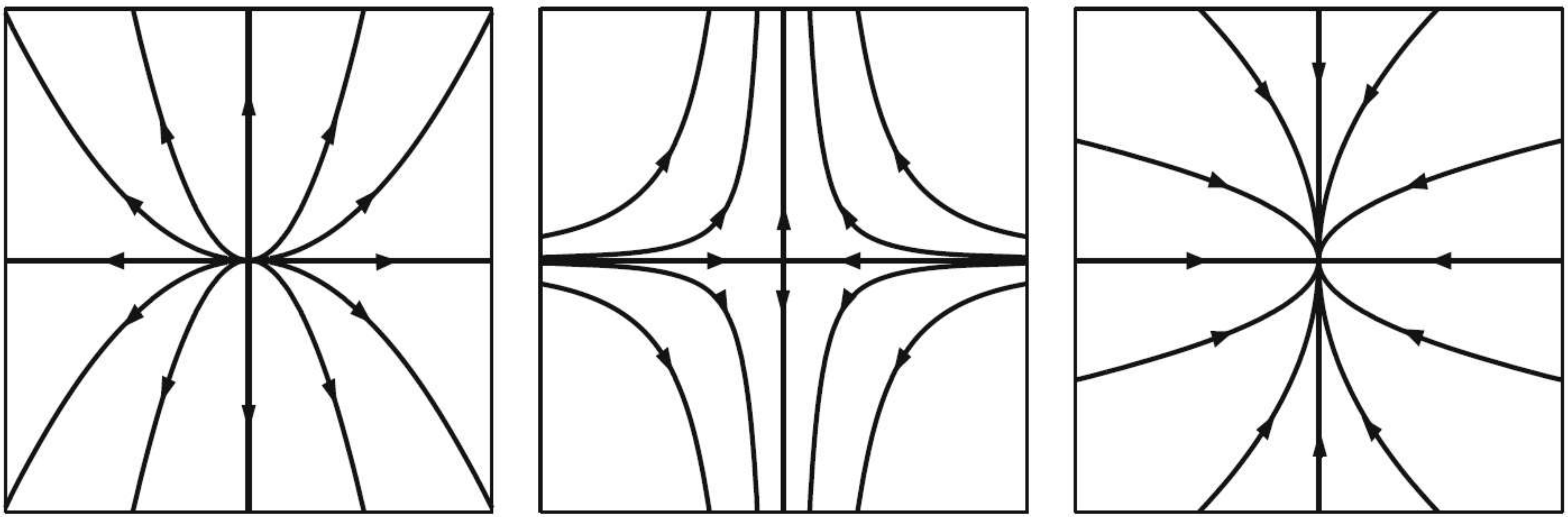




Jordan Form Characterization (1)

Phase portraits corresponding to Jordan matrix

$$J_1 = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$



$0 < \lambda_1 < \lambda_2$
unstable node

$\lambda_1 < 0 < \lambda_2$
saddle

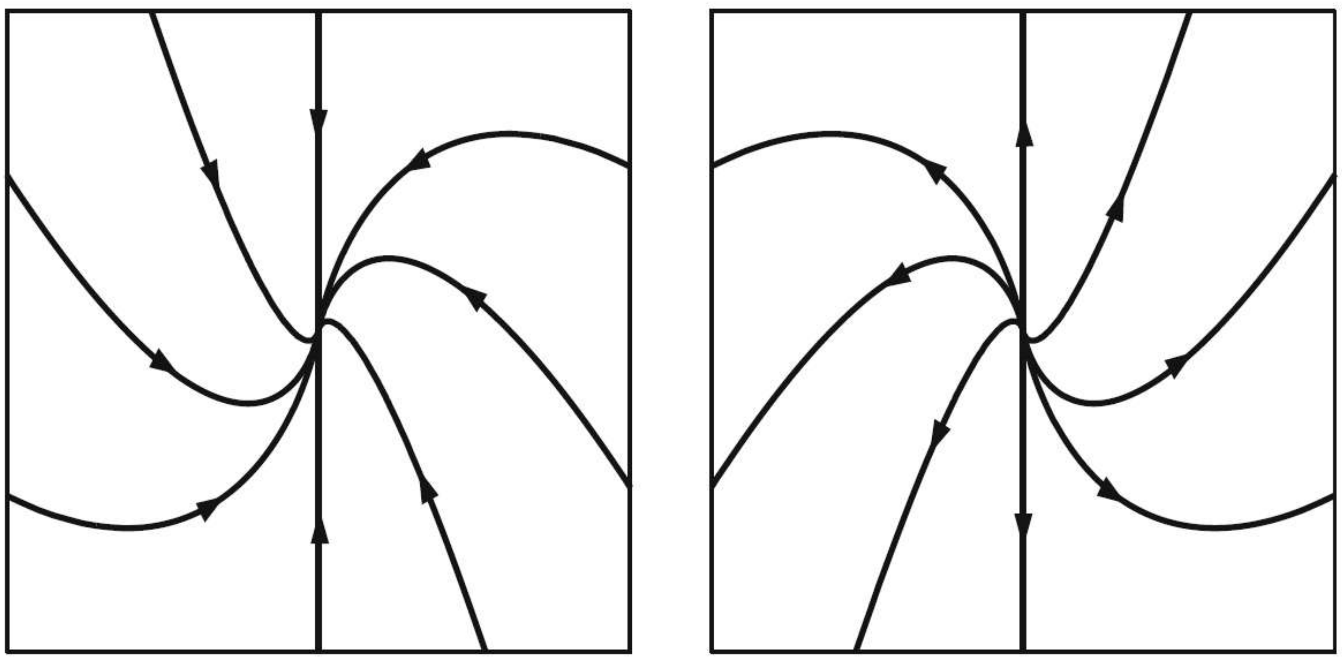
$\lambda_1 < \lambda_2 < 0$
stable node

Jordan Form Characterization (2)

Phase portraits corresponding to Jordan matrix

(matrix is defective: eigenspaces collapse, geometric multiplicity less than algebraic multiplicity)

$$J_2 = \begin{bmatrix} \lambda & 0 \\ 1 & \lambda \end{bmatrix}$$



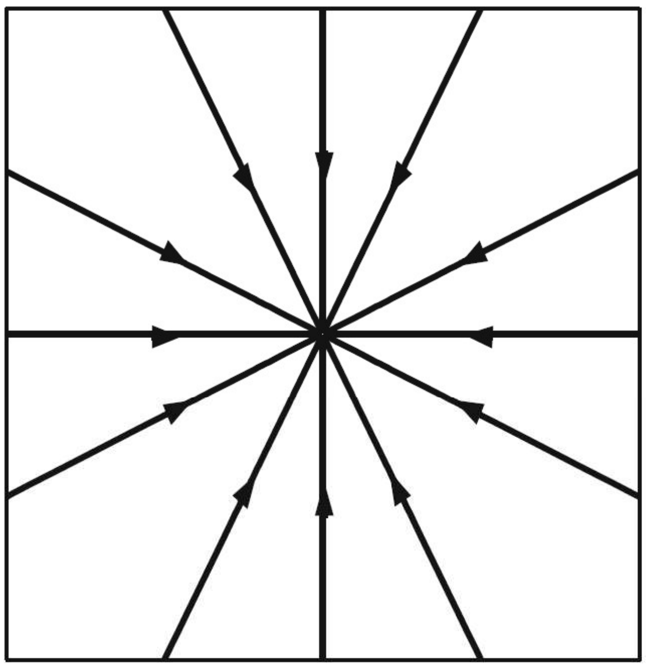
$\lambda < 0$
stable improper node

$\lambda > 0$
unstable improper node

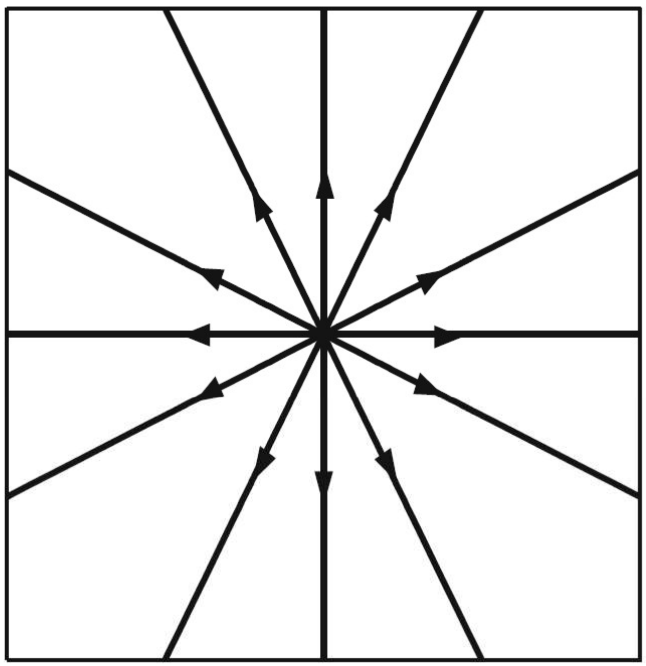
Jordan Form Characterization (3)

Phase portraits corresponding to Jordan matrix

$$J_3 = \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix}$$



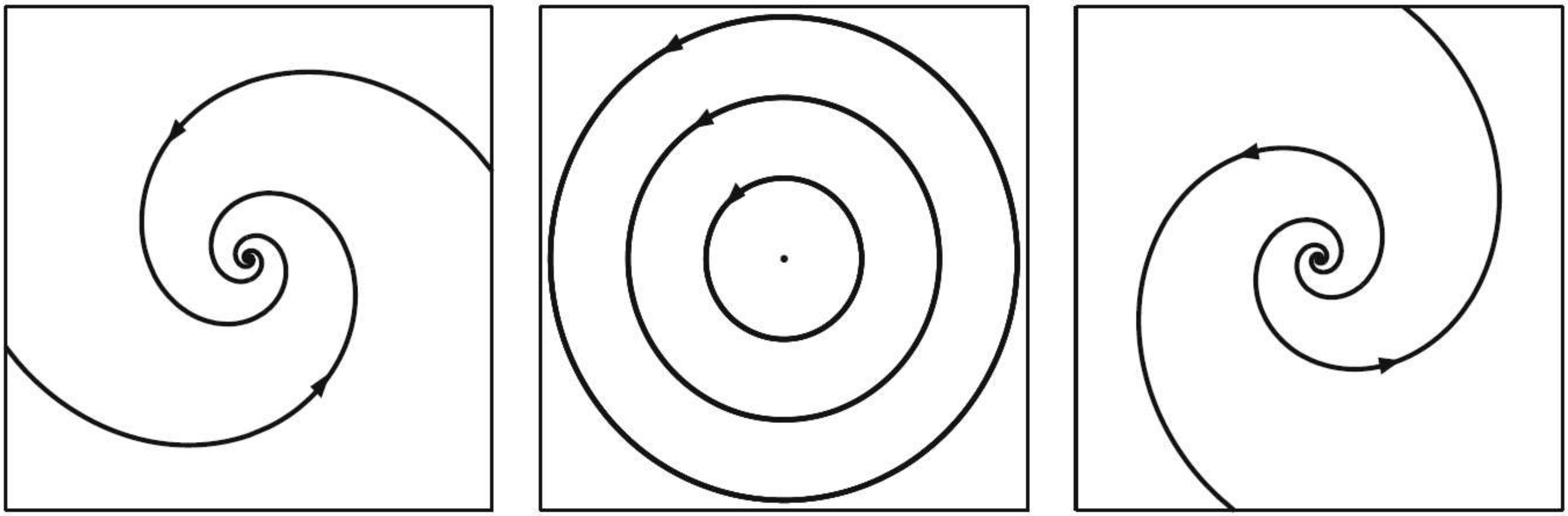
$\lambda < 0$
stable star node



$\lambda > 0$
unstable star node

Jordan Form Characterization (4)

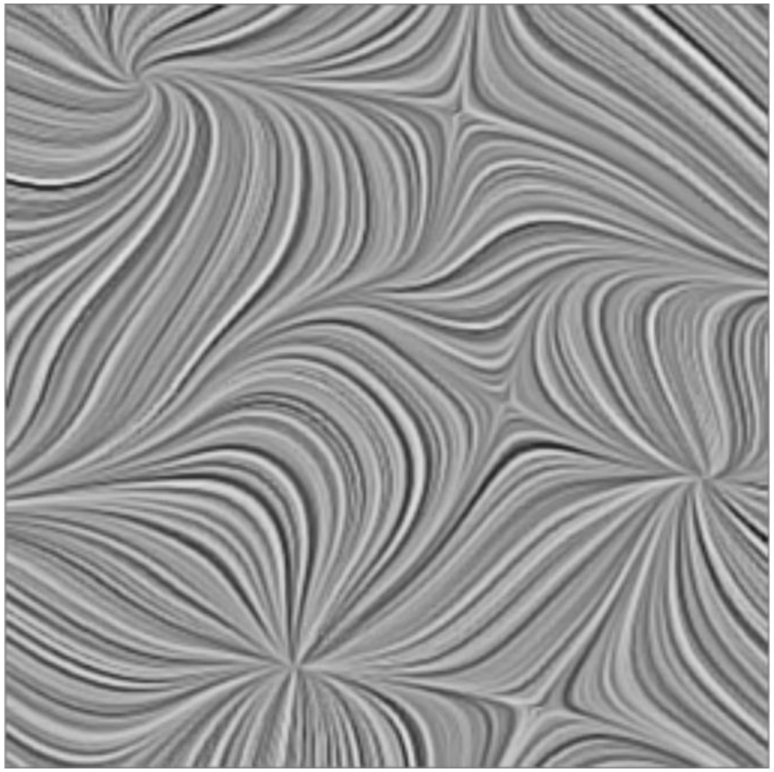
Phase portraits corresponding to Jordan matrix $J_4 = \begin{bmatrix} a & -b \\ b & a \end{bmatrix}$



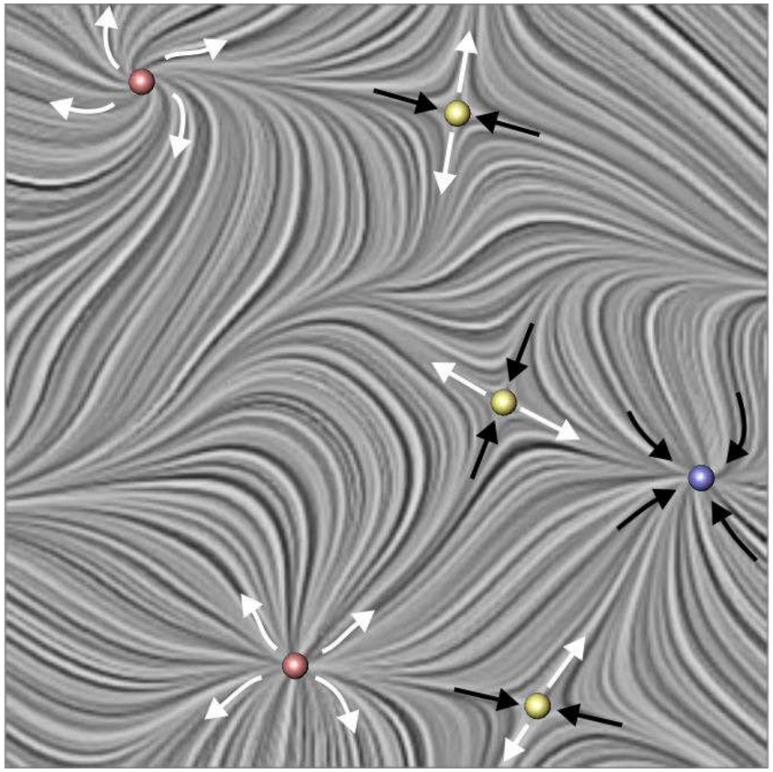
$a < 0$ stable spiral node | $a = 0$ center | $a > 0$ unstable spiral node

Critical Points (Steady Flow!)

Classify critical points according to the *eigenvalues* of the velocity gradient tensor at the critical point



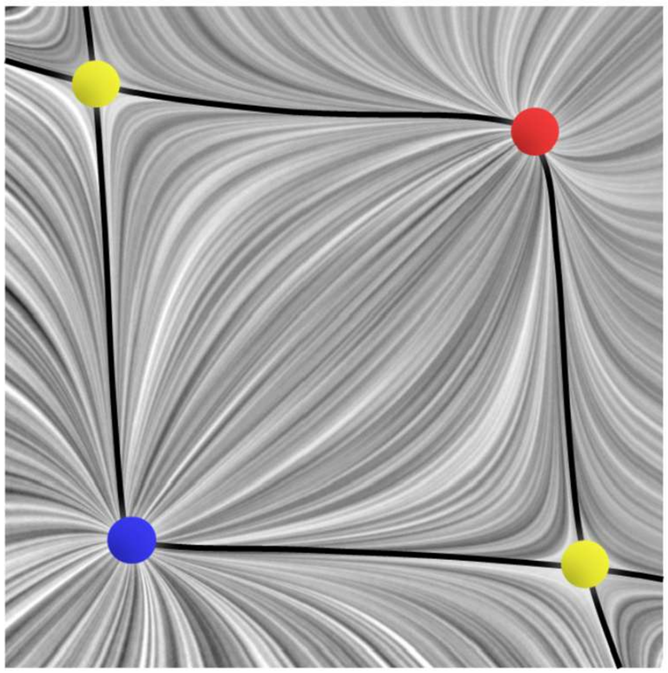
stream lines (LIC)

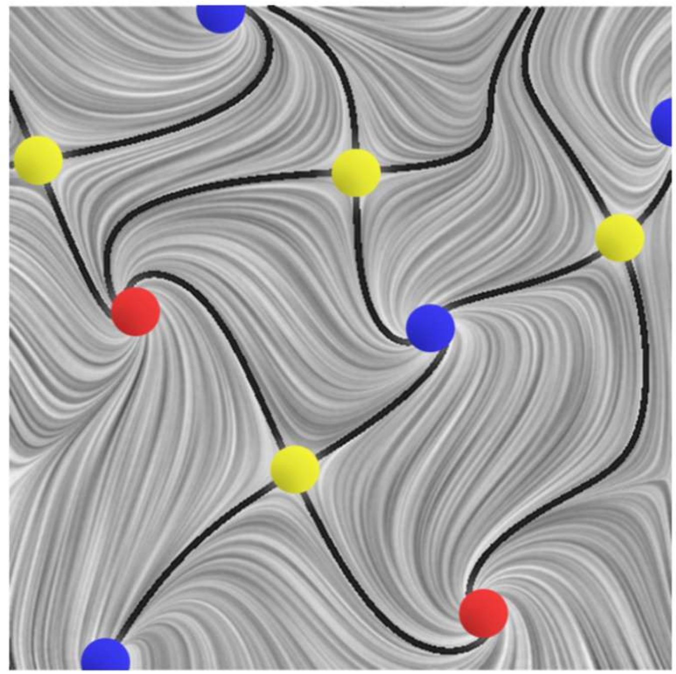


critical points (**v** = 0)

Vector Field Topology: Topological Skeleton

Connect critical points by *separatrices*

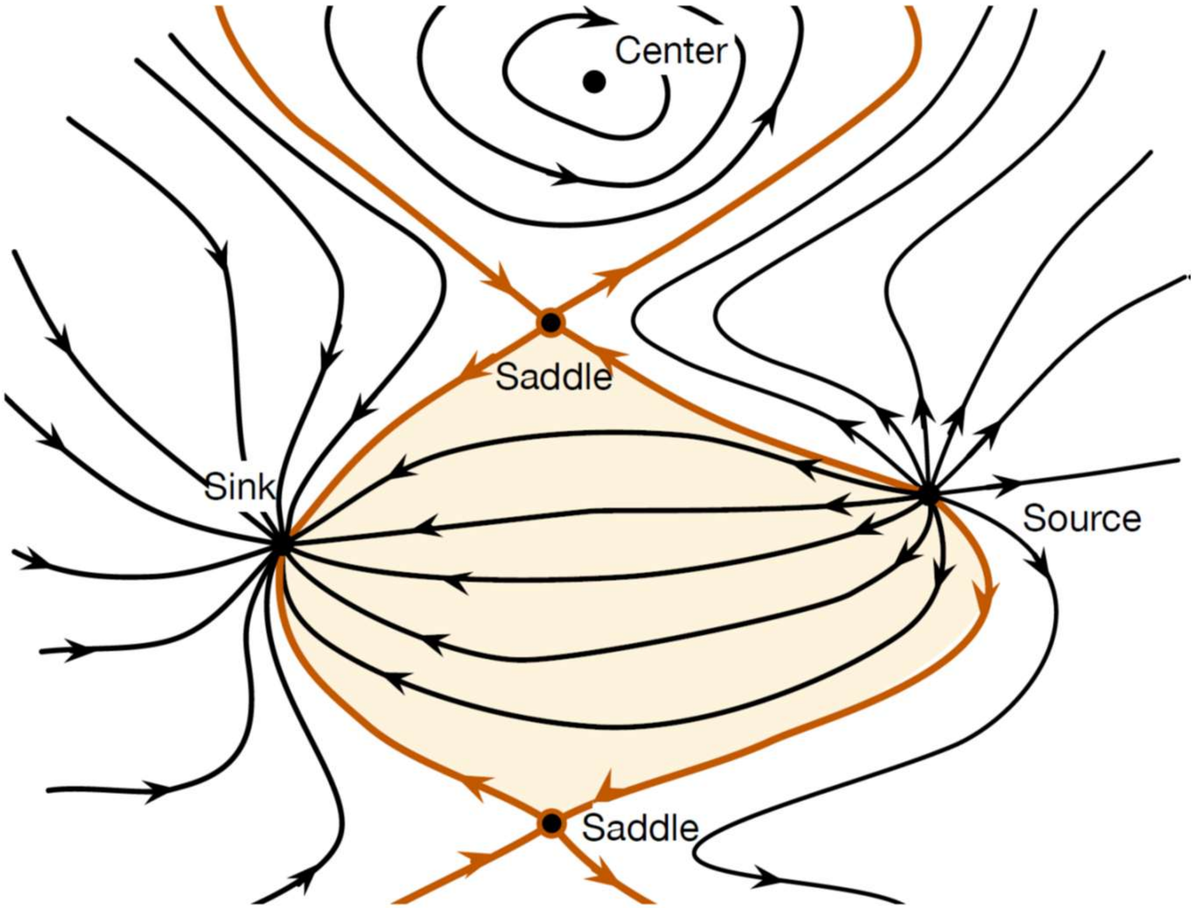




Sources (red), sinks (blue), saddles (yellow)

Vector Field Topology: Topological Skeleton

Connect critical points by *separatrices*



Index of Critical Points / Vector Fields

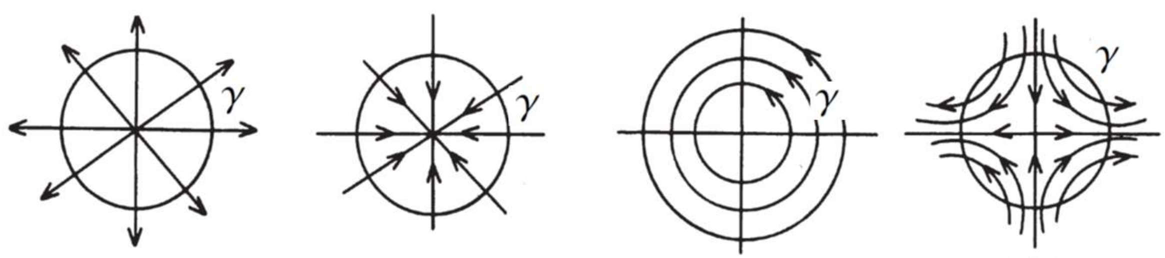
Poincaré index (in scalar field topology we have the *Morse index*)

- Can compute index (winding number) for each critical point
- Index of a region is the sum of the critical point indexes inside
- Sum of all indexes over a manifold is its Euler characteristic

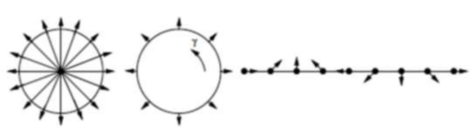
Do a loop (Jordan curve) around each critical point: the index is its (Brouwer) degree: integer how often the vector field along the loop turns around (determined by angle 1-form integrated over oriented 1-manifold)

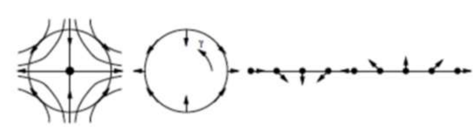
$$\text{index}_\gamma = \frac{1}{2\pi} \oint_\gamma \mathrm{d}\alpha$$

$$\alpha = \arctan \frac{v}{u}$$



index = +1   index = +1   index = +1   index = −1



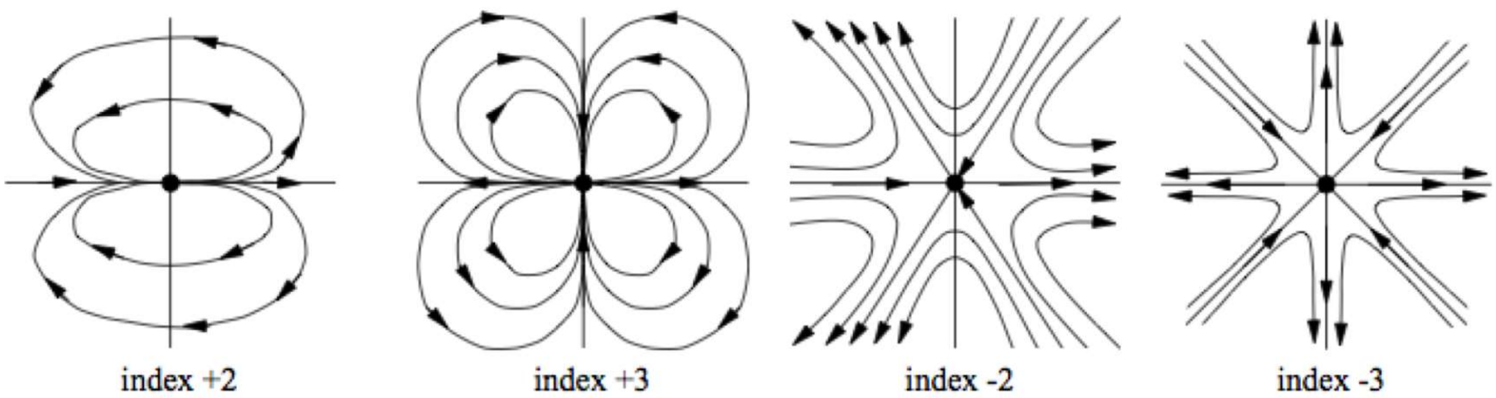


Higher-Order Critical Points

Higher than first-order

- Sectors can by elliptic, parabolic, hyperbolic
- For index sum over number of elliptic and hyperbolic sectors

$$\text{index}_{cp} = 1 + \frac{n_e - n_h}{2}$$



(dipole)

(see monkey saddle)

Example: Differential Topology

Topological information from vector fields on manifold

- Independent of actual vector field! Poincaré-Hopf theorem (sum of indexes == Euler char.)
- Useful constraints: vector field editing, simplification, sphere always has critical point, …

Topological invariant: Euler characteristic $\chi(M)$ of manifold M

(for 2-manifold mesh: $\chi(M) = V - E + F$)

$\chi = 2 - 2g$ (orientable)

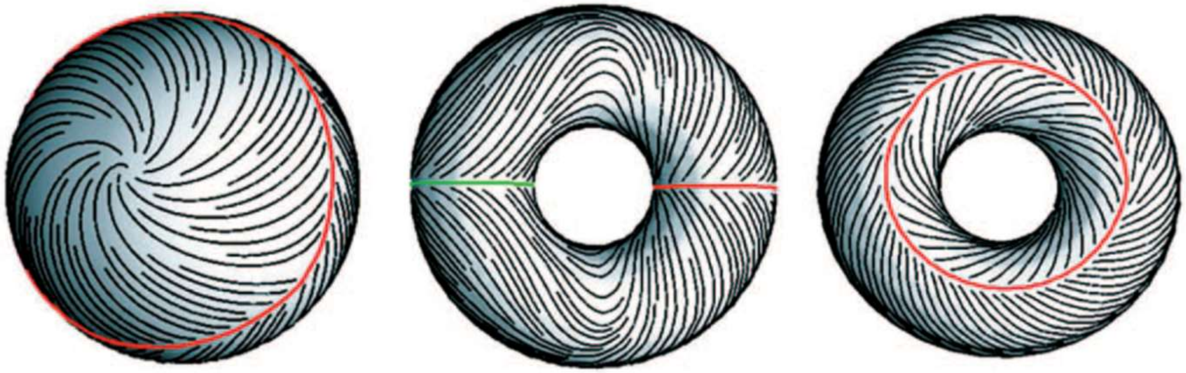
genus $g = 0$
Euler characteristic $\chi = 2$

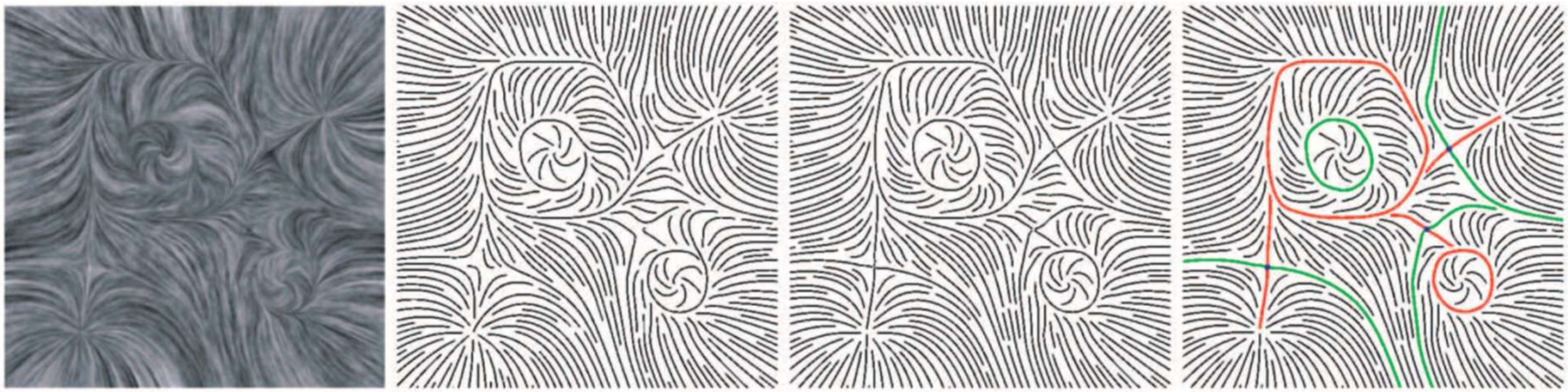
genus $g = 1$
Euler characteristic $\chi = 0$

genus $g = 2$
Euler characteristic $\chi = -2$

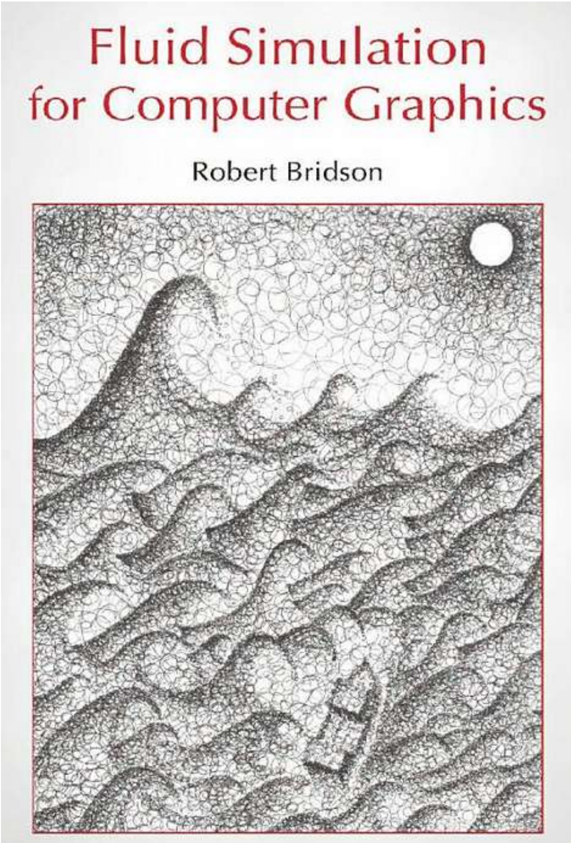
Example: Vector Field Editing

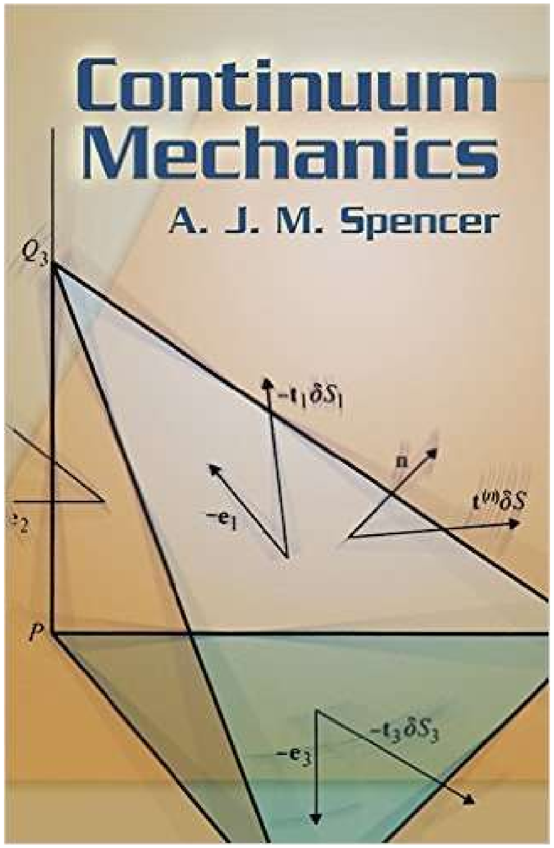
Guoning Chen et al., Vector Field Editing and Periodic Orbit Extraction Using Morse Decomposition, IEEE TVCG, 2007

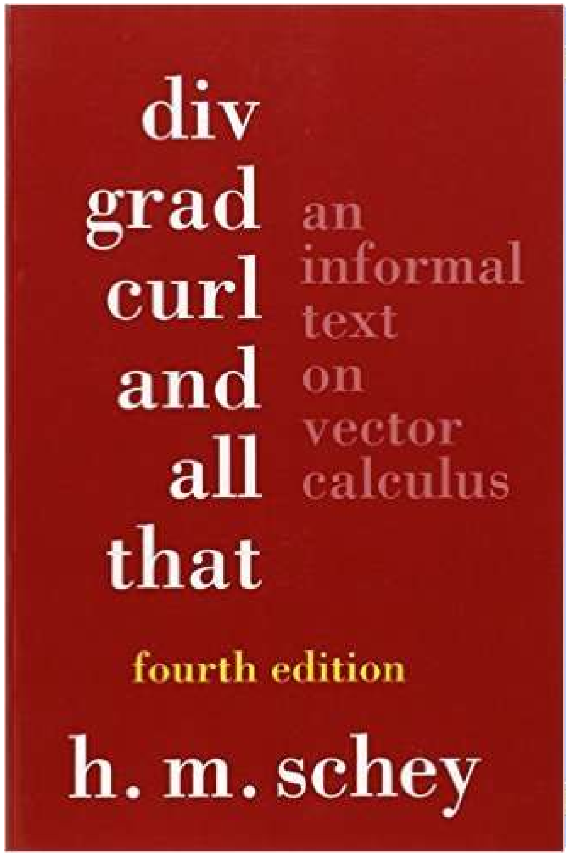




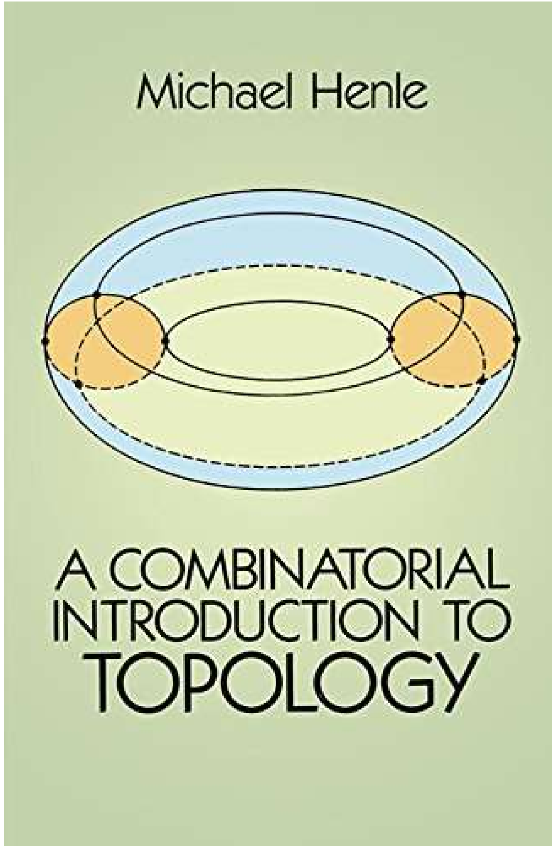
Recommended Books (1)

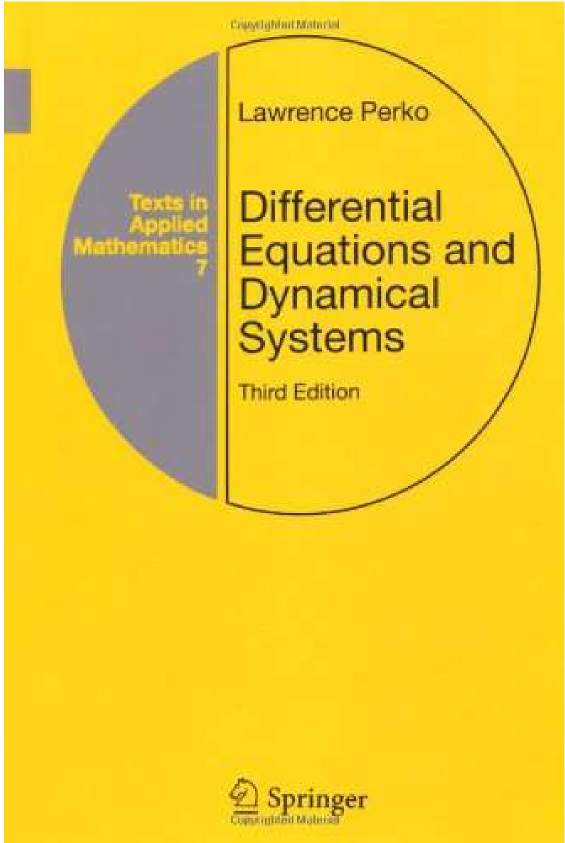


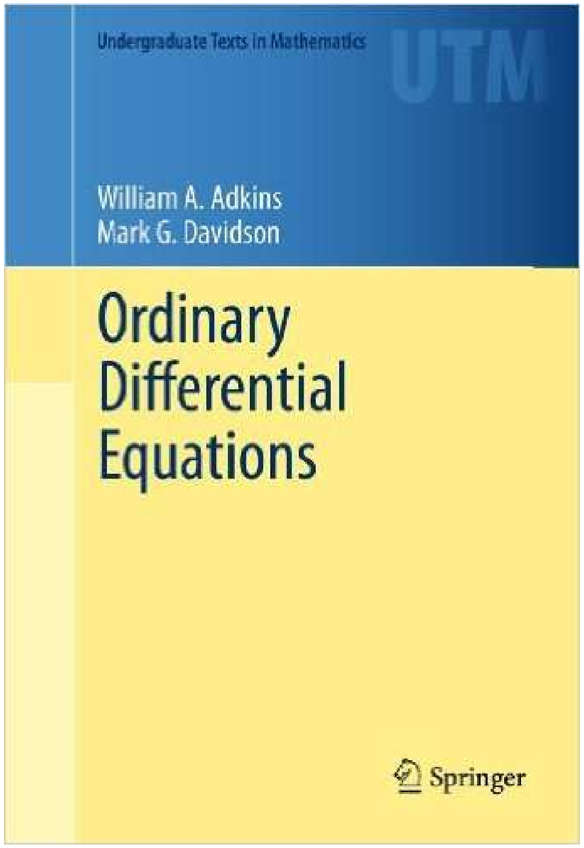




Recommended Books (2)







Thank you.

Thanks for material

- Helwig Hauser
- Eduard Gröller
- Daniel Weiskopf
- Torsten Möller
- Ronny Peikert
- Philipp Muigg
- Christof Rezk-Salama