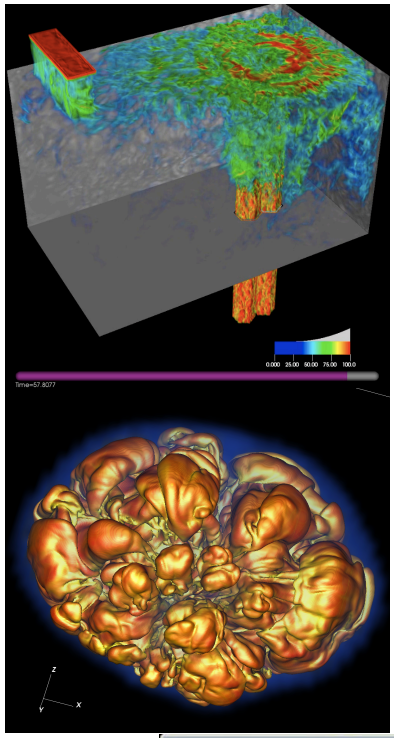
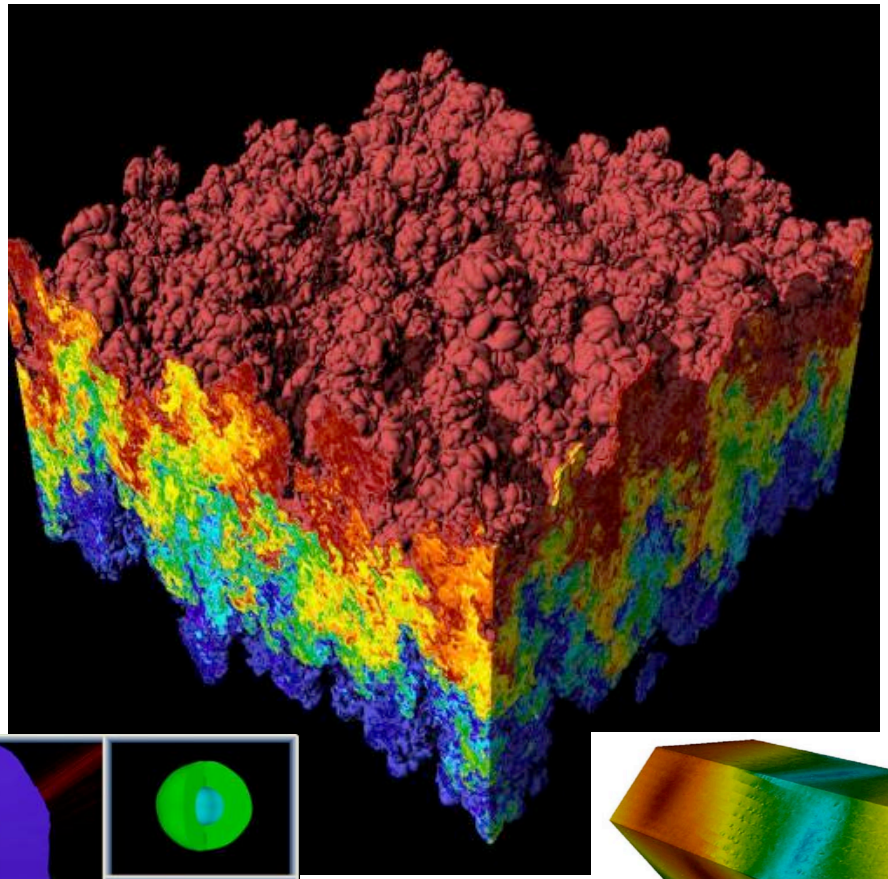
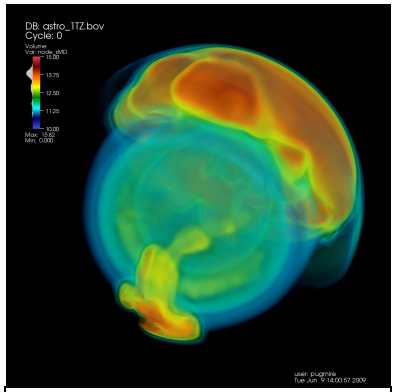
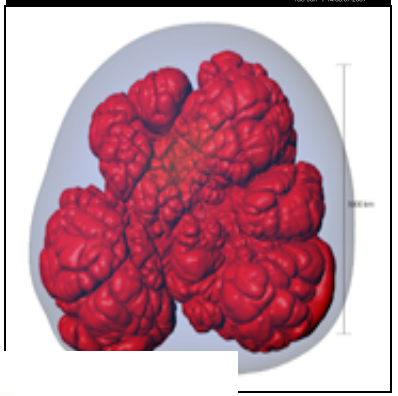
UNIVERSITY OF OREGON

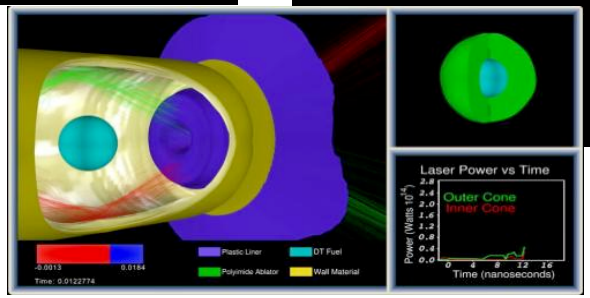
CIS 410/510: Advection

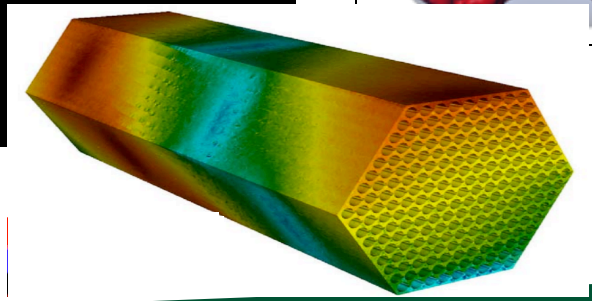












Lecture #6

Hank Childs, University of Oregon

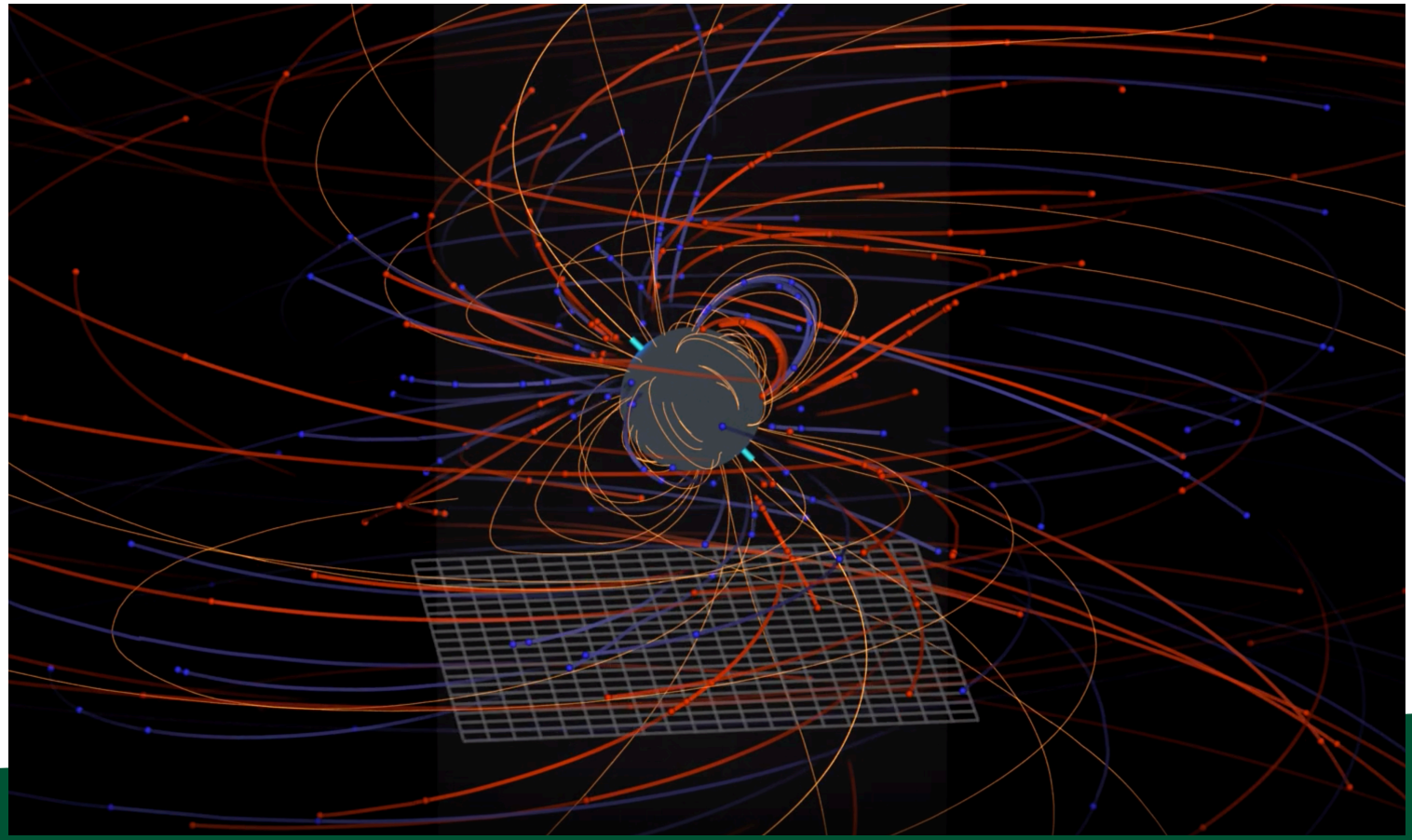
Quiz #2

- Focal point this time: minimal noise
  - At home? – use Zoom chat
  - In class? – come to the front
- Canvas folks:
  - Hyperlink in quiz (also full link)
- Let's look at the quiz

Advection

One Example Flow Vis Technique

- https://svs.gsfc.nasa.gov/4644



Advection

ad·vec·tion (ăd-vĕk′shən)

n.

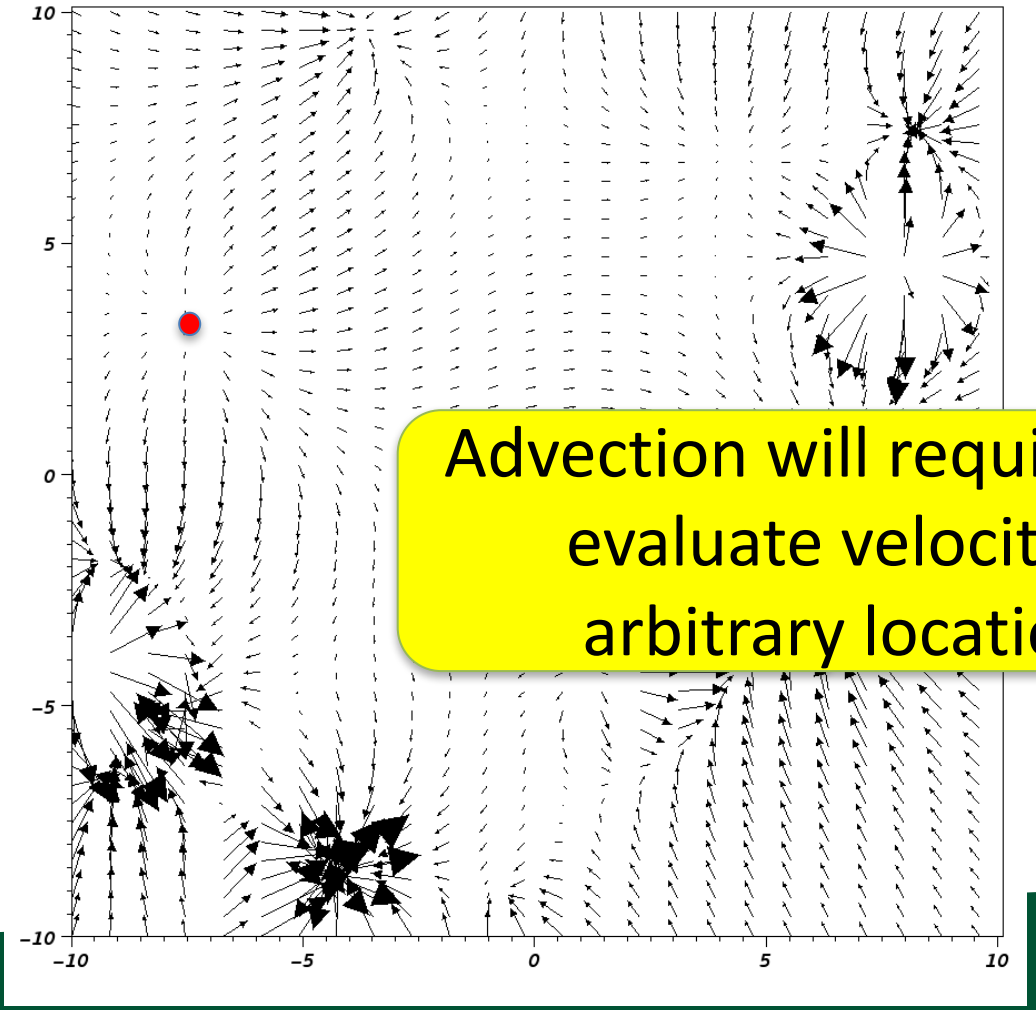
1. The transfer of a property of the atmosphere, such as heat, cold, or humidity, by the horizontal movement of an air mass: *Today's temperatures were higher due to the advection of warm air into the region.*

2. The rate of change of an atmospheric property caused by the horizontal movement of air.

3. The horizontal movement of water, as in an ocean current.

[Latin `advectiō, advectiōn-`, *act of conveying*, from `advectus`, past participle of `advehere`, *to carry to* : `ad-`, *ad-* + `vehere`, *to carry*; see `wegh-` in Indo-European roots.]

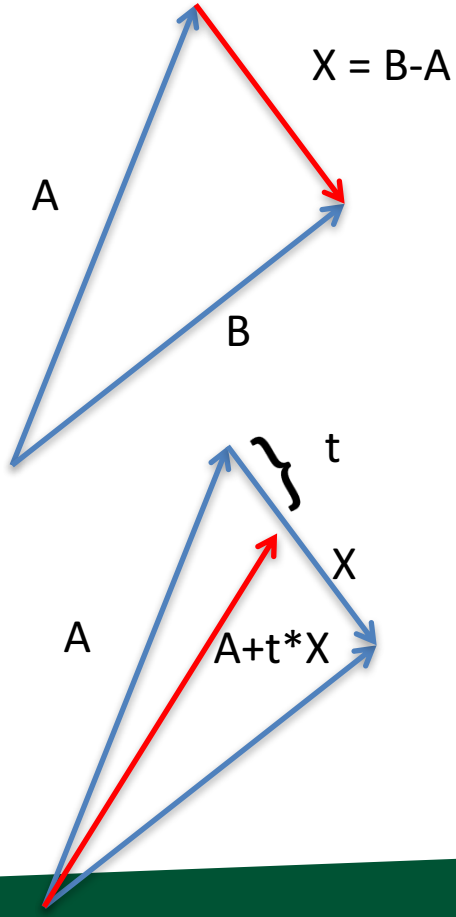
Particle Advection is the Foundation to Many Visualization Algorithms



Advection will require us to evaluate velocity at arbitrary locations

LERPing Vectors

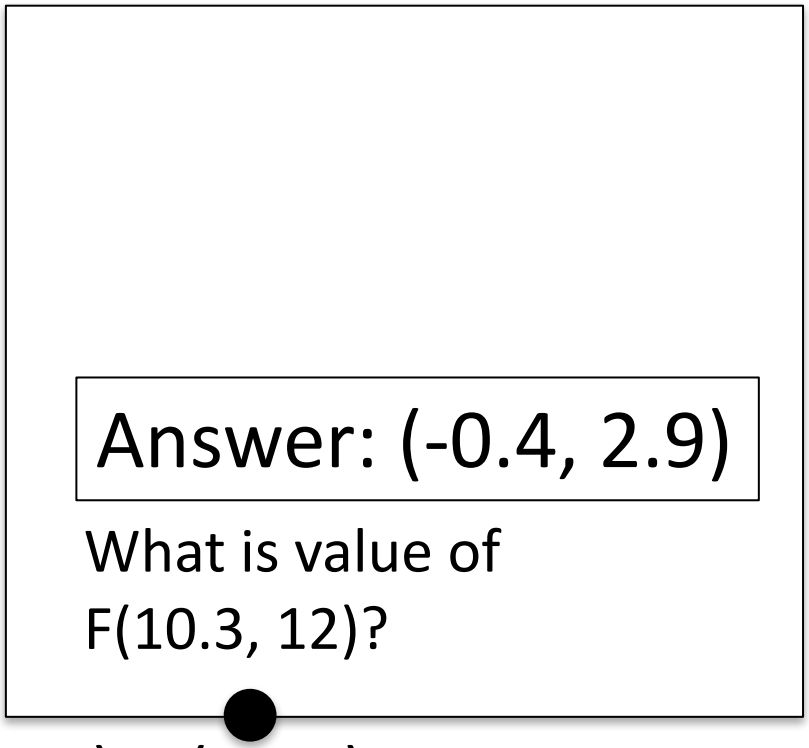
- LERP = Linear intERPolate
- Goal: interpolate vector between A and B
- Consider vector X, where X = B-A
- A + t*(B-A) = A+t*X
- Takeaway:
  - you can LERP the components individually
  - the above provides motivation for why this works



Quiz Time: LERPing Vectors

F(10,13) = (1,1)

F(11,13) = (6,1)



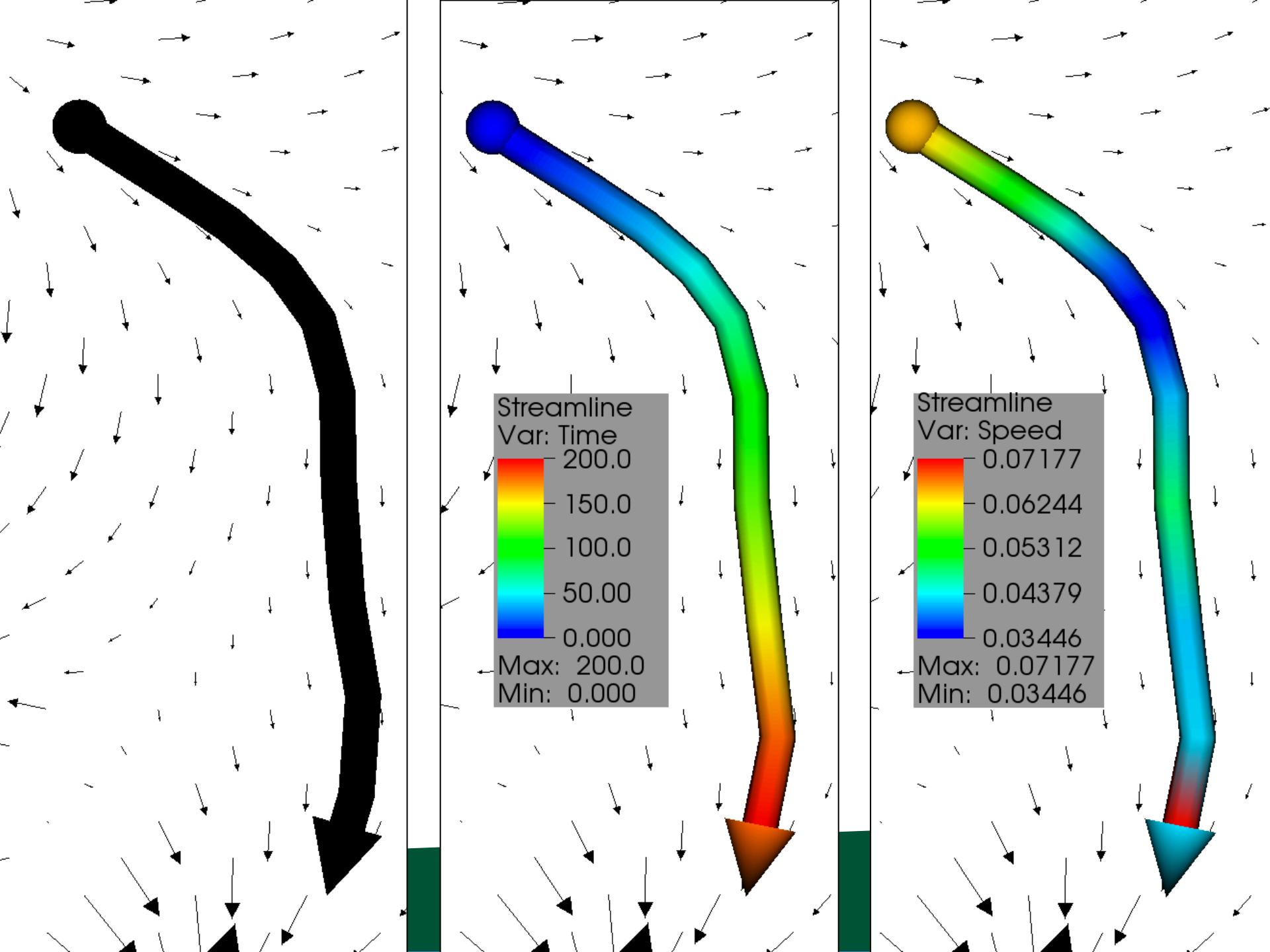
F(10,12) = (-1, 2)

F(11,12) = (1,5)

Overview of Advection Process

- Place a massless particle at a seed location
- Displace the particle according to the vector field
- Result is an “integral curve” corresponding to the trajectory the particle travels
- Math gets tricky

What would be the difference between a massless and “mass-ful” particle?


Streamline
Var: Time
200.0
150.0
100.0
50.00
0.000
Max: 200.0
Min: 0.000
Streamline
Var: Speed
0.07177
0.06244
0.05312
0.04379
0.03446
Max: 0.07177
Min: 0.03446

Formal Definition for Particle Advection

- Output is an integral curve, S, which follows trajectory of the advection
- S(t) = position of curve at time t
  - $S(t_0) = p_0$
    - t_0: initial time
    - p_0: initial position
  - $S'(t) = v(t, S(t))$
    - v(t, p): velocity at time t and position p
    - S'(t): derivative of the integral curve at time t

This is an ordinary differential equation (ODE)

The Integral Curve from Particle Advection Is Calculated Iteratively

Many possible criteria for ShouldContinue. For now, assume fixed number of steps.

$S(t_0) = p_0$

while (ShouldContinue())

{

$S(t_i) = \text{AdvanceStep}(S(t_{(i-1)})$

}

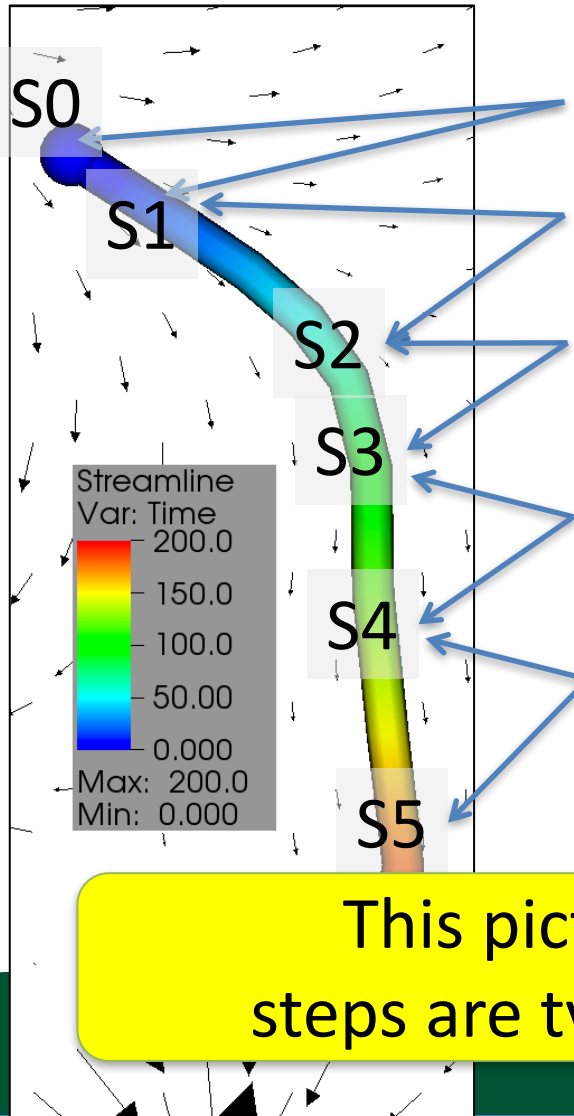
Integral Curve Calculation with a Fixed Number of Steps

$S_0 = P_0$

```
for (int i = 1 ; i < numSteps ; i++)
{
    Si = AdvanceStep(S(i-1))
}
```

How to do an advance?

AdvanceStep Goal: to Calculate S_i from $S_{(i-1)}$



$S_1 = \text{AdvanceStep}(S_0)$

$S_2 = \text{AdvanceStep}(S_1)$

$S_3 = \text{AdvanceStep}(S_2)$

$S_4 = \text{AdvanceStep}(S_3)$

$S_5 = \text{AdvanceStep}(S_4)$

This picture is misleading:
steps are typically much smaller

AdvanceStep Overview

- Think of AdvanceStep as a function:
  - Input arguments:
    - $S_{(i-1)}$: Position, time
  - Output arguments:
    - S_i: New position, new time (later than input time)
  - Optional input arguments:
    - More parameters to control the stepping process

AdvanceStep Overview

- Different numerical methods for implementing AdvanceStep:
  - Simplest version: Euler step
  - Most common: Runge-Kutta-4 (RK4)
  - Several others as well

Euler Method

- Most basic method for solving an ODE
- Idea:
  - First, choose step size: h
  - Second, AdvanceStep(p_i, t_i) returns:
    - New position: $p_{i+1} = p_i + h*v(t_i, p_i)$
    - New time: $t_{i+1} = t_i + h$

Quiz Time: Euler Method

- Euler Method:
  - New position: $p_{i+1} = p_i + h*v(t_i, p_i)$
  - New time: $t_{i+1} = t_i + h$
- Let h=0.01s
- Let $p_0 = (1,2,1)$
- Let $t_0 = 0s$
- Let $v(p_0, t_0) = (-1, -2, -1)$
- What is (p_1, t_1) if you are using the Euler method?

Answer: ((0.99, 1.98, 0.99), 0.01)

Quiz Time #2: Euler Method

- Euler Method:
  - New position: $p_{i+1} = p_i + h*v(t_i, p_i)$
  - New time: $t_{i+1} = t_i + h$
- Let h=0.01s
- Let p_1 = (0.99,1.98,0.99)
- Let t_1 = 0.01s
- Let $v(p_1, t_1)$ = (1, 2, 1)
- What is (p_2, t_2) if you are using the Euler method?

Answer: ((1, 2, 1), 0.02)

Quiz Time #3: Euler Method

- Euler Method:
  - New position: $p_{i+1} = p_i + h*v(t_i, p_i)$
  - New time: $t_{i+1} = t_i + h$
- Let h=0.01s
- Let $p_2 = (1,2,1)$
- Let $t_2 = 0.02s$
- Let $v(p_2, t_2) = (1, 0, 0)$
- What is (p_3, t_3) if you are using the Euler method?

Answer: ((1.01, 2, 1), 0.03)

Euler Method: Pros and Cons

- Pros:
  - Simple to implement
  - Computationally very efficient
- (quiz) Cons:
  - Prone to inaccuracy

- Above statements are an oversimplification:
  - Can be very accurate with small steps size, but then also very inefficient
  - Can be very fast, but then also inaccurate

Quiz Time

- You want to perform a particle advection
- What inputs do you need?
  - Velocity field
  - Step size
  - Termination criteria / # of steps
  - Initial seed position / time

Quiz Time

- Write down pseudo-code to do advection with an Euler step:
  - Initial seed location: (0,0,0)
  - Initial seed time: 0s
  - Step size = 0.01s
  - Velocity field: v
  - Termination criteria: advance 0.1s (take 10 steps)
  - Function to evaluate velocity:
    - EvaluateVelocity(position, time)

Quiz Time (answer)

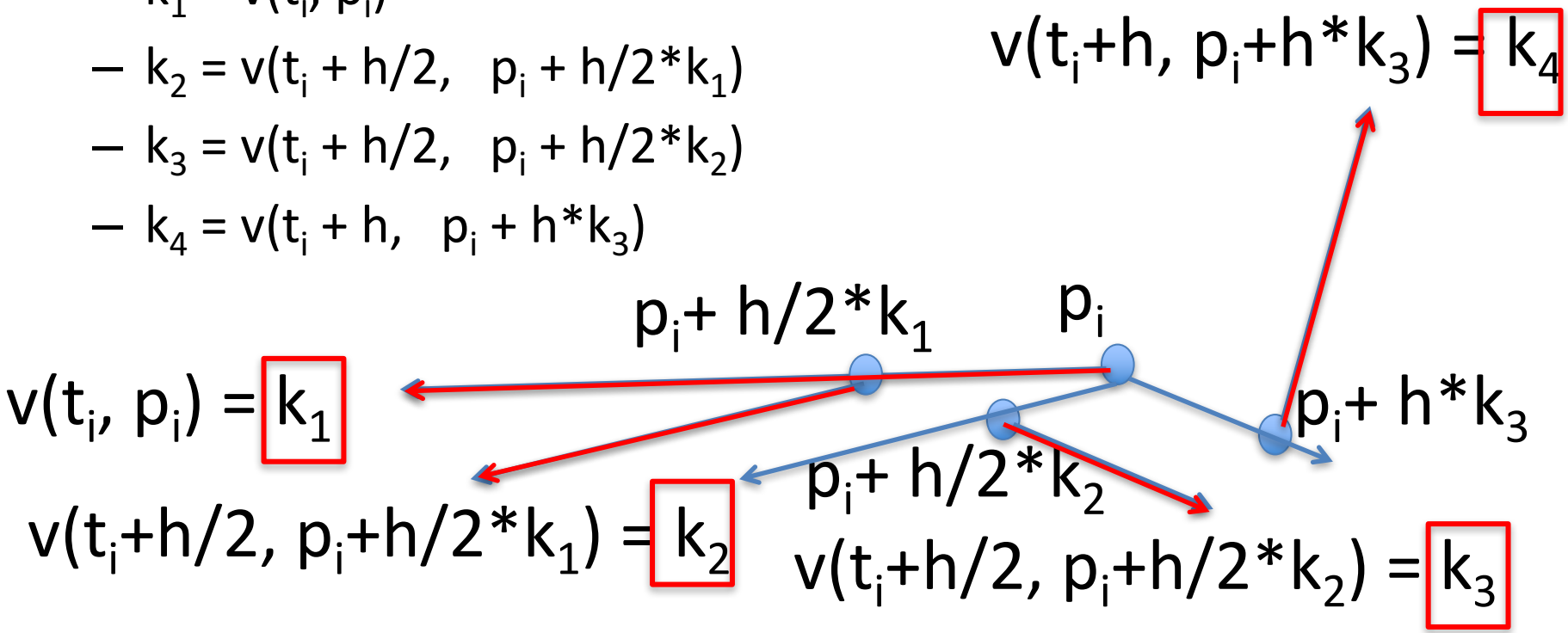
```
S[0] = (0,0,0);
time = 0;
for (int i = 0 ; i < 10 ; i++)
{
    S[i+1] = S[i]+h*EvaluateVelocity(S[i], time);
    time += h;
}
```

Runge-Kutta Method (RK4)

- Most common method for solving an ODE
- Definition:
  - First, choose step size, h
  - Second, AdvanceStep(p_i, t_i) returns:
    - New position: $p_{i+1} = p_i+(1/6)*h*(k_1+2k_2+2k_3+k_4)$
      - $k_1 = v(t_i, p_i)$
      - $k_2 = v(t_i + h/2,\ \ p_i + h/2*k_1)$
      - $k_3 = v(t_i + h/2,\ \ p_i + h/2*k_2)$
      - $k_4 = v(t_i + h,\ \ p_i + h*k_3)$
    - New time: $t_{i+1} = t_i+h$

Physical interpretation of RK4

- New position: $p_{i+1} = p_i+(1/6)*h*(k_1+2k_2+2k_3+k_4)$
  - $k_1 = v(t_i, p_i)$
  - $k_2 = v(t_i + h/2,\ \ p_i + h/2*k_1)$
  - $k_3 = v(t_i + h/2,\ \ p_i + h/2*k_2)$
  - $k_4 = v(t_i + h,\ \ p_i + h*k_3)$



Evaluate 4 velocities, use combination to calculate p_{i+1}

Quiz time: Runge-Kutta 4

- Just kidding

Runge-Kutta vs Euler

- Euler Method:
  - "1st order numerical method for solving ordinary differential equations (ODEs)"
    → error per step is $O(h^2)$, total error is $O(h)$
- Runge-Kutta:
  - "4th order numerical method for solving ODEs"
    → error per step is $O(h^5)$, total error is $O(h^4)$

h is small, so h^x is smaller still

- Oversimplification: all of RK4's "look-aheads" prevents you from stepping too far into the wrong place

Quiz Time: RK4 vs Euler

- Let h=0.01s
  - How many velocity field evaluations for RK4 to advance 1s?
  - How many velocity field evaluations for Euler to advance 1s?

400 for RK4, 100 for Euler

Quiz Time: RK4 vs Euler

- Let h=0.01 for an RK4
- What h for Euler to achieve similar accuracy?

- Error for RK4 is: O(1e-2^4) = O(1e-8)
- Error for Euler is: O(h) --- > h=1e-8

- What is the ratio of velocity evaluations for Euler to achieve same accuracy?

250000X more for Euler

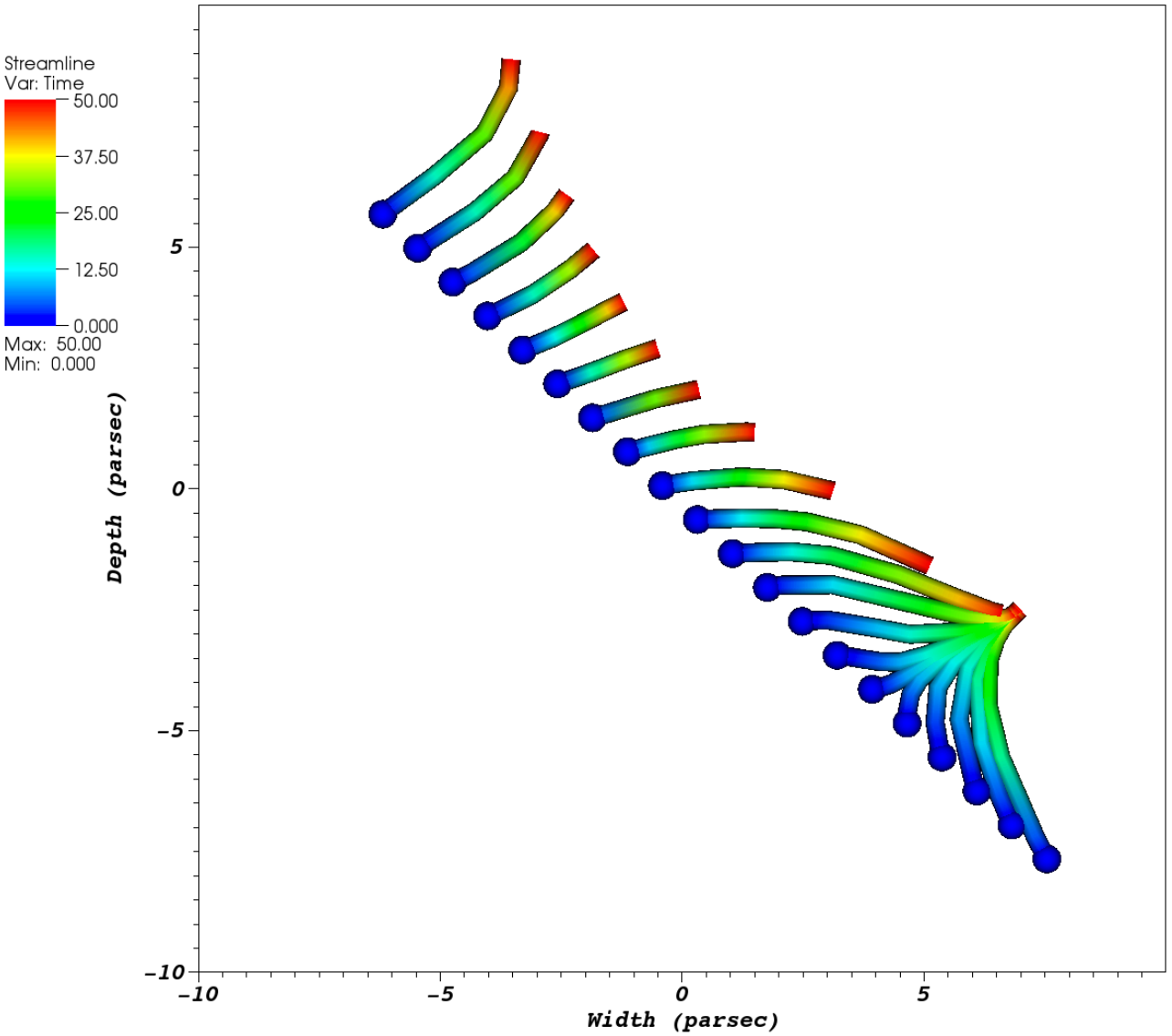
Other ODE solvers

- Adams/Bashforth, Dormand/Prince are also used
- Idea: adaptive step size
  - If the field is homogeneous, then take a bigger step size (bigger h)
  - If the field is heterogeneous, then take a smaller step size
- Quiz: why would you want to do this?

Answer: great accuracy at reduced computational cost

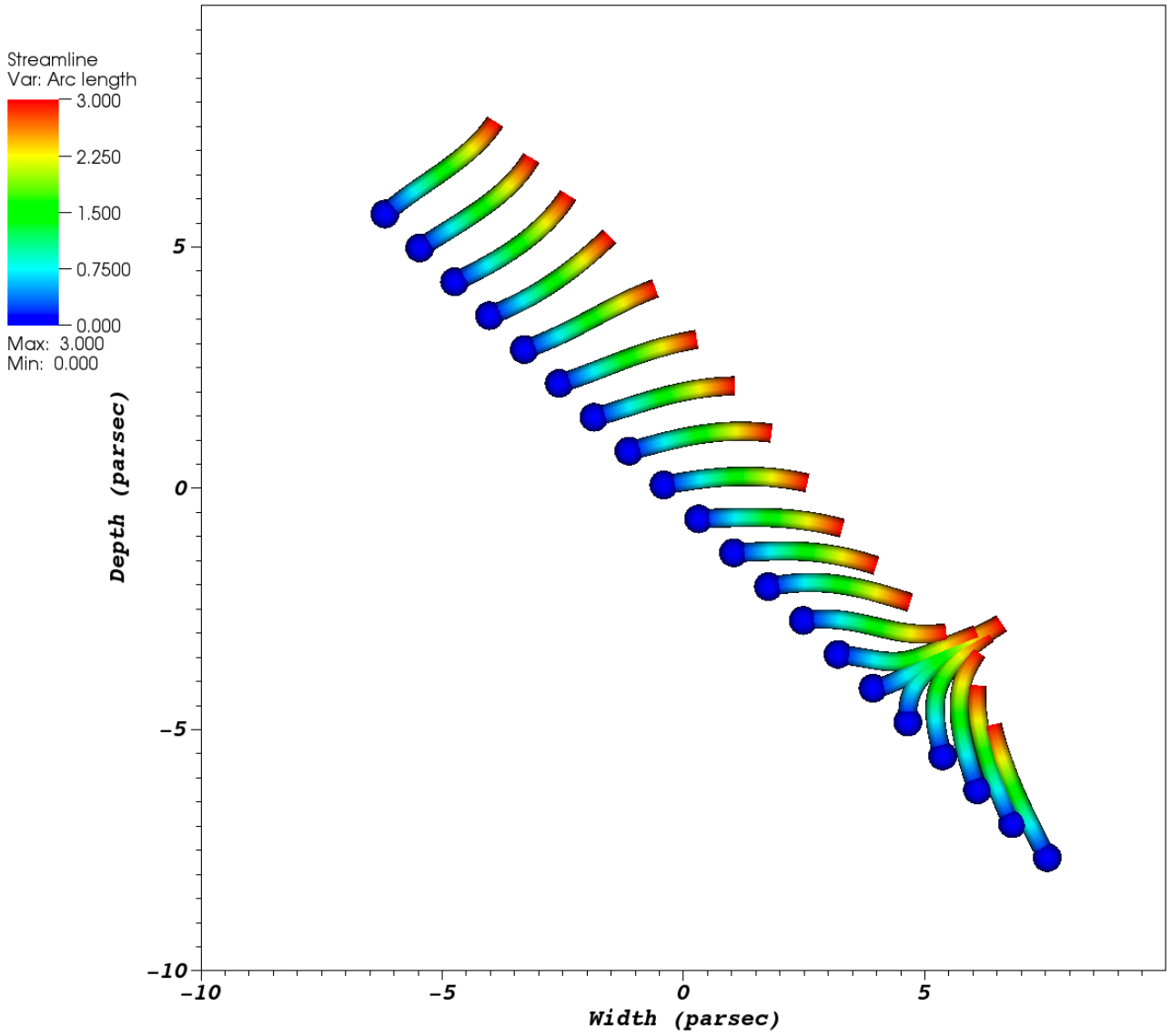
Termination criteria

- Time



Termination criteria

- Time
- Distance

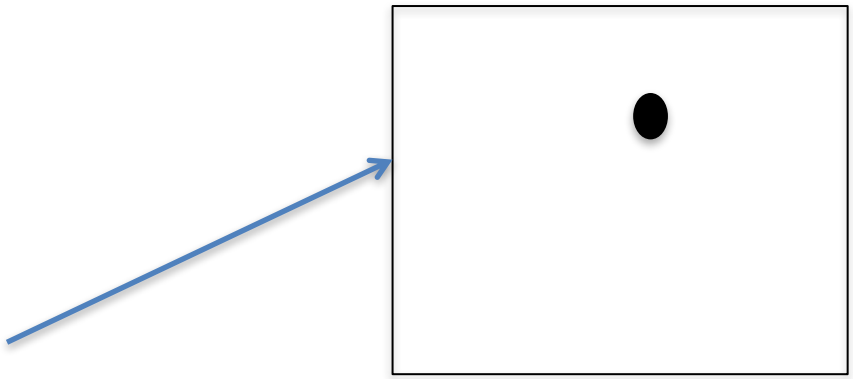


Termination criteria

- Time
- Distance
- Number of steps
  - Same as time?
- Other advanced criteria, based on particle advection purpose

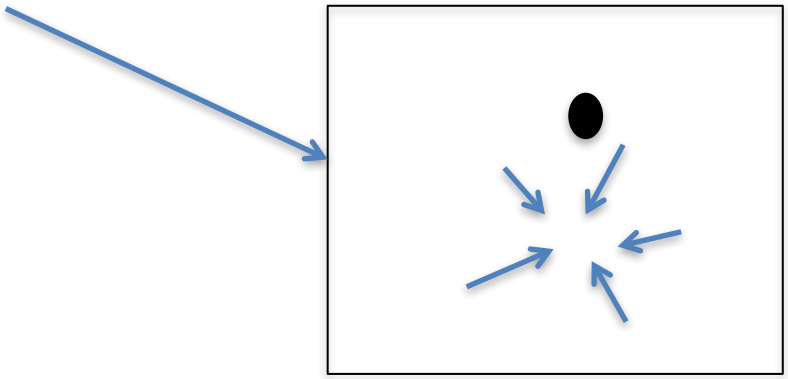
Other reasons for advection termination

- Exit the volume



Other reasons for advection termination

- Advect into a sink



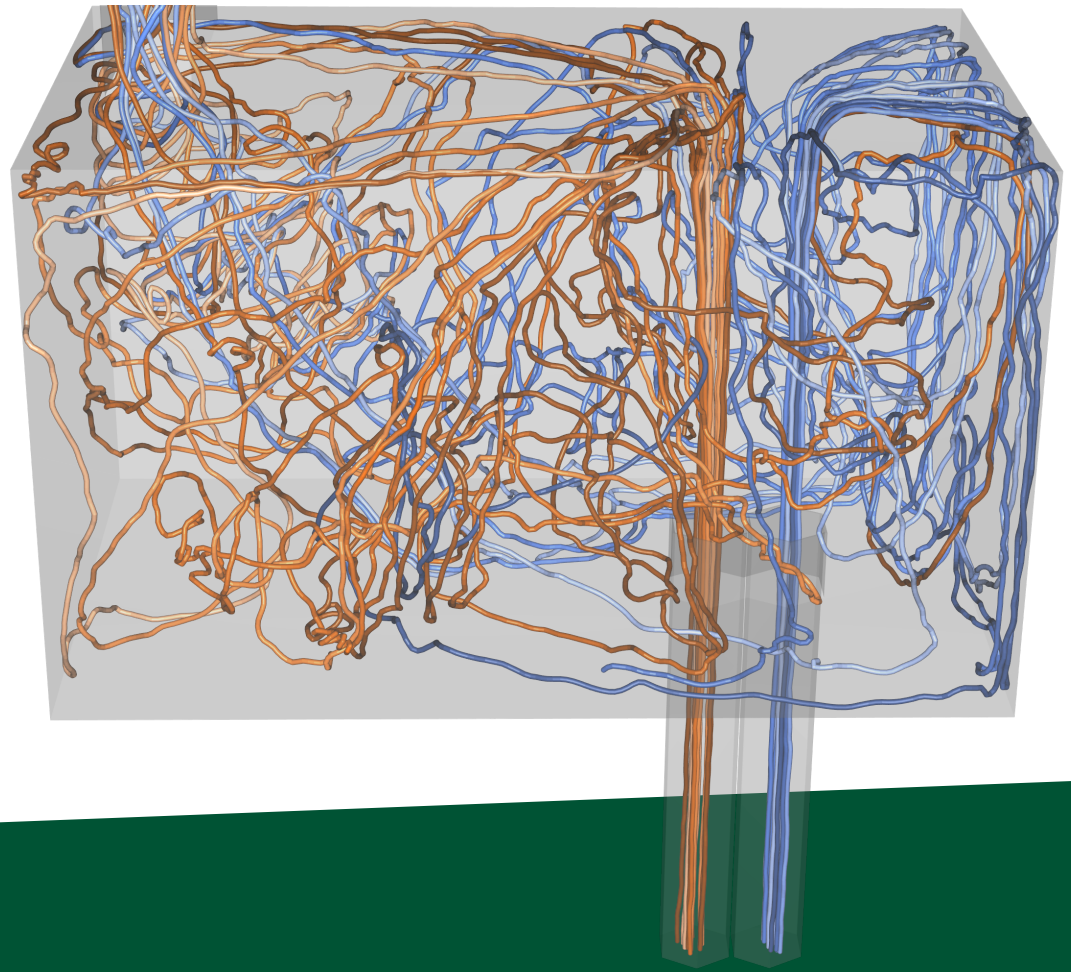
Steady Versus Unsteady State

- Unsteady state: the velocity field evolves over time
- Steady state: the velocity field has reached steady state and remains unchanged as time evolves

Most Common Particle Advection Technique: Streamlines / Pathlines

- Idea: plot the entire trajectory of the particle all at one time

Streamlines in the “fish tank”

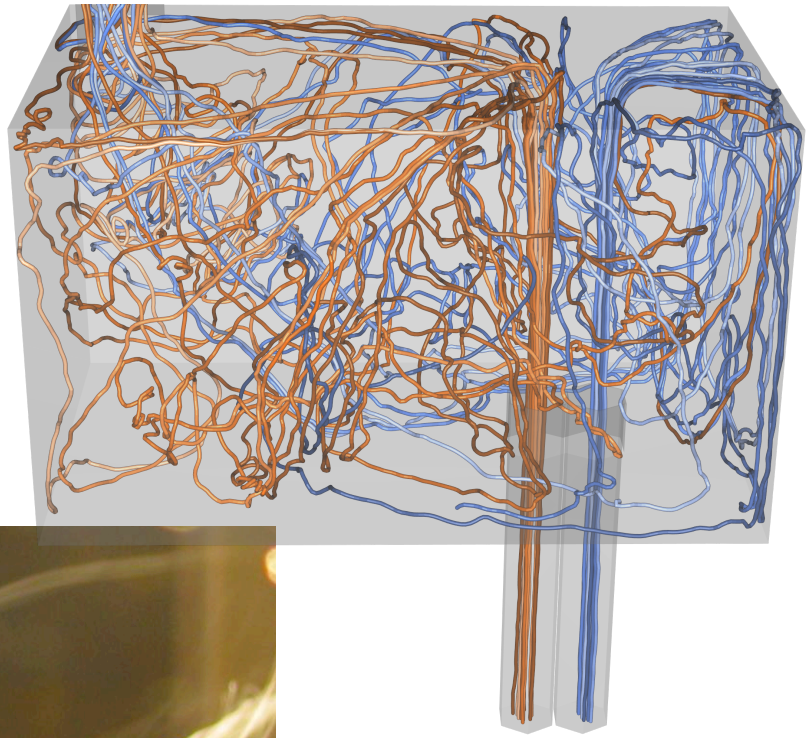


Streamline vs Pathlines

- Streamlines: plot trajectory of a particle from a steady state field
- Pathlines: plot trajectory of a particle from an unsteady state field

- Quiz: most common configuration?
  - Neither!!
  - Pretend an unsteady state field is actually steady state and plot streamlines from one moment in time.
- Quiz: why would anyone want to do this?
  - (answer: performance)

Pathlines in the real world



Lots more to talk about

- How do we pragmatically deal with unsteady state flow (velocities that change over time)?
- More visualization algorithms based on particle advection
- Stability of results

Dealing with Steady State Velocities

- Euler Method (unsteady):
  - New position: $p_{i+1} = p_i + h*v(t_i, p_i)$
  - New time: $t_{i+1} = t_i + h$
- Euler Method (steady):
  - New position: $p_{i+1} = p_i + h*v(t_0, p_i)$
  - New time: $t_{i+1} = t_i + h$

Unsteady vs Steady: RK4

- Unsteady:
  - New position: $p_{i+1} = p_i+(1/6)*h*(k_1+2k_2+2k_3+k_4)$
    - $k_1 = v(t_i, p_i)$
    - $k_2 = v(t_i + h/2,\ \ p_i + h/2*k_1)$
    - $k_3 = v(t_i + h/2,\ \ p_i + h/2*k_2)$
    - $k_4 = v(t_i + h,\ \ p_i + h*k_3)$
    - New time: $t_{i+1} = t_i+h$

Unsteady vs Steady: RK4

- Steady:
  - New position: $p_{i+1} = p_i+(1/6)*h*(k_1+2k_2+2k_3+k_4)$
    - $k_1 = v(t_0, p_i)$
    - $k_2 = v(t_0, \ p_i + h/2*k_1)$
    - $k_3 = v(t_0, \ p_i + h/2*k_2)$
    - $k_4 = v(t_0, \ p_i + h*k_3)$
    - New time: $t_{i+1} = t_i+h$