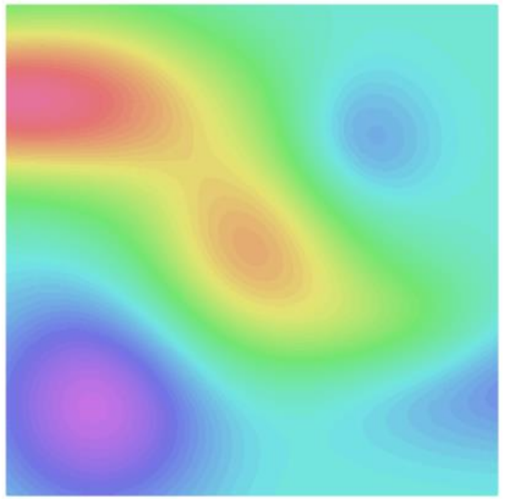
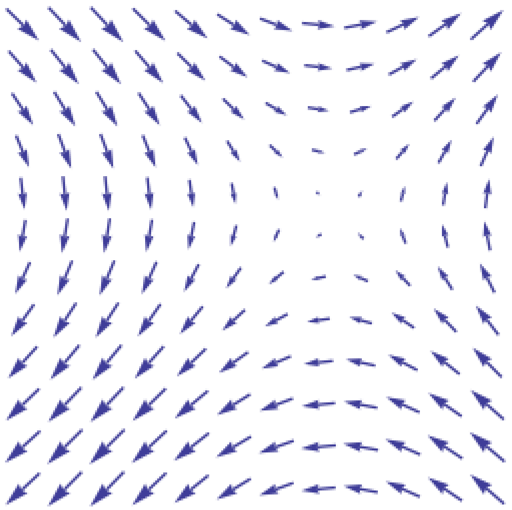
Differential Vector Calculus

Steve Rotenberg
CSE291: Physics Simulation
UCSD
Spring 2019

Fields

- A *field* is a function of position **x** and may vary over time *t*
- A *scalar field* such as *s*(**x**,*t*) assigns a scalar value to every point in space. An example of a scalar field would be the temperature throughout a room
- A *vector field* such as **v**(**x**,*t*) assigns a vector to every point in space. An example of a vector field would be the velocity of the air





Del Symbol

- The *Del* symbol ∇ is useful for defining several types of *spatial derivatives* of fields

$$\nabla = \left[\frac{\partial}{\partial x} \quad \frac{\partial}{\partial y} \quad \frac{\partial}{\partial z}\right]^T$$

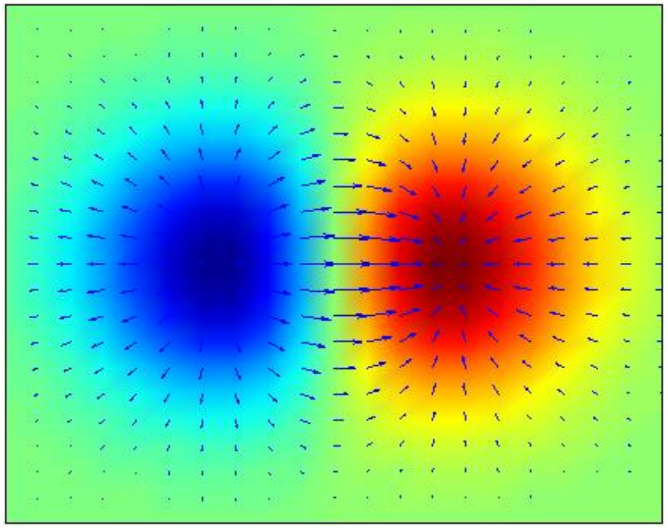
- Technically, ∇ by itself is neither a vector nor an operator, although it acts like both. It is used to define the gradient ∇, divergence $\nabla\ \cdot$, curl $\nabla\ \times$, and Laplacian ∇^2 operators

Gradient

- The *gradient* is a generalization of the concept of a derivative

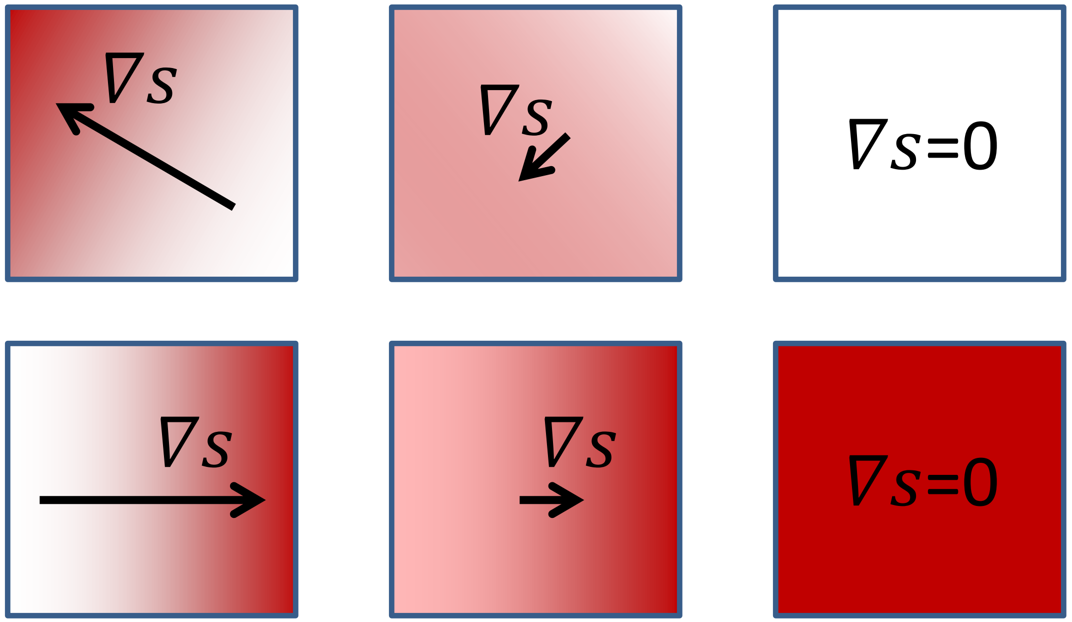
$$\nabla s = \begin{bmatrix} \frac{\partial s}{\partial x} & \frac{\partial s}{\partial y} & \frac{\partial s}{\partial z} \end{bmatrix}^T$$

- When applied to a *scalar* field, the result is a *vector* pointing in the direction the field is increasing and the magnitude indicates the rate of increase
- In 1D, this reduces to the standard derivative (slope)



Gradient

- The gradient ∇s is a vector that points "uphill" in the direction that scalar field s is increasing
- The magnitude of ∇s is equal to the rate that s is increasing per unit of distance



Divergence

- The *divergence* of a vector field is a scalar measure of how much the vectors are expanding

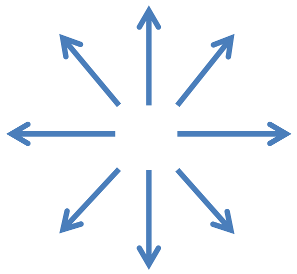
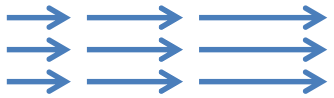
$$\nabla \cdot \mathbf{v} = \frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} + \frac{\partial v_z}{\partial z}$$

- For example, when air is heated in a region, it will locally expand, causing a positive divergence in the region of expansion
- The divergence operator works on a vector field and produces a scalar field as a result

Divergence

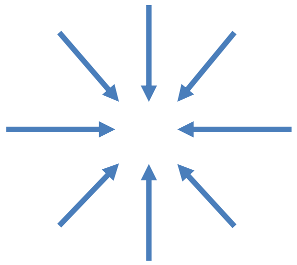
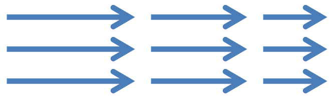
- The divergence is positive where the field is expanding:

$$\nabla \cdot \mathbf{v} > 0$$

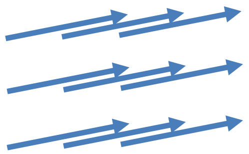
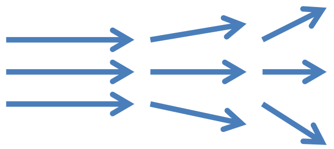
- The divergence is negative where the field is contracting:

$$\nabla \cdot \mathbf{v} < 0$$

- A constant field has zero divergence, as can many others:

$$\nabla \cdot \mathbf{v} = 0$$

Curl

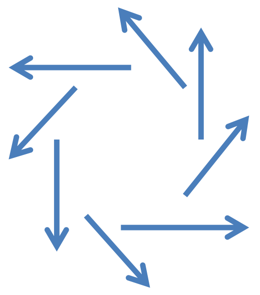
- The *curl* operator produces a new vector field that measures the rotation of the original vector field

$$\nabla \times \mathbf{v} = \begin{bmatrix} \frac{\partial v_z}{\partial y} - \frac{\partial v_y}{\partial z} & \frac{\partial v_x}{\partial z} - \frac{\partial v_z}{\partial x} & \frac{\partial v_y}{\partial x} - \frac{\partial v_x}{\partial y} \end{bmatrix}^T$$

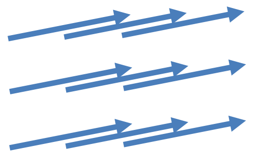
- For example, if the air is circulating in a particular region, then the curl in that region will represent the axis of rotation
- The magnitude of the curl is twice the angular velocity of the vector field

Curl

- A counter-clockwise rotating field has a curl vector pointing out of the screen towards the viewer, perpendicular to the rotation plane



- A constant vector field has zero curl: $\nabla \times \mathbf{v} = [0 \quad 0 \quad 0]^T$



Laplacian

- The *Laplacian* operator is one type of second derivative of a scalar or vector field

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

- Just as in 1D where the second derivative relates to the curvature of a function, the Laplacian relates to the curvature of a field
- The Laplacian of a scalar field is another scalar field:

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

- And the Laplacian of a vector field is another vector field

$$\nabla^2 \mathbf{v} = \frac{\partial^2 \mathbf{v}}{\partial x^2} + \frac{\partial^2 \mathbf{v}}{\partial y^2} + \frac{\partial^2 \mathbf{v}}{\partial z^2}$$

Laplacian

- The Laplacian is positive in an area of the field that is surrounded by higher values

- The Laplacian is negative where the field is surrounded by lower values

- The Laplacian is zero where the field is either flat, linear sloped, or the positive and negative curvatures cancel out (saddle points)

Del Operations

- Del: $\nabla = \begin{bmatrix} \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \end{bmatrix}^T$
- Gradient: $\nabla s = \begin{bmatrix} \frac{\partial s}{\partial x} & \frac{\partial s}{\partial y} & \frac{\partial s}{\partial z} \end{bmatrix}^T$
- Divergence: $\nabla \cdot \mathbf{v} = \frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} + \frac{\partial v_z}{\partial z}$
- Curl: $\nabla \times \mathbf{v} = \begin{bmatrix} \frac{\partial v_z}{\partial y} - \frac{\partial v_y}{\partial z} & \frac{\partial v_x}{\partial z} - \frac{\partial v_z}{\partial x} & \frac{\partial v_y}{\partial x} - \frac{\partial v_x}{\partial y} \end{bmatrix}^T$
- Laplacian: $\nabla^2 s = \frac{\partial^2 s}{\partial x^2} + \frac{\partial^2 s}{\partial y^2} + \frac{\partial^2 s}{\partial z^2}$

Numerical Representation of Fields

Computational Vector Calculus

- Now that we’ve seen the basic operations of differential vector calculus, we turn to the issue of computer implementation
- The Del operations are defined in terms of general fields
- We must address the issue of how we represent fields on the computer and how we perform calculus operations on them

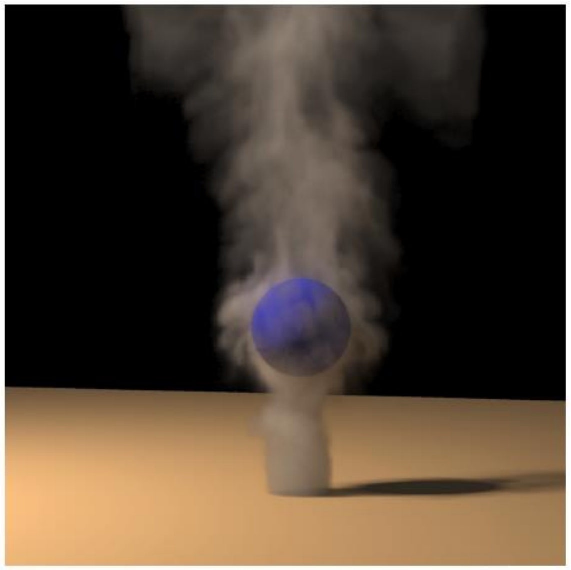
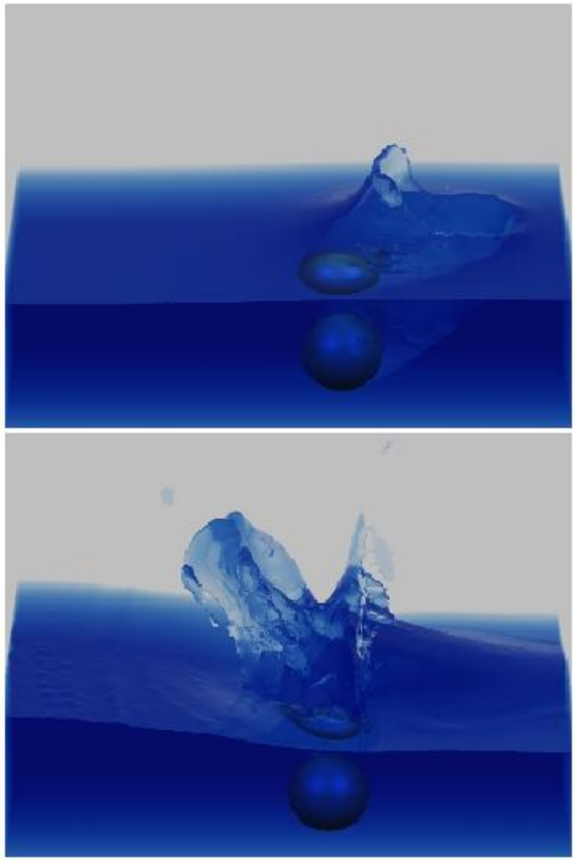
Numerical Representation of Fields

- Mathematically, a scalar or vector field represents a continuously variable value across space that can have infinite detail
- Obviously, on the computer, we can’t truly represent the value of the field everywhere to this level, so we must use some form of approximation
- A standard approach to representing a continuous field is to sample it at some number of discrete points and use some form of interpolation to get the value between the points
- There are several choices of how to arrange our samples:
  - Uniform grid
  - Hierarchical grid
  - Irregular mesh
  - Particle based

Uniform Grids

- *Uniform grids* are easy to deal with and tend to be computationally efficient due to their simplicity
- It is very straightforward to compute derivatives on uniform grids
- However, they require large amounts of memory to represent large domains
- They don’t adapt well to varying levels of detail, as they represent the field to an even level of detail everywhere

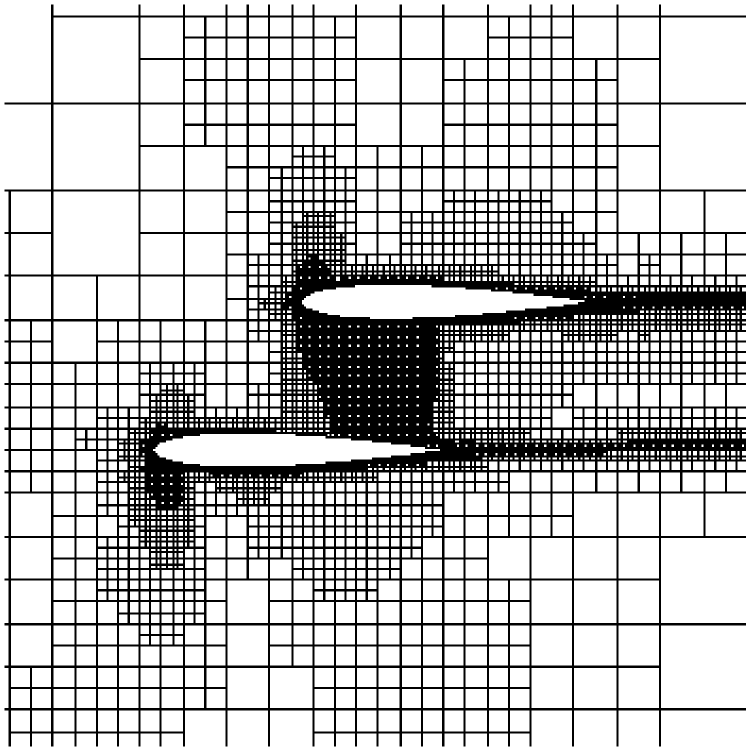
Uniform Grids

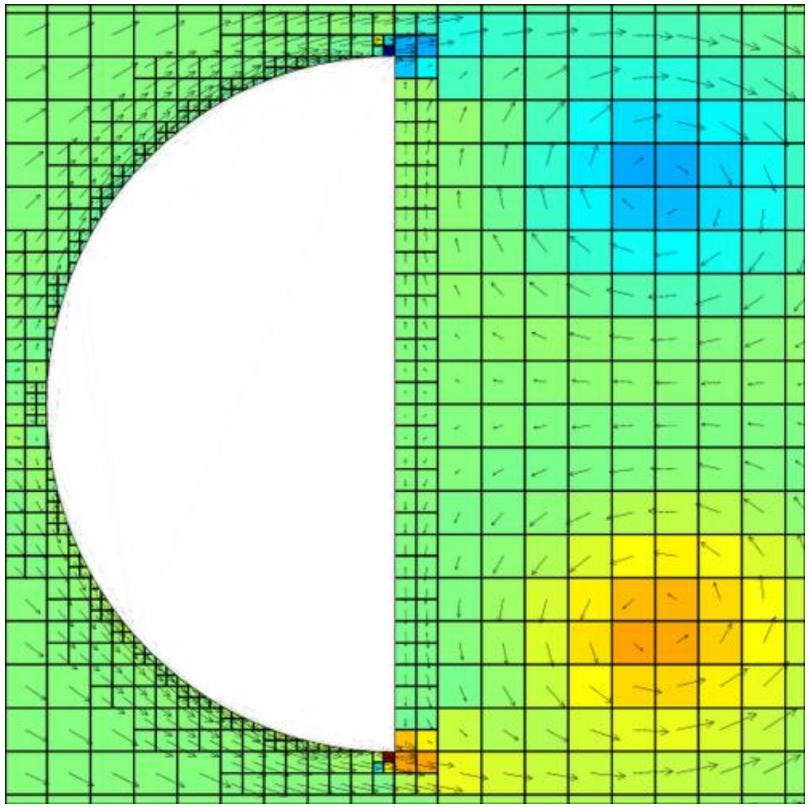



Hierarchical Grids

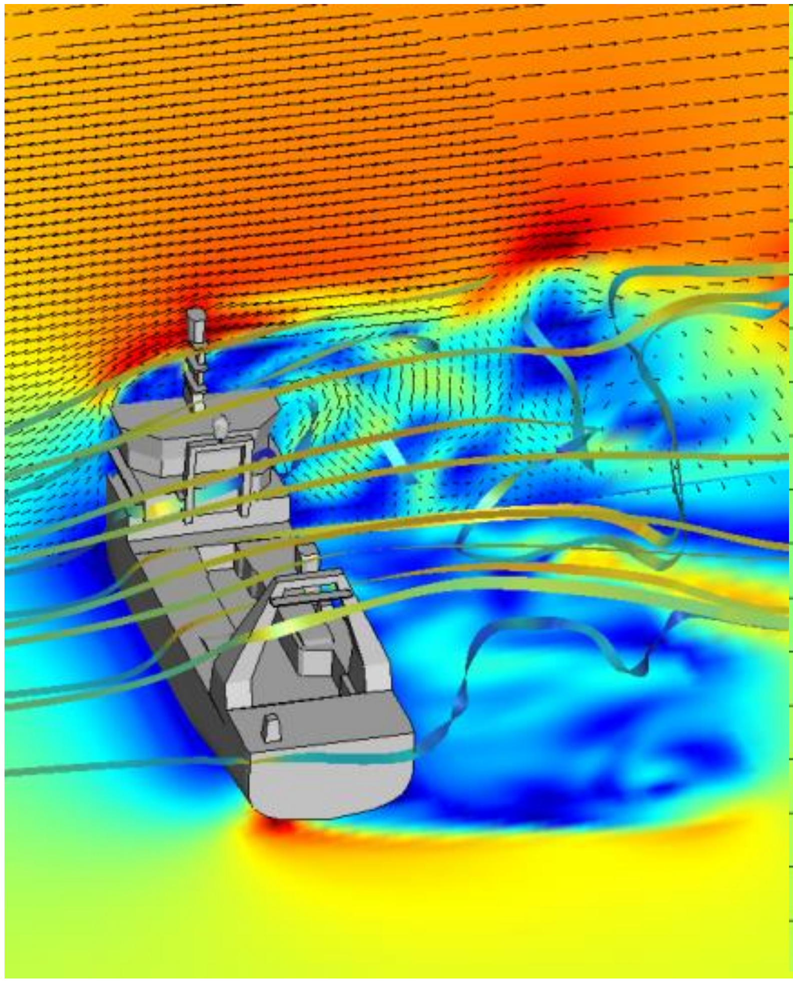
- *Hierarchical grids* such as quadtrees and octrees attempt to benefit from the simplicity of uniform grids, but also have the additional benefit of scaling well to large problems and varying levels of detail
- The grid resolution can locally increase to handle more detail in regions that require it
- This allows both memory and compute time to be used efficiently and adapt automatically to the problem complexity

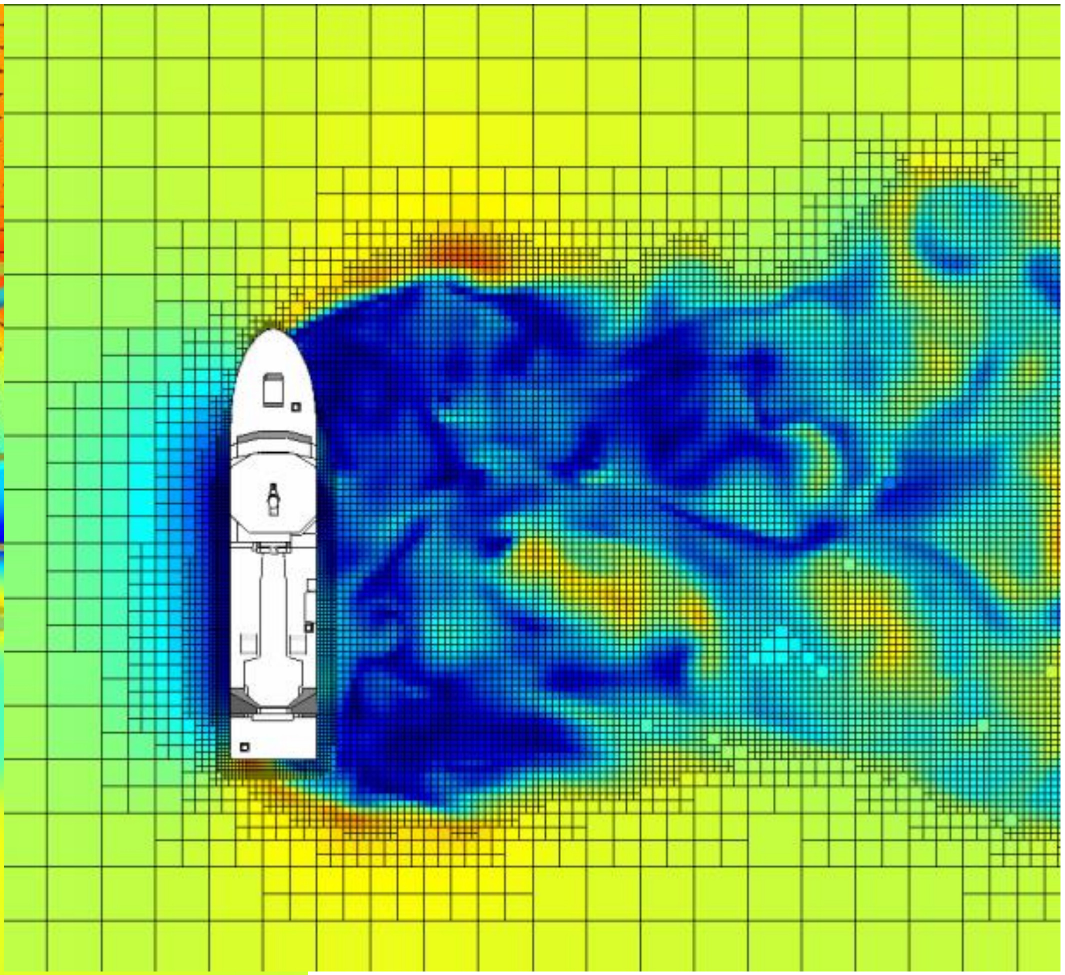
Hierarchical Grids





Hierarchical Grids

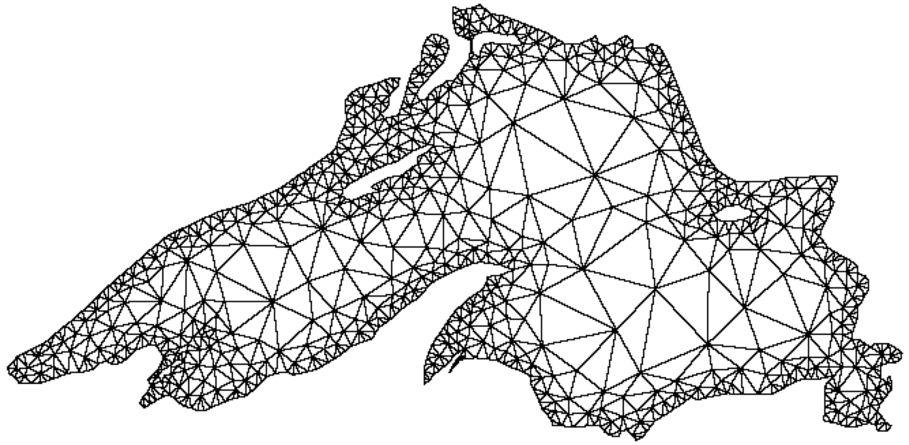
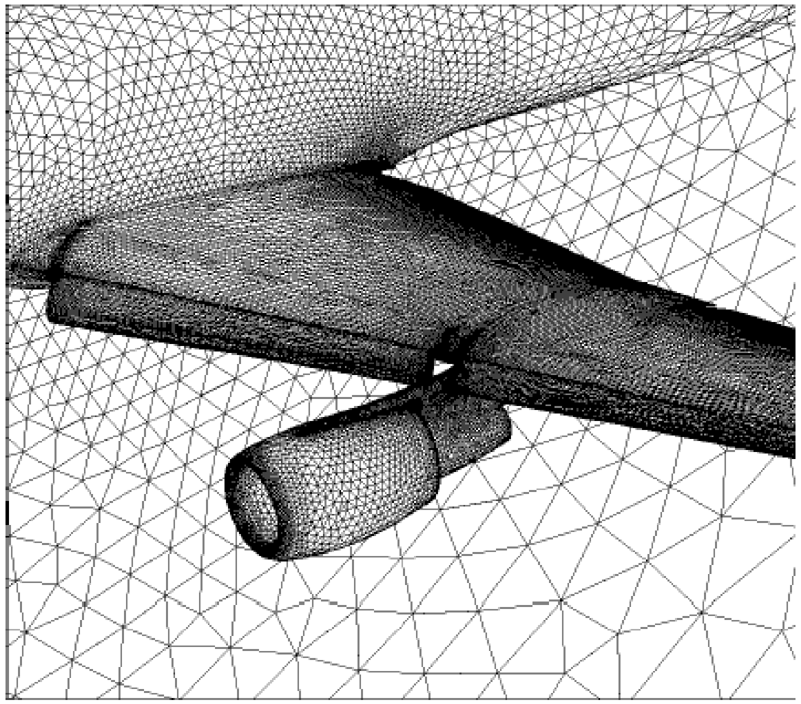
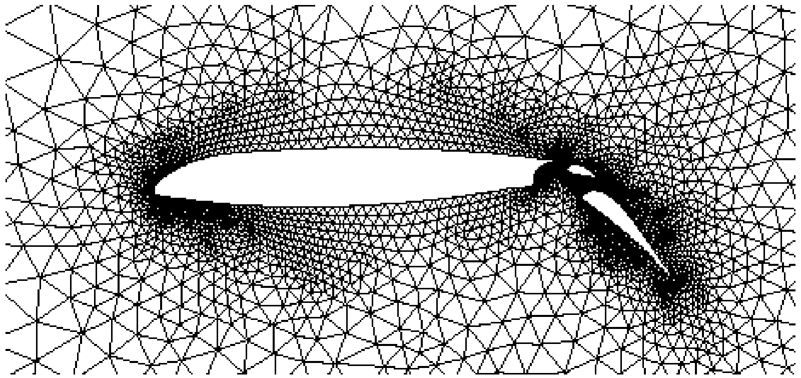
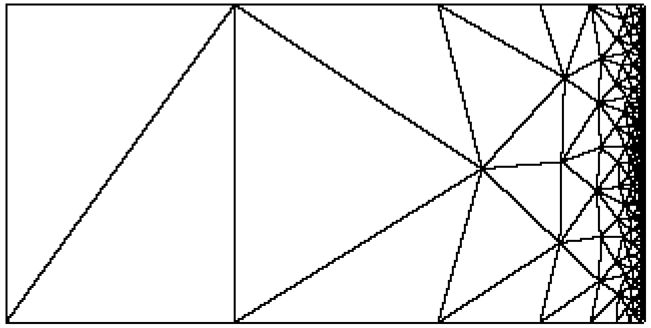




Irregular Meshes

- *Irregular meshes* are built from primitive cells (usually triangles in 2D and tetrahedra in 3D)
- Irregular meshes are used extensively in engineering applications, but less so in computer animation
- One of the main benefits of irregular meshes is their ability to adapt to complex domain geometry
- They also adapt well to varying levels of detail
- They can be quite complex to generate however and can have a lot of computational overhead in highly dynamic situations with moving objects
- If the irregular mesh changes over time to adapt to the problem complexity, it is called an *adaptive mesh*

Irregular Mesh

Adaptive Meshes

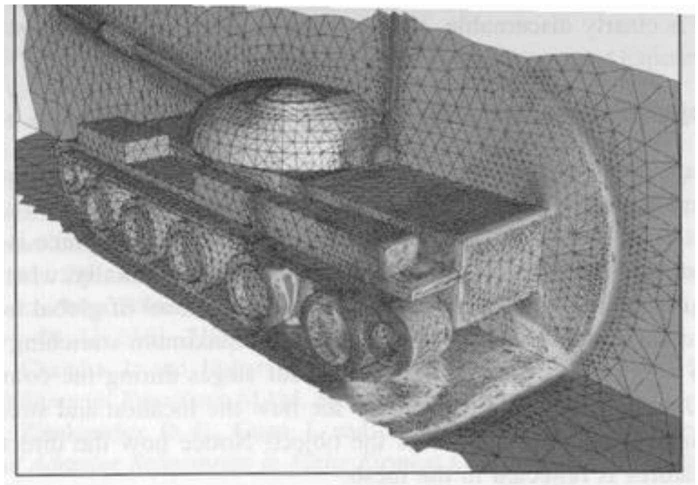
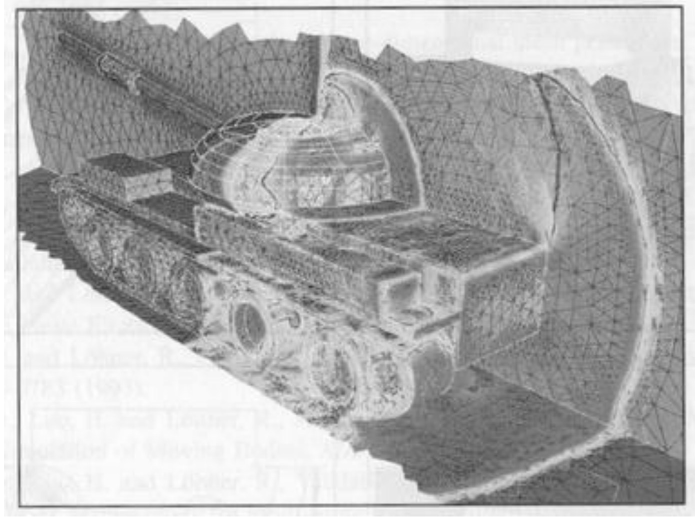
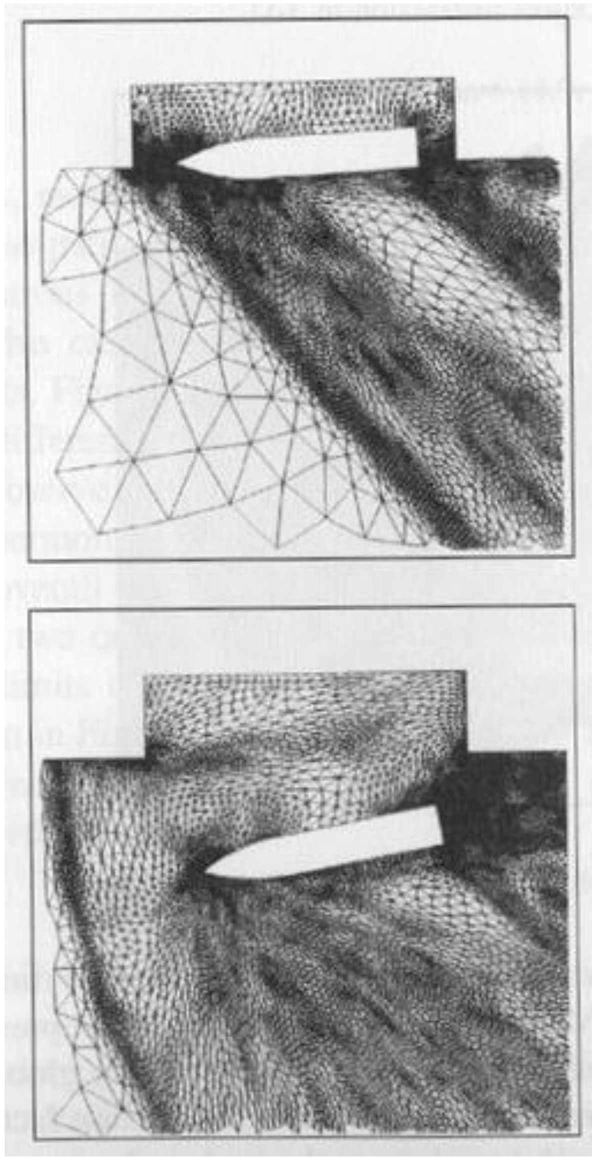


Figure 14.10a Shock-object interaction in 3-D

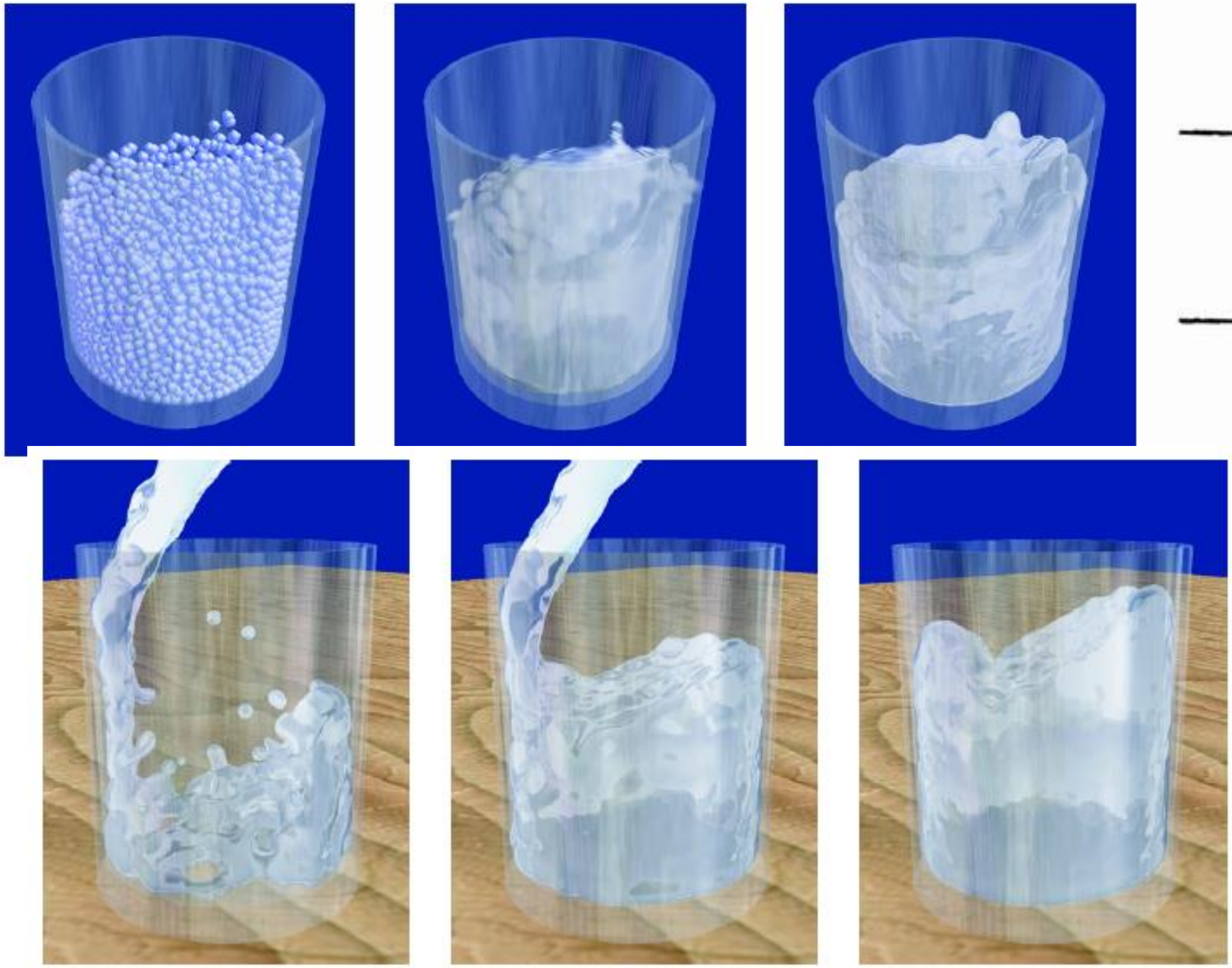




Particle-Based (Meshless)

- Instead of using a mesh with well defined connectivity, particle methods sample the field on a set of irregularly distributed particles
- Particles aren't meant to be 0 dimensional points- they are assumed to represent a small 'smear' of the field, over some radius, and the value of the field at any point is determined by several nearby particles
- Calculating derivatives can be tricky and there are several approaches
- Particle methods are very well suited to water and liquid simulation for a variety of reasons and have been gaining a lot of popularity in the computer graphics industry recently

Particle Based



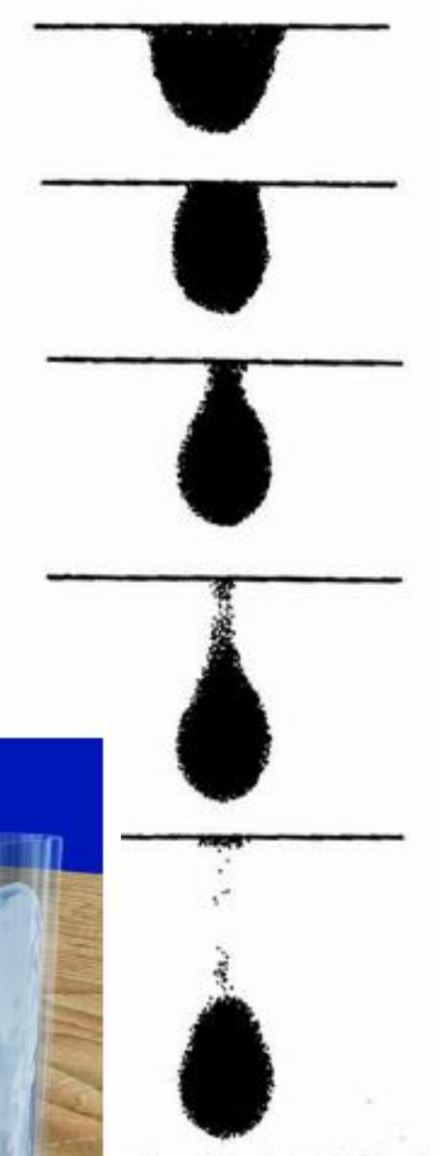
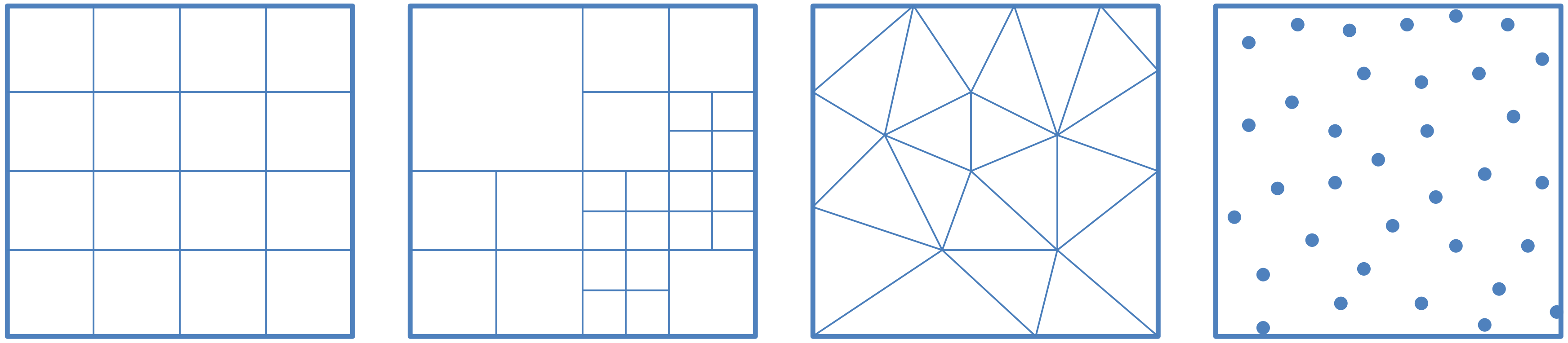


Figure 9.2 — Drop fall.

Field Representations

- Each method uses its own way of sampling the field at some interval
- Each method requires a way to interpolate the field between sample points
- Each method requires a way to compute the different spatial derivatives $(\nabla, \nabla\cdot, \nabla\times, \nabla^2)$



Derivative Computation

Uniform Grids & Finite Differencing

- For today, we will just consider the case of uniform grid
- A scalar field is represented as a 2D/3D array of floats and a vector field is a 2D/3D array of vectors
- We will use a technique called *finite differencing* to compute derivatives of the fields

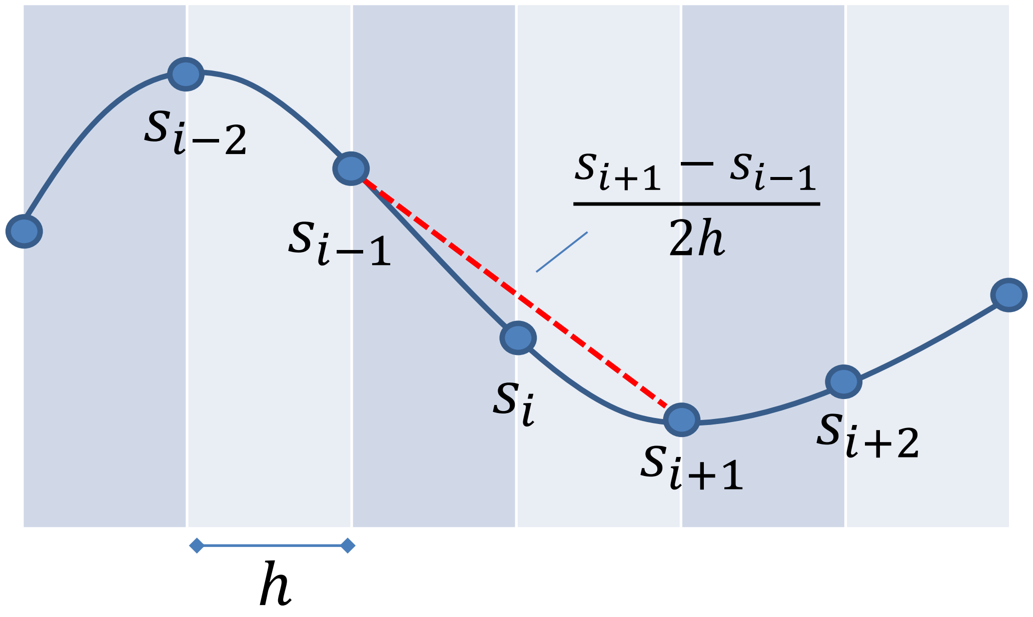
Finite Difference First Derivative

- The derivative (slope) of a 1D function $s(x)$ stored uniformly spaced at values of x_i can be approximated by finite differencing:

$$\frac{ds}{dx}(x_i) \approx \frac{\Delta s}{\Delta x}(x_i) = \frac{s_{i+1} - s_{i-1}}{2h}$$

- Where h is the grid size ($h = x_{i+1} - x_i$)

Finite Difference First Derivative



$$\frac{ds}{dx}(x_i) \approx \frac{\Delta s}{\Delta x}(x_i) = \frac{s_{i+1} - s_{i-1}}{2h}$$

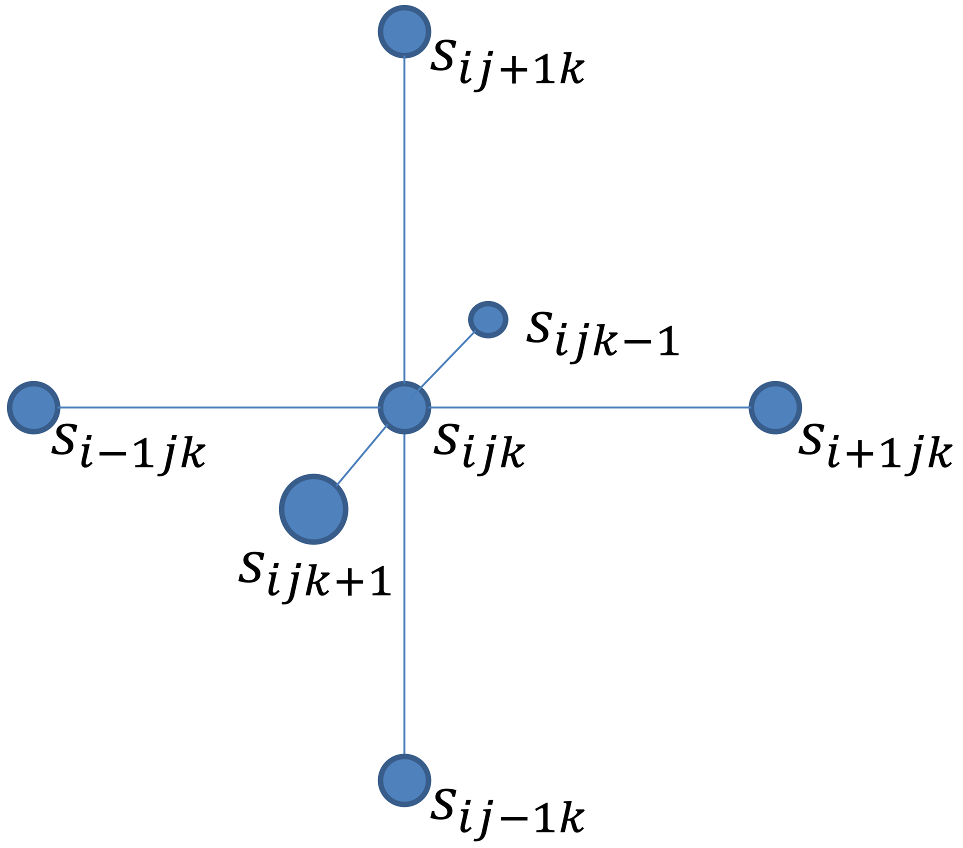
Finite Difference Partial Derivatives

- If we have a scalar field $s(\mathbf{x}, t)$ stored on a uniform 3D grid, we can approximate the partial derivative along the *x* direction at grid cell ijk as:

$$\frac{\partial s}{\partial x}\left(\mathbf{x}_{ijk}\right) \approx \frac{\Delta s}{\Delta x}\left(\mathbf{x}_{ijk}\right) = \frac{s_{i+1jk} - s_{i-1jk}}{2h}$$

- Where cell $i + 1jk$ is the cell in the +*x* direction and cell $i - 1jk$ is in the –*x* direction
- Also h is the cell size in the *x* direction
- The partials along *y* and *z* are done in the same fashion
- All of the partial derivatives in the gradient, divergence, and curl can be computed in this way

Neighboring Grid Points



Finite Difference Gradient

- We can compute the finite difference gradient ∇s at grid point ijk from s values at neighboring grid points

$$\nabla s(\mathbf{x}_{ijk}) = \begin{bmatrix} \frac{\partial s}{\partial x} & \frac{\partial s}{\partial y} & \frac{\partial s}{\partial z} \end{bmatrix}^T$$

$$\approx \begin{bmatrix} \frac{\Delta s}{\Delta x} & \frac{\Delta s}{\Delta y} & \frac{\Delta s}{\Delta z} \end{bmatrix}^T$$

$$= \begin{bmatrix} \frac{s_{i+1jk} - s_{i-1jk}}{2h} & \frac{s_{ij+1k} - s_{ij-1k}}{2h} & \frac{s_{ijk+1} - s_{ijk-1}}{2h} \end{bmatrix}^T$$

$$= \frac{1}{2h} \begin{bmatrix} s_{i+1jk} - s_{i-1jk} \\ s_{ij+1k} - s_{ij-1k} \\ s_{ijk+1} - s_{ijk-1} \end{bmatrix}$$

Finite Difference Divergence

- We can compute a finite difference of the divergence at grid point *ijk* in a similar fashion:

$$\nabla \cdot \mathbf{v}(\mathbf{x}_{ijk}) = \frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} + \frac{\partial v_z}{\partial z}$$

$$\approx \frac{\Delta v_x}{\Delta x} + \frac{\Delta v_y}{\Delta y} + \frac{\Delta v_z}{\Delta z}$$

$$= \frac{v_{x_{i+1jk}} - v_{x_{i-1jk}}}{2h} + \frac{v_{y_{ij+1k}} - v_{y_{ij-1k}}}{2h} + \frac{v_{z_{ijk+1}} - v_{z_{ijk-1}}}{2h}$$

$$= \frac{1}{2h}\left(v_{x_{i+1jk}} - v_{x_{i-1jk}} + v_{y_{ij+1k}} - v_{y_{ij-1k}} + v_{z_{ijk+1}} - v_{z_{ijk-1}}\right)$$

Finite Difference Curl

- For the finite difference curl at grid point *ijk* we have:

$$\nabla \times \mathbf{v}(\mathbf{x}_{ijk}) = \begin{bmatrix} \frac{\partial v_z}{\partial y} - \frac{\partial v_y}{\partial z} & \frac{\partial v_x}{\partial z} - \frac{\partial v_z}{\partial x} & \frac{\partial v_y}{\partial x} - \frac{\partial v_x}{\partial y} \end{bmatrix}^T$$

$$\approx \begin{bmatrix} \frac{\Delta v_z}{\Delta y} - \frac{\Delta v_y}{\Delta z} & \frac{\Delta v_x}{\Delta z} - \frac{\Delta v_z}{\Delta x} & \frac{\Delta v_y}{\Delta x} - \frac{\Delta v_x}{\Delta y} \end{bmatrix}^T$$

$$= \frac{1}{2h} \begin{bmatrix} \left(v_{z_{ij+1k}} - v_{z_{ij-1k}}\right) - \left(v_{y_{ijk+1}} - v_{y_{ijk-1}}\right) \\ \left(v_{x_{ijk+1}} - v_{x_{ijk-1}}\right) - \left(v_{z_{i+1jk}} - v_{z_{i-1jk}}\right) \\ \left(v_{y_{i+1jk}} - v_{y_{i-1jk}}\right) - \left(v_{x_{ij+1k}} - v_{x_{ij-1k}}\right) \end{bmatrix}$$

Finite Difference Second Derivative

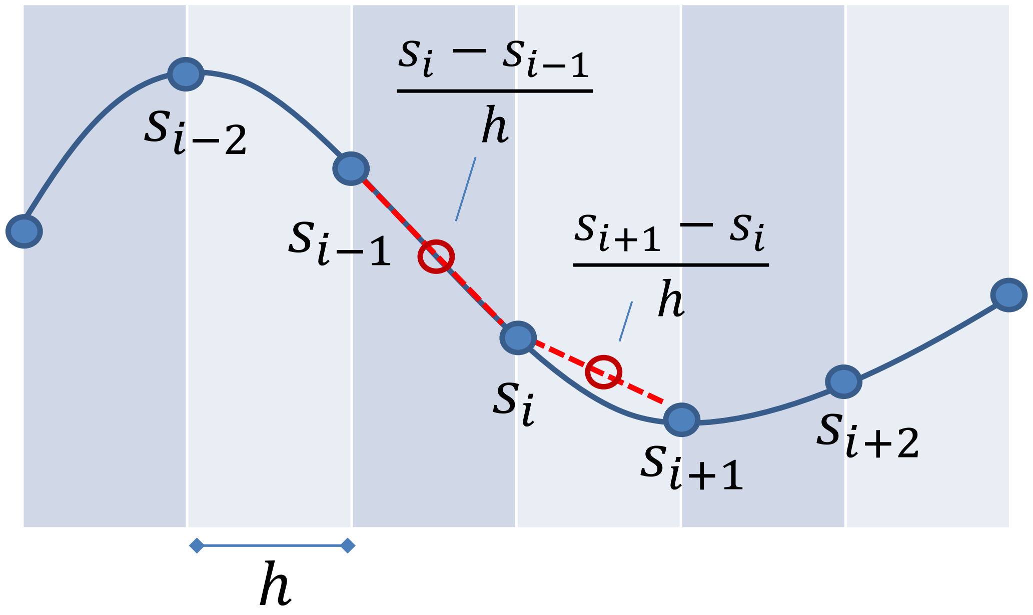
- The second derivative can be approximated by finite differencing in a similar way:

$$\frac{d^2 s}{dx^2}(x_i) \approx \frac{\Delta^2 s}{\Delta x^2} = \frac{\Delta\left(\frac{\Delta s}{\Delta x}\right)}{\Delta x}$$

$$= \frac{\left(\frac{s_{i+1} - s_i}{h}\right) - \left(\frac{s_i - s_{i-1}}{h}\right)}{h}$$

$$= \frac{s_{i+1} - 2s_i + s_{i-1}}{h^2}$$

Finite Difference Second Derivative



$$\frac{d^2s}{dx^2}(x_i) \approx \frac{\Delta\left(\frac{\Delta s}{\Delta x}\right)}{\Delta x}(x_i) = \frac{\left(\frac{s_{i+1} - s_i}{h}\right) - \left(\frac{s_i - s_{i-1}}{h}\right)}{h} = \frac{s_{i+1} - 2s_i + s_{i-1}}{h^2}$$

Finite Difference Laplacian

- The finite difference Laplacian at point *ijk* is:

$$\nabla^2 s(\mathbf{x}_{ijk}) = \frac{\partial^2 s}{\partial x^2} + \frac{\partial^2 s}{\partial y^2} + \frac{\partial^2 s}{\partial z^2}$$

$$\approx \frac{\Delta^2 s}{\Delta x^2} + \frac{\Delta^2 s}{\Delta y^2} + \frac{\Delta^2 s}{\Delta z^2}$$

$$= \frac{s_{i+1jk} - 2s_{ijk} + s_{i-1jk}}{h^2} + \frac{s_{ij+1k} - 2s_{ijk} + s_{ij-1k}}{h^2} + \frac{s_{ijk+1} - 2s_{ijk} + s_{ijk-1}}{h^2}$$

$$= \frac{1}{h^2}\left(s_{i+1jk} + s_{i-1jk} + s_{ij+1k} + s_{ij-1k} + s_{ijk+1} + s_{ijk-1} - 6s_{ijk}\right)$$

Finite Difference Operations

- Gradient: $$\nabla s \approx \frac{1}{2h}\begin{bmatrix} s_{i+1jk} - s_{i-1jk} \\ s_{ij+1k} - s_{ij-1k} \\ s_{ijk+1} - s_{ijk-1} \end{bmatrix}$$
- Divergence: $$\nabla \cdot \mathbf{v} \approx \frac{1}{2h}\left(v_{x_{i+1jk}} - v_{x_{i-1jk}} + v_{y_{ij+1k}} - v_{y_{ij-1k}} + v_{z_{ijk+1}} - v_{z_{ijk-1}}\right)$$
- Curl: $$\nabla \times \mathbf{v} \approx \frac{1}{2h}\begin{bmatrix} \left(v_{z_{ij+1k}} - v_{z_{ij-1k}}\right) - \left(v_{y_{ijk+1}} - v_{y_{ijk-1}}\right) \\ \left(v_{x_{ijk+1}} - v_{x_{ijk-1}}\right) - \left(v_{z_{i+1jk}} - v_{z_{i-1jk}}\right) \\ \left(v_{y_{i+1jk}} - v_{y_{i-1jk}}\right) - \left(v_{x_{ij+1k}} - v_{x_{ij-1k}}\right) \end{bmatrix}$$
- Laplacian: $$\nabla^2 s \approx \frac{1}{h^2}\left(s_{i+1jk} + s_{i-1jk} + s_{ij+1k} + s_{ij-1k} + s_{ijk+1} + s_{ijk-1} - 6s_{ijk}\right)$$
- NOTE: These are based on computing the derivatives at the grid points on a uniform grid

Boundary Conditions

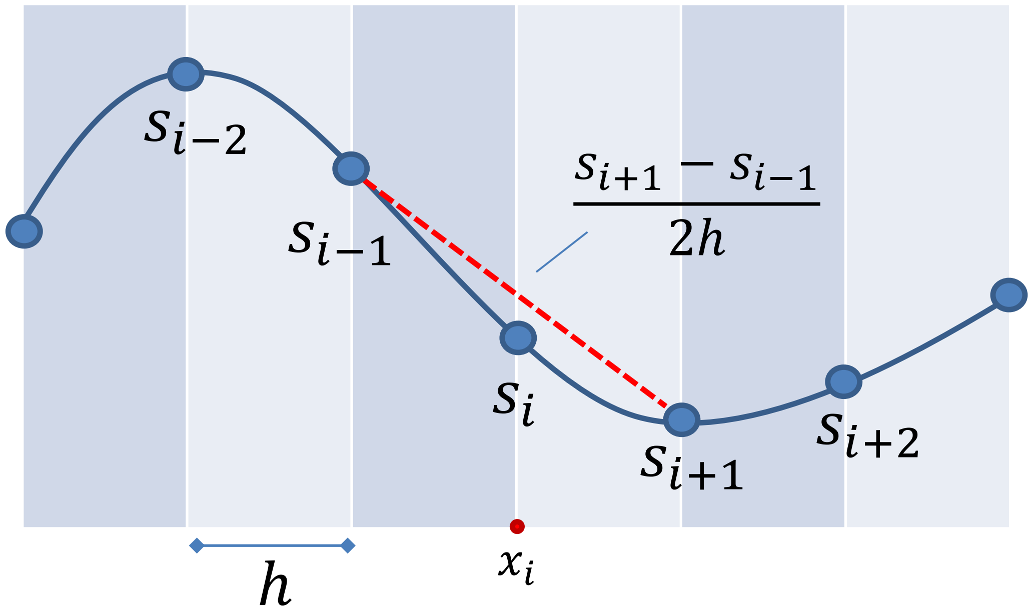
- We saw that computing various spatial derivatives requires using values from neighboring grid points
- What do we do on the boundaries where we might not have neighboring grid points?
- The answer is problem specific, but it falls within the general subject of *boundary conditions*

Boundary Conditions

- There are some options for dealing with derivatives at the boundaries
  - Use directionally biased methods that shift the derivative computation to the right or left by using values to the right or left of the boundary (or up/down…)
  - In some cases, boundary values can be set to known values, such as 0 for the fluid velocity at a solid wall boundary (and 0 for all velocity derivatives)
- We'll talk about some more specifics when we get into fluid dynamics in the next lecture

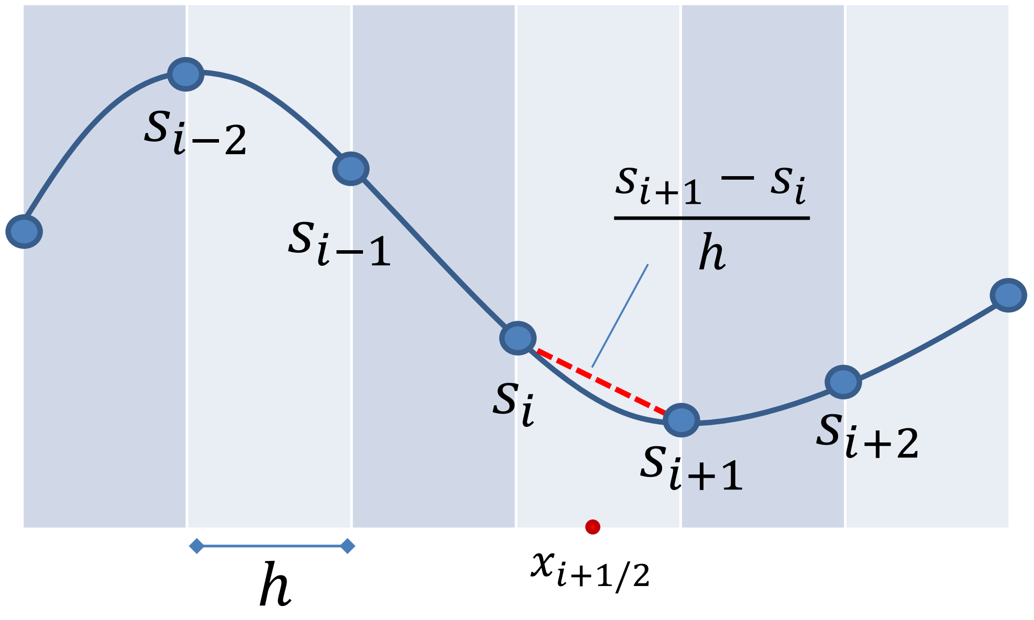
Grid Structures

First Derivative at Grid Point



$$\frac{\partial s}{\partial x}(x_i) \approx \frac{\Delta s}{\Delta x}(x_i) = \frac{s_{i+1} - s_{i-1}}{2h}$$

First Derivative at Midpoint



$$\frac{\partial s}{\partial x}\left(x_{i+1/2}\right) \approx \frac{\Delta s}{\Delta x}\left(x_{i+1/2}\right) = \frac{s_{i+1}-s_i}{h}$$

Midpoint Derivative

- If we want to calculate the derivative at the grid points, we use:

$$\frac{\Delta s}{\Delta x}(x_i) = \frac{s_{i+1} - s_{i-1}}{2h}$$

- If we want to calculate the derivative halfway between grid points, we can use:

$$\frac{\Delta s}{\Delta x}\left(x_{i+1/2}\right) = \frac{s_{i+1} - s_i}{h}$$

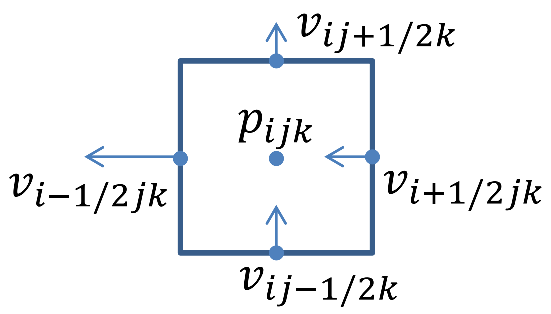
- The second method is usually better because it uses a more localized estimate of the derivative. It also makes use of all nearby data, instead of the first method, which ignores the closest value of the scalar field available
- To make use of this however, one must formulate the equations of interest in a way that is compatible, which tends to be problem-specific

Collocated Grids

- The finite difference derivative computations we looked at so far are based on the assumption that we want to calculate the derivatives at the exact same points that we are storing the field values
- This is known as a *collocated grid*, since all values of interest and their derivatives are collocated at the same points
- However, this leads to the same inaccuracy in computing derivatives that we see in 1D problems

Staggered Grids

- When possible, it is often better to use a *staggered grid*, where certain values are stored at the grid points and other values are stored between points
- In fact, values can be stored at the grid points, on segment edges, on cell faces, or in cell centers
- The 3 values of a 3D vector don't even have to be stored in the same place
- For example, some fluid simulation approaches store the *x*-component of velocity on the *x*-face of each cell, and the *y*-component on the *y*-face, etc. Pressures are computed at the cell centers, based on the velocities through the 6 faces of the cell



Staggered Divergence

- Consider the case where each component of a vector is stored on the corresponding face
- If a cell is indexed as *ijk*, the vectors will be at the halfway values
- We compute the divergence at the center of cell *ijk* as:

$$\nabla \cdot \mathbf{v}(\mathbf{x}_{ijk}) = \frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} + \frac{\partial v_z}{\partial z}$$

$$\approx \frac{\Delta v_x}{\Delta x} + \frac{\Delta v_y}{\Delta y} + \frac{\Delta v_z}{\Delta z}$$

$$= \frac{v_{x_{i+1/2jk}} - v_{x_{i-1/2jk}}}{h} + \frac{v_{y_{ij+1/2k}} - v_{y_{ij-1/2k}}}{h} + \frac{v_{z_{ijk+1/2}} - v_{z_{ijk-1/2}}}{h}$$

$$= \frac{1}{h}\left(v_{x_{i+1/2jk}} - v_{x_{i-1/2jk}} + v_{y_{ij+1/2k}} - v_{y_{ij-1/2k}} + v_{z_{ijk+1/2}} - v_{z_{ijk-1/2}}\right)$$

Higher Order Methods

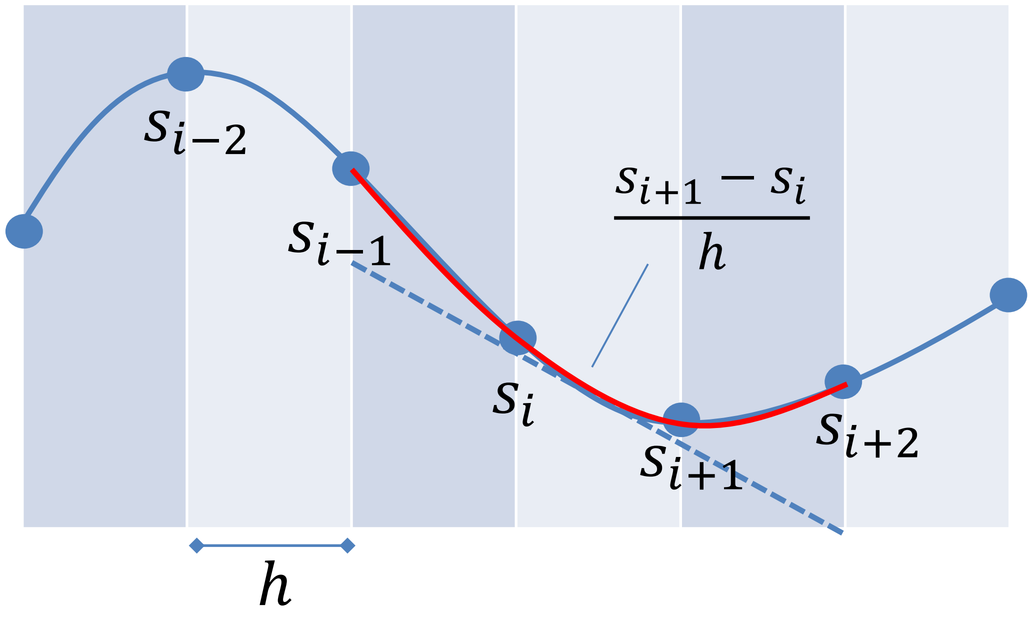
Higher Order Approximations

- We chose to construct our finite derivative operators by using only the nearest neighboring values and using linear (slope) approximations to the derivatives
- We could optionally use more nearby points to get a better approximation to a derivative
- Higher order spatial methods like this can produce better quality and more accurate simulations
- However, they make certain assumptions about smoothness and behave poorly in areas of rapid change
- They also require special handling near the boundaries
- Higher order spatial methods are not too hard to design for uniform grids, but are tricky to derive for other geometries

High Order Derivative

- Consider estimating the derivative on a grid
- With a linear method, we looked at two values and fit a straight line between them and used the slope of that line as the derivative
- With high order methods, we use a weighted blend (usually derived through a Taylor series) at more than two values to compute the slope at the point we're interested in
- These methods are straightforward to implement as they mainly just involve computing a blend of nearby values weighted by pre-specified coefficients

Fourth Order Midpoint Method



$$\frac{ds}{dx}\left(x_{i+1/2}\right) \approx \frac{s_{i-1} - 27s_i + 27s_{i+1} - s_{i+2}}{24h}$$

High Order Derivatives

$$\frac{ds}{dx}(x_i) \approx \frac{-s_{i-1} + s_{i+1}}{2h}$$

$$\frac{ds}{dx}(x_i) \approx \frac{s_{i-2} - 8s_{i-1} + 8s_{i+1} - s_{i+2}}{12h}$$

$$\frac{ds}{dx}(x_i) \approx \frac{-s_{i-3} + 9s_{i-2} - 45s_{i-1} + 45s_{i+1} - 9s_{i+2} + s_{i+3}}{60h}$$