# Flow Visualization

Part 2 (of 3)

## Retrospect: Flow Visualization, Part 1

- introduction, overview
  - simulation vs. measurement vs. modelling
  - 2D vs. surfaces vs. 3D
  - steady vs time-dependent
  - direct vs. indirect FlowVis
- experimental FlowVis
  - general possibilities
  - PIV + example
- visualization of models
- flow visualization with arrows



## Overview: Flow Visualization, Part 2

- numerical integration
  - Euler-integration
  - Runge-Kutta-integration
- streamlines
  - in 2D
  - particle paths
  - in 3D, sweeps
  - illuminated streamlines
- streamline placement



# Integration of Streamlines

Numerical Integration

## Streamlines – Theory

- Correlations:
  - flow data **v**: derivative information
  - $d\mathbf{x}/dt = \mathbf{v}(\mathbf{x})$;
    spatial points $\mathbf{x} \in R^n$, time $t \in R$, flow vectors $\mathbf{v} \in R^n$
  - streamline **s**: integration over time,
    also called trajectory, solution, curve
  - $\mathbf{s}(t) = \mathbf{s}_0 + \int_{0 \le u \le t} \mathbf{v}(\mathbf{s}(u))\, du$;
    seed point $\mathbf{s}_0$, integration variable $u$
  - difficulty: result **s** also in the integral $\Rightarrow$ analytical solution usually impossible!

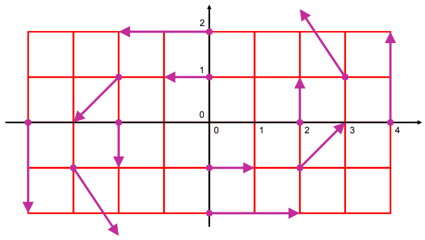


## Streamlines – Practice

- Basic approach:
  - theory: $\mathbf{s}(t) = \mathbf{s}_0 + \int_{0 \le u \le t} \mathbf{v}(\mathbf{s}(u))\, du$
  - practice: numerical integration
  - idea:
    (very) locally, the solution is (approx.) linear
  - Euler integration:
    follow the current flow vector $\mathbf{v}(\mathbf{s}_i)$ from the current streamline point $\mathbf{s}_i$ for a very small time ($dt$) and therefore distance
  - Euler integration: $\mathbf{s}_{i+1} = \mathbf{s}_i + dt \cdot \mathbf{v}(\mathbf{s}_i)$,
    integration of small steps ($dt$ very small)

## Euler Integration – Example

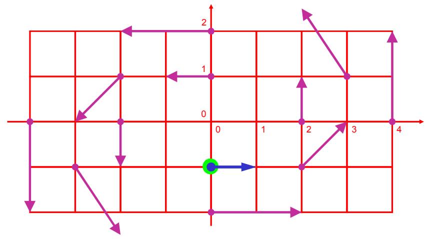
- 2D model data: $v_x = dx/dt = -y$
  $v_y = dy/dt = x/2$
- Sample arrows:
- True solution: ellipses!





## Euler Integration – Example

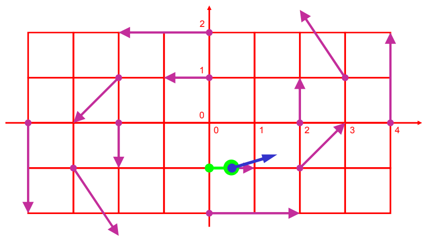
- Seed point $\mathbf{s}_0 = (0\,|\,{-1})^T$;
  current flow vector $\mathbf{v}(\mathbf{s}_0) = (1\,|\,0)^T$;
  $dt = 1/2$





## Euler Integration – Example

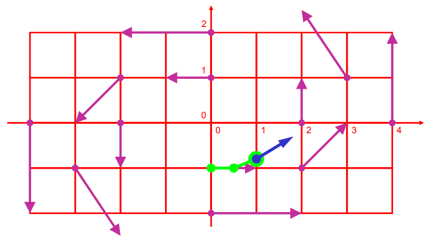
- New point $\mathbf{s}_1 = \mathbf{s}_0 + \mathbf{v}(\mathbf{s}_0) \cdot dt = (1/2\,|\,{-1})^T$;
  current flow vector $\mathbf{v}(\mathbf{s}_1) = (1\,|\,1/4)^T$;





## Euler Integration – Example

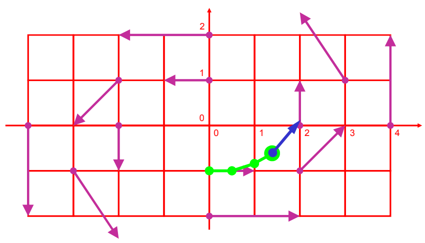
- New point $\mathbf{s}_2 = \mathbf{s}_1 + \mathbf{v}(\mathbf{s}_1) \cdot dt = (1\,|\,{-7/8})^T$;
  current flow vector $\mathbf{v}(\mathbf{s}_2) = (7/8\,|\,1/2)^T$;





## Euler Integration – Example

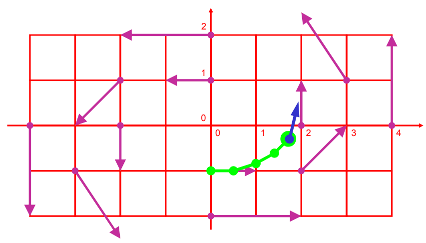
- $\mathbf{s}_3 = (23/16\,|\,{-5/8})^T \approx (1.44\,|\,{-0.63})^T$;
  $\mathbf{v}(\mathbf{s}_3) = (5/8\,|\,23/32)^T \approx (0.63\,|\,0.72)^T$;





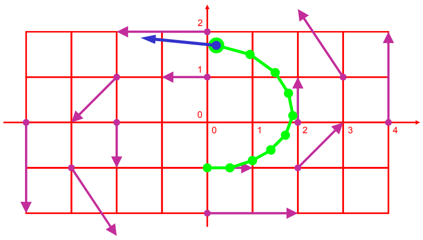
## Euler Integration – Example

- $\mathbf{s}_4 = (7/4\,|\,{-17/64})^T \approx (1.75\,|\,{-0.27})^T$;
  $\mathbf{v}(\mathbf{s}_4) = (17/64\,|\,7/8)^T \approx (0.27\,|\,0.88)^T$;



## Euler Integration – Example

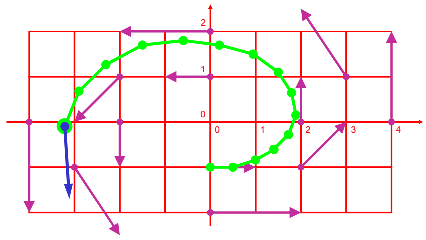
- $\mathbf{s}_9 \approx (0.20\,|\,1.69)^T$;
  $\mathbf{v}(\mathbf{s}_9) \approx (-1.69\,|\,0.10)^T$;





## Euler Integration – Example

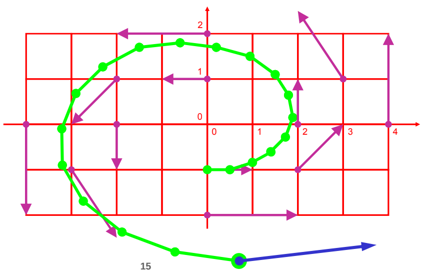
- $\mathbf{s}_{14} \approx (-3.22\,|\,-0.10)^T$;
  $\mathbf{v}(\mathbf{s}_{14}) \approx (0.10\,|\,-1.61)^T$;





## Euler Integration – Example

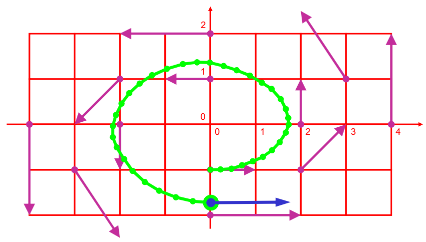
- $\mathbf{s}_{19} \approx (0.75\,|\,-3.02)^T$; $\mathbf{v}(\mathbf{s}_{19}) \approx (3.02\,|\,0.37)^T$;
  clearly: large integration error, *dt* too large!
  19 steps





## Euler Integration – Example

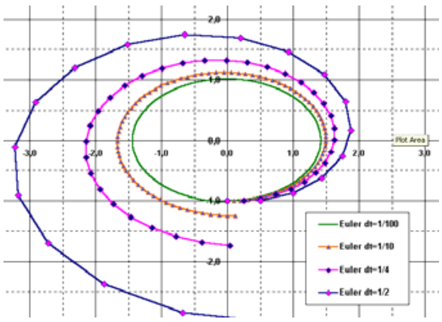
- *dt* smaller (1/4): more steps, more exact!
  $\mathbf{s}_{36} \approx (0.04\,|\,-1.74)^T$; $\mathbf{v}(\mathbf{s}_{36}) \approx (1.74\,|\,0.02)^T$;
- 36 steps





## Comparison Euler, Step Sizes

Euler is getting better proportionally to *dt*






## Euler Example – Error Table

| dt | #steps | error | |
|---|---|---|---|
| 1/2 | 19 | ~200% | |
| 1/4 | 36 | ~75% | |
| 1/10 | 89 | ~25% | |
| 1/100 | 889 | ~2% | |
| 1/1000 | 8889 | ~0.2% | ✓ |

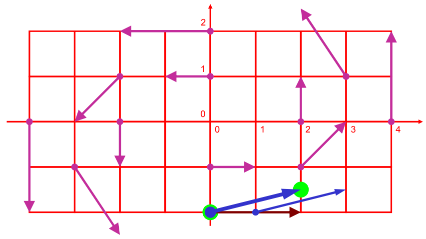
## Better than Euler Integr.: RK

- Runge-Kutta Approach:
  - theory: $\mathbf{s}(t) = \mathbf{s}_0 + \int_{0 \le u \le t} \mathbf{v}(\mathbf{s}(u))\,du$
  - Euler: $\mathbf{s}_i = \mathbf{s}_0 + \Sigma_{0 \le u < i} \mathbf{v}(\mathbf{s}_u) \cdot dt$
  - Runge-Kutta integration:
    - idea: cut short the curve arc
    - RK-2 (second order RK):
      1.: do half a Euler step
      2.: evaluate flow vector there
      3.: use it in the origin
    - RK-2 (two evaluations of **v** per step):
      $\mathbf{s}_{i+1} = \mathbf{s}_i + \mathbf{v}(\mathbf{s}_i + \mathbf{v}(\mathbf{s}_i) \cdot dt/2) \cdot dt$



## RK-2 Integration – One Step

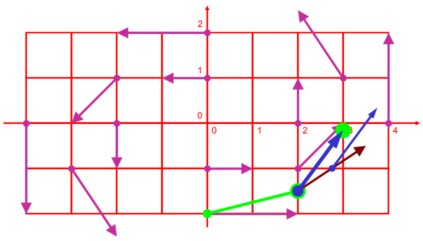
- Seed point $\mathbf{s}_0 = (0|-2)^T$;
  current flow vector $\mathbf{v}(\mathbf{s}_0) = (2|0)^T$;
  preview vector $\mathbf{v}(\mathbf{s}_0 + \mathbf{v}(\mathbf{s}_0) \cdot dt/2) = (2|0.5)^T$;
  $dt = 1$






## RK-2 – One more step

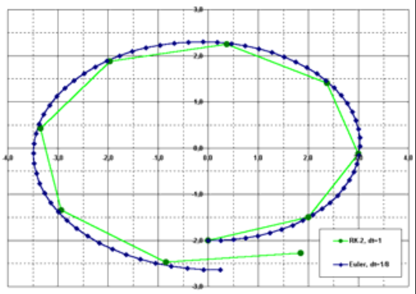
- Seed point $\mathbf{s}_1 = (2|-1.5)^T$;
  current flow vector $\mathbf{v}(\mathbf{s}_1) = (1.5|1)^T$;
  preview vector $\mathbf{v}(\mathbf{s}_1 + \mathbf{v}(\mathbf{s}_1) \cdot dt/2) \approx (1|1.4)^T$;
  $dt = 1$






## RK-2 – A Quick Round

- RK-2: even with $dt=1$ (9 steps) better than Euler with $dt=1/8$ (72 steps)





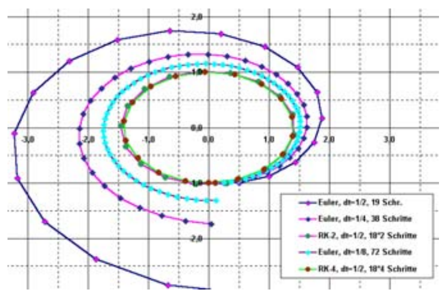
## RK-4 vs. Euler, RK-2

- Even better: fourth order RK:
  - four vectors **a**, **b**, **c**, **d**
  - one step is a convex combination:
    $\mathbf{s}_{i+1} = \mathbf{s}_i + (\mathbf{a} + 2 \cdot \mathbf{b} + 2 \cdot \mathbf{c} + \mathbf{d})/6$
  - vectors:
    - $\mathbf{a} = dt \cdot \mathbf{v}(\mathbf{s}_i)$ … original vector
    - $\mathbf{b} = dt \cdot \mathbf{v}(\mathbf{s}_i + \mathbf{a}/2)$ … RK-2 vector
    - $\mathbf{c} = dt \cdot \mathbf{v}(\mathbf{s}_i + \mathbf{b}/2)$ … use RK-2 …
    - $\mathbf{d} = dt \cdot \mathbf{v}(\mathbf{s}_i + \mathbf{c})$ … and again!



## Euler vs. Runge-Kutta

- RK-4: pays off only with complex flows
- Here approx. like RK-2



## Integration, Conclusions

- Summary:
  - analytic determination of streamlines usually not possible
  - hence: numerical integration
  - several methods available (Euler, Runge-Kutta, etc.)
  - Euler: simple, imprecise, esp. with small $dt$
  - RK: more accurate in higher orders
  - furthermore: adaptive methods, implicit methods, etc.

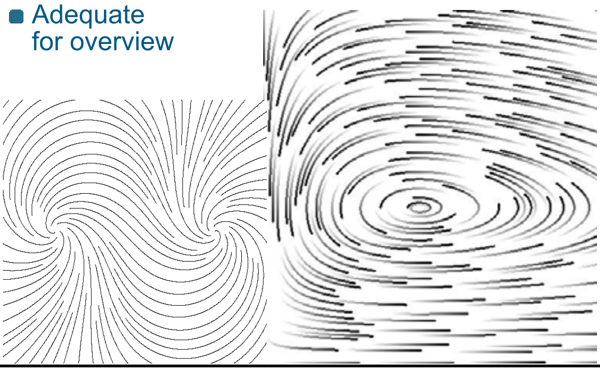


# Flow Visualization with Streamlines

Streamlines,
Particle Paths, etc.

## Streamlines in 2D

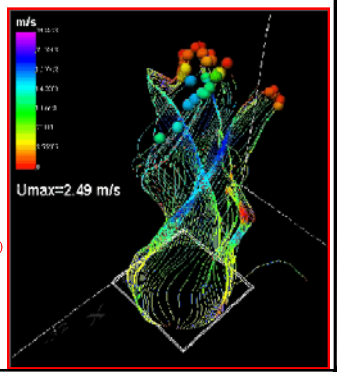
- Adequate for overview



## Visualization with Particles

- Particle paths = streamlines (steady flows)
- Variants (time-dependent data):
  - streak lines: steadily new particles
  - path lines: long-term path of one particle

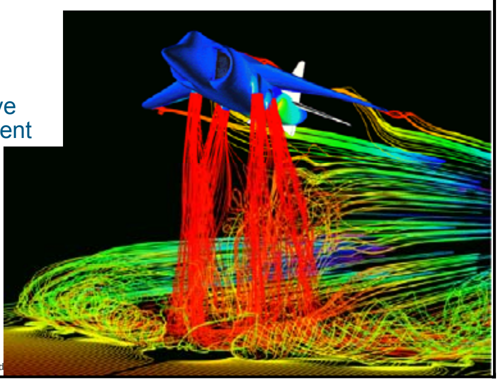
click2demo (F9!)






## Streamlines in 3D

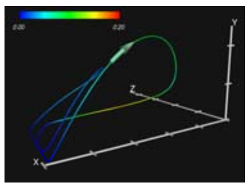
- Color coding: Speed
- Selective Placement

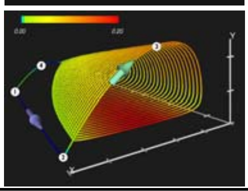




## 3D Streamlines with Sweeps

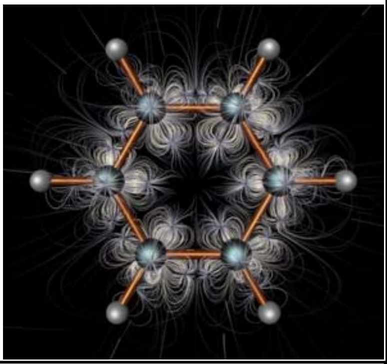
- Sweeps: better spatial 3D perception





## Illuminated Streamlines

- Illuminated 3D curves ⇒ better 3D perception!



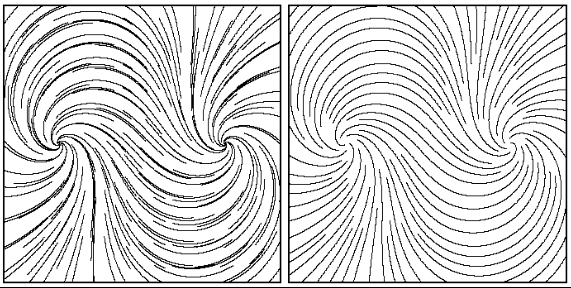


## Streamline Placement

in 2D

## Problem: Choice of Seed Points

- Streamline placement:
  - If regular grid used: very irregular result





## Overview of Algorithm

- Idea: streamlines should not get too near to each other
- Approach:
  - choose a seed point with distance $d_{sep}$ from an already existing streamline
  - forward- and backward-integration until distance $d_{test}$ is reached (or …).
  - two parameters:
    - $d_{sep}$ … start distance
    - $d_{test}$ … minimum distance



## Algorithm – Pseudocode

- Compute initial streamline, put it into a queue
- Initial streamline becomes current streamline
- WHILE not finished DO:

TRY: get new seed point which is $d_{sep}$ away from current streamline

IF successful THEN compute new streamline and put to queue

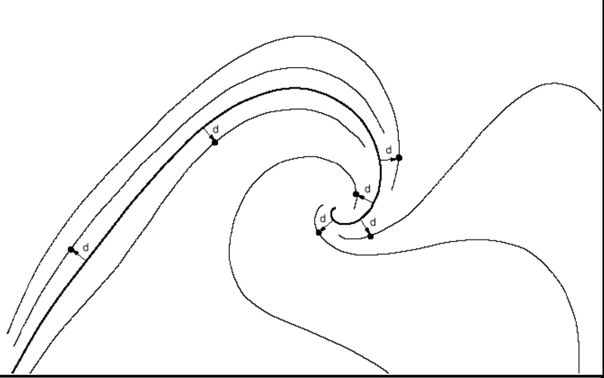
ELSE IF no more streamline in queue
THEN exit loop
ELSE next streamline in queue becomes current streamline



## Streamline Termination

- When to stop streamline integration:
  - when dist. to neighboring streamline ≤ $d_{test}$
  - when streamline leaves flow domain
  - when streamline runs into fixed point (**v**=0)
  - when streamline gets too near to itself
  - after a certain amount of maximal steps

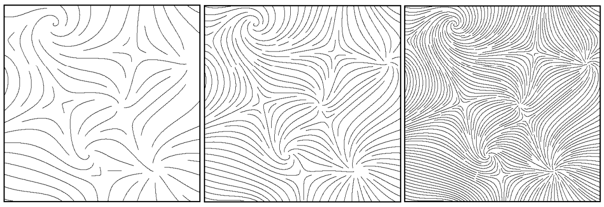
## New Streamlines



## Different Streamline Densities

- Variations of $d_{sep}$ in rel. to image width:

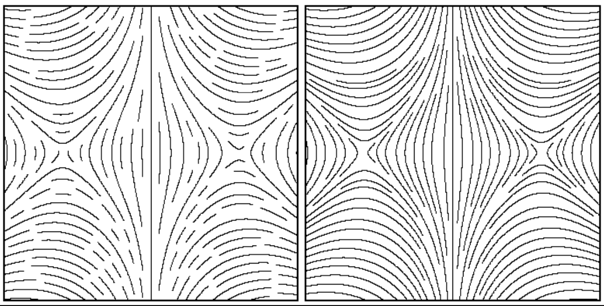
6% 3% 1.5%





## $d_{sep}$ vs. $d_{test}$

$d_{test} = 0.9 \cdot d_{sep}$ $d_{test} = 0.5 \cdot d_{sep}$

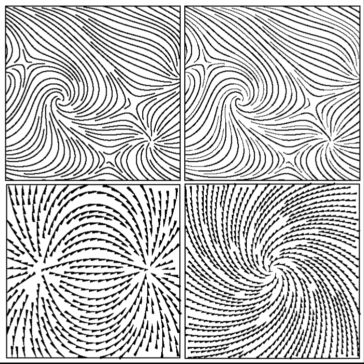


## Tapering and Glyphs

- Thickness in rel. to dist.

$$\begin{cases} 1.0 & \forall d \geq d_{sep} \\ \frac{d - d_{test}}{d_{sep} - d_{test}} & \forall d < d_{sep} \end{cases}$$

- Directional glyphs:





## Literature

- Paper (more details):
  - B. Jobard & W. Lefer: "**Creating Evenly-Spaced Streamlines of Arbitrary Density**" in *Proceedings of 8th Eurographics Workshop on Visualization in Scientific Computing*, April 1997, pp. 45-55



## Acknowledgements


- For material used in this lecture:
  - Bruno Jobard
  - Malte Zöckler
  - Georg Fischel
  - Frits Post
  - Roger Crawfis
  - myself... ;–) (i.e., Helwig Hauser)
  - etc.