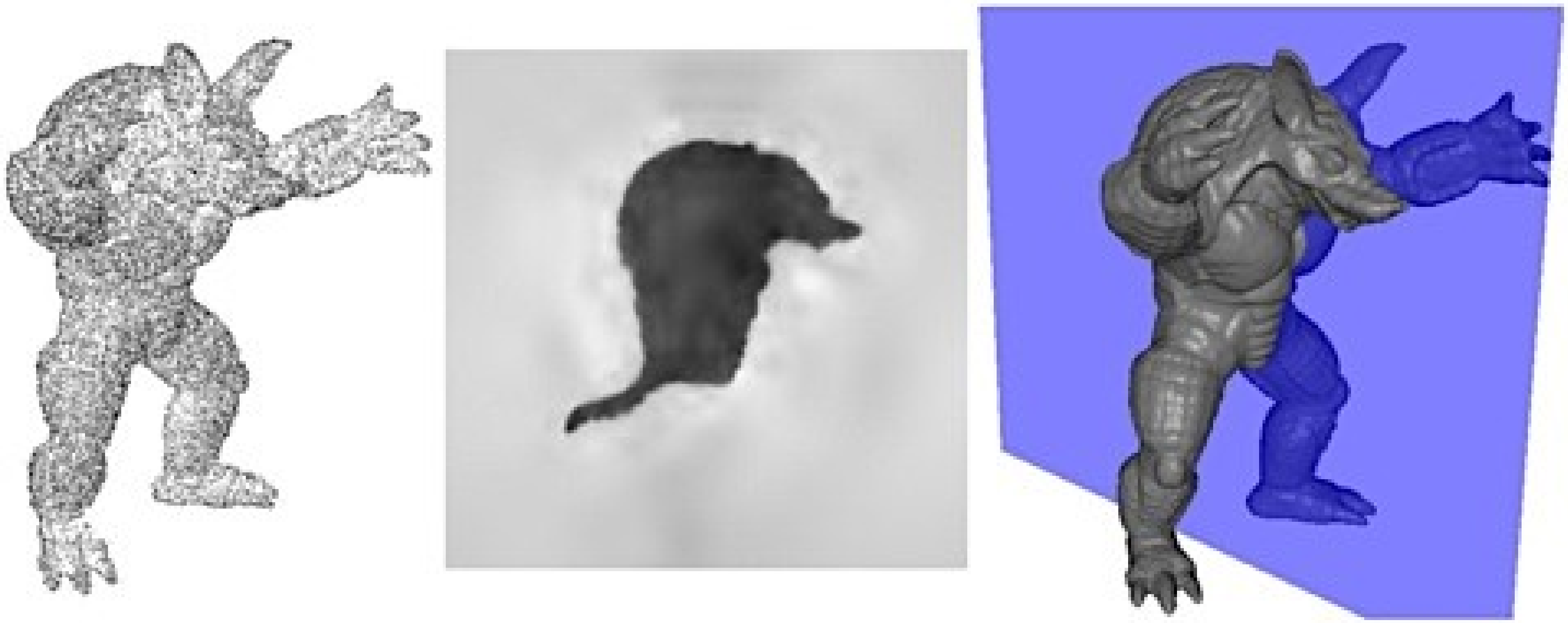
Kazhdan, Bolitho and Hoppe

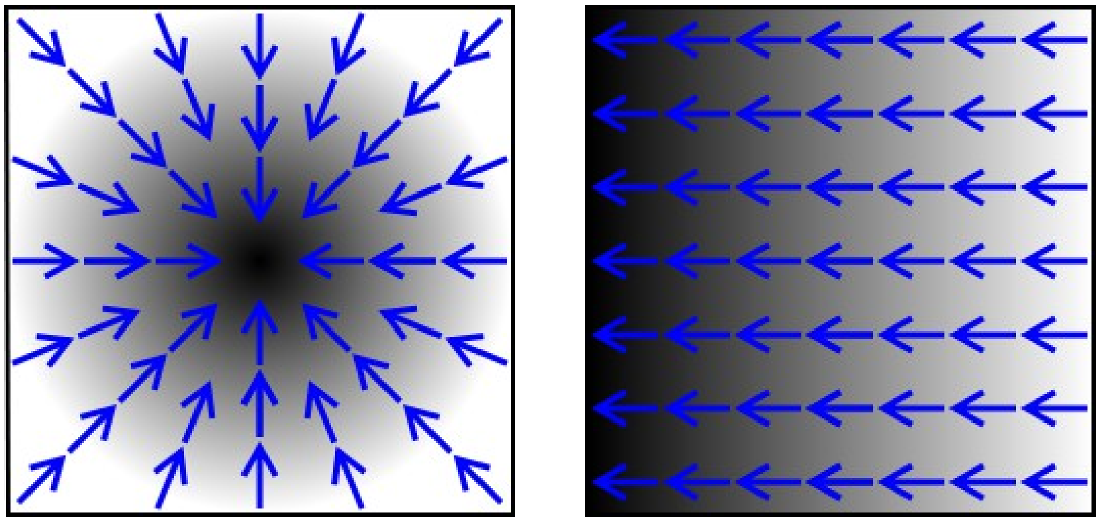


Poisson Surface Reconstruction 3/3

Siddhartha Chaudhuri http://www.cse.iitb.ac.in/~cs749

Recap of differential operators (in 3D)

- **Gradient** (of scalar-valued function): $\nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z} \right)$
  - In operator form: $\nabla = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z} \right)$
  - Maps scalar field to vector field



Scalar fields (black: high, white: low) and their gradients (blue arrows)

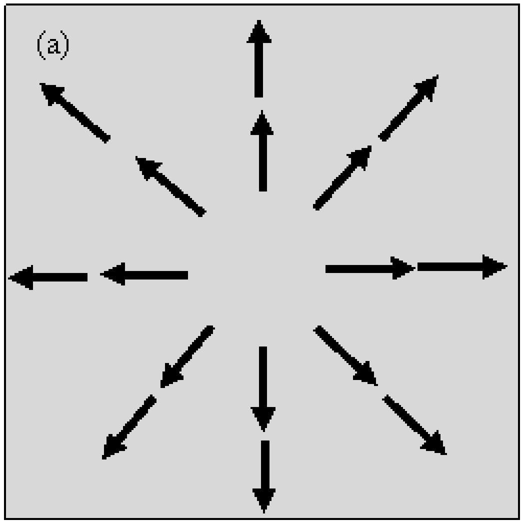
Wikipedia

Recap of differential operators (in 3D)

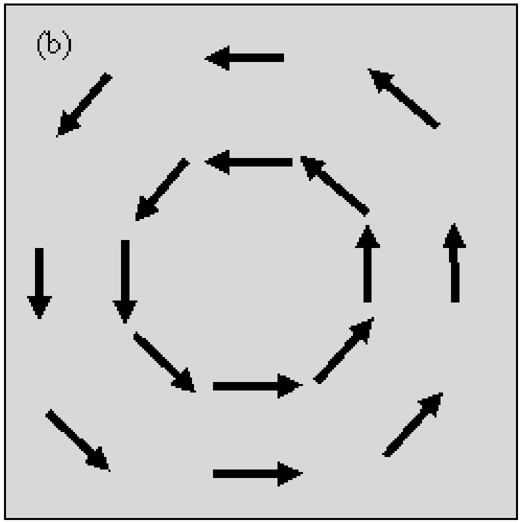
- **Divergence** (of vector-valued function):

$$\nabla \cdot V = \frac{\partial V_x}{\partial x} + \frac{\partial V_y}{\partial y} + \frac{\partial V_z}{\partial z}$$

  - Maps vector field to scalar field



Has divergence



Divergence-free

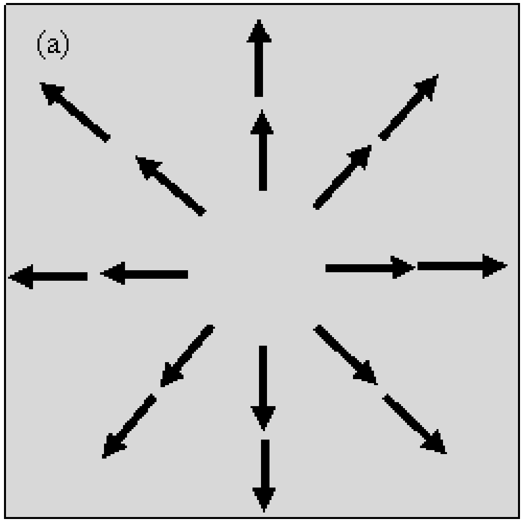
Gamasutra

Recap of differential operators (in 3D)

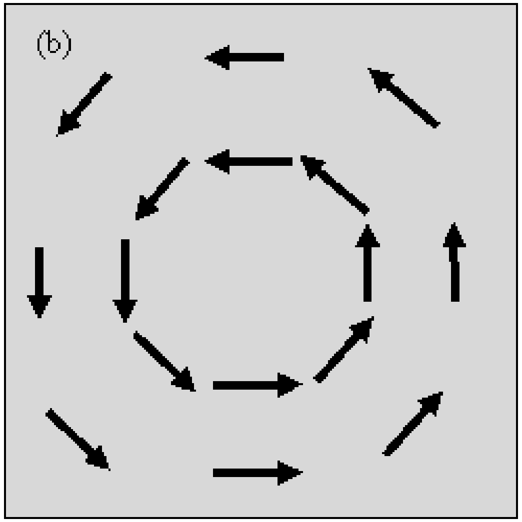
- **Curl** (of vector-valued function):

$$\nabla \times V = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z} \right) \times \left(V_x, V_y, V_z \right)$$

  - Maps vector field to vector field



Curl-free



Has curl

Recap of differential operators (in 3D)

- **Laplacian** (of scalar-valued function):
  $$\Delta f = \nabla \cdot \nabla f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial z^2}$$
  - In operator form:
    $$\Delta = \left(\frac{\partial^2}{\partial x^2}, \frac{\partial^2}{\partial y^2}, \frac{\partial^2}{\partial z^2} \right)$$
  - Maps scalar field to scalar field

Original function

After applying Laplacian

Recap

- The boundary of a shape is a level set of its indicator function χ
- The gradient $\nabla\chi$ of χ is the normal field V at the boundary (after some smoothing which we won't go into here)
- We can solve for χ by integrating the normal field
- ... but in general, we can't get an exact solution since an arbitrary vector field need not be the gradient of a function (field needs to be curl-free)
- So we find a least-squares fit, minimizing $||\nabla\chi - V||^2$

Recap

- So we find a least-squares fit, minimizing $\|\nabla\chi - V\|^2$
- This reduces to solving the Poisson Equation

$$\nabla\cdot(\nabla\chi) = \nabla\cdot V \quad \Leftrightarrow \quad \Delta\chi = \nabla\cdot V$$

- We can discretize the system by representing the functions as vectors of values at sample points
  - Gradient, divergence and Laplacian operators become matrices
- Solving the resulting linear system gives a least squares fit at the sample positions

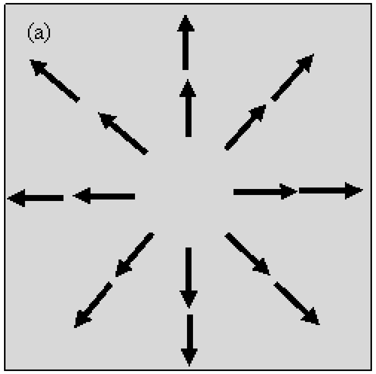
Why can't we solve it exactly?

- Over a non-loop 1D range (which we studied closely), this isn't very useful – the gradient $\frac{d}{dx}$ is invertible by integration and we can solve the system $\frac{d\chi}{dx}=V$ exactly
  - We can also do this in the discrete setting – the corresponding operator matrix is invertible
- But in 2 and higher dimensions, the gradient is not invertible, and neither is its operator matrix
  - Gradient maps scalar field to vector field: intuitively, “lower-dimensional” to “higher-dimensional”
  - In 1D, scalars and vectors are the same

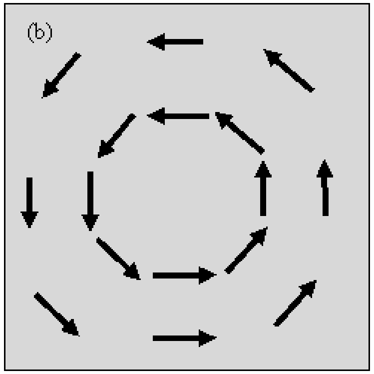
Non-invertibility of k-D continuous operators

- A vector field (over a simply-connected region) is the gradient of a scalar function if and only if it is curl-free (has no circulation about any point)
  - In other words, we can solve $\nabla\chi = V$ (over a simply-connected region) if and only if $\nabla \times V = 0$
  - If the region is not simply-connected, even this may not be enough

$$\nabla \times V = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}\right) \times (V_x, V_y, V_z)$$



Curl-free



Not curl-free

Non-invertibility of k-D discrete operators

n rows

$$\underbrace{\begin{pmatrix} \nabla \end{pmatrix}}_{n \text{ columns}} \begin{pmatrix} \chi \end{pmatrix} = \begin{pmatrix} V \end{pmatrix}$$

kn rows

(k-D discrete gradient)

kn rows

(1 row for each coordinate of each point)

Overdetermined

Non-invertibility of k-D discrete operators

n rows

$$\left[\nabla\cdot \right] \left(V \right) = \left(\mathbf{g} \right)$$

(k-D discrete divergence)

kn columns

kn rows

(1 row for each coordinate of each point)

n rows

Underdetermined

Thought for the Day #1

What about the Laplacian? Is it invertible?

$$\begin{array}{cccccc} & & & n \text{ rows} & & n \text{ rows} \\ n \text{ rows} & \left(\begin{array}{c} \nabla \cdot \nabla \\ \text{(}k\text{-D discrete} \\ \text{Laplacian)} \end{array} \right) & \left(\chi \right) & & = \left(\mathbf{g} \right) & \\ & n \text{ columns} & & & & \end{array}$$

Is this over- or under-determined?

What we have so far

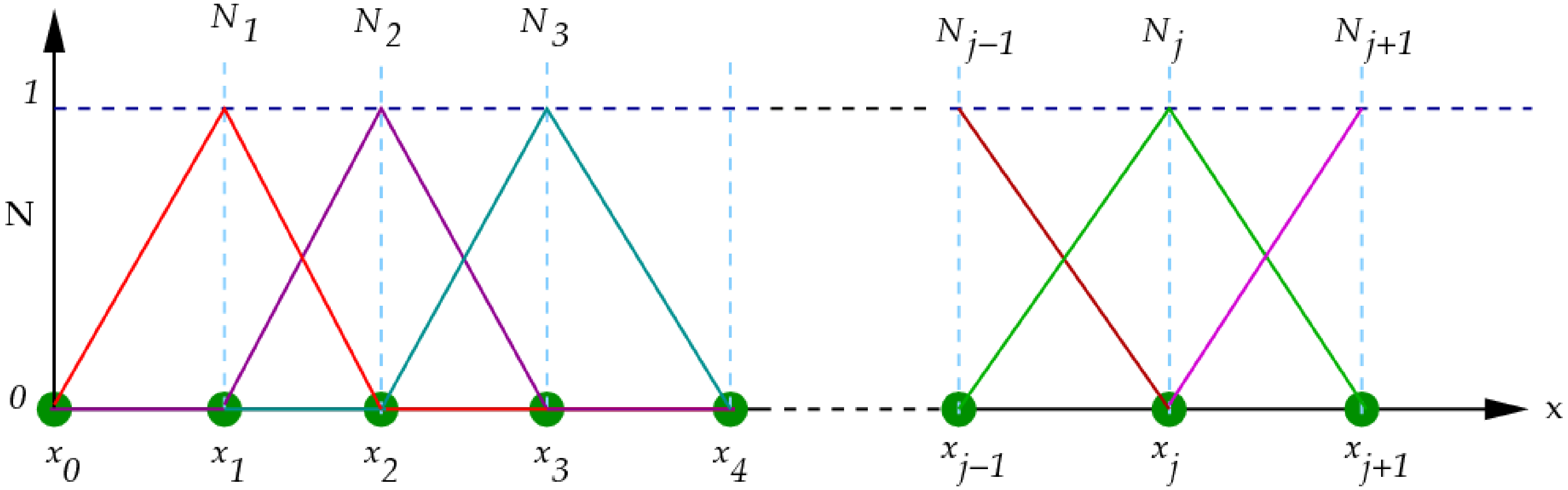
- Transform continuous variational problems to discrete linear algebra problems
- Solve in a least squares sense, since the problem is overdetermined in higher dimensions
- **BUT:** the results are also discrete: the values of the function f at the sampled points
  - **Solution:** A different type of discretization

Galerkin Approximation

- Restrict the solution space F to weighted sums of basis functions, i.e. $F = \{ \sum_i w_i B_i \}$, for some set of functions $B_1, B_2 \ldots B_m$
- Why? Allows us to discretize the problem in terms of the m-D vector of *weights*
- We will choose functions that are locally supported
  - ... i.e. each f_i is non-zero only around some local region of space
  - This keeps the resulting linear system sparse

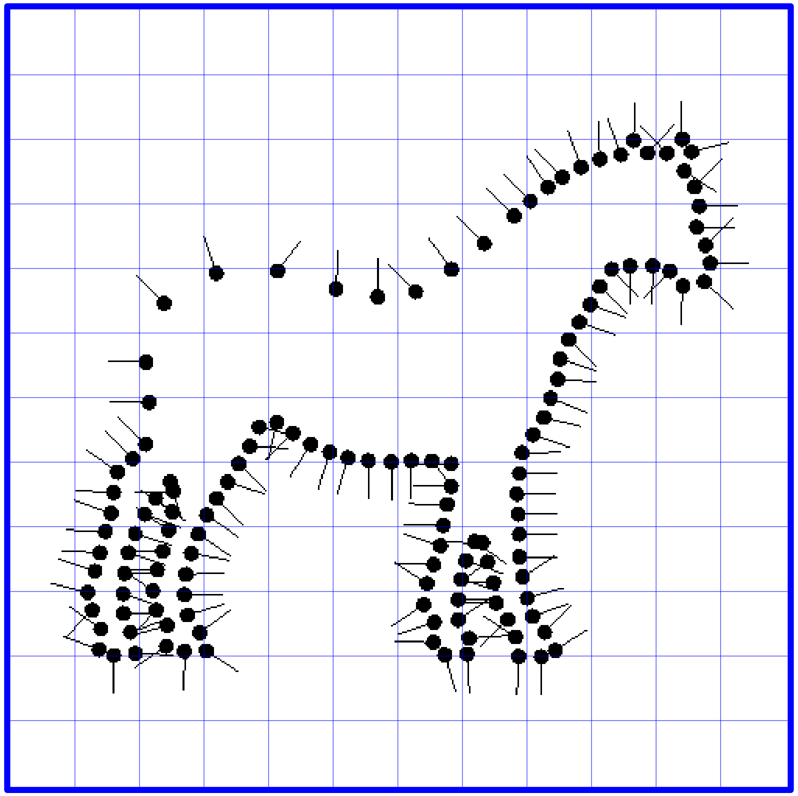
Basis Functions with Local Support

- A **finite element** model
- Discretize space into cells, then define a basis function centered around each cell



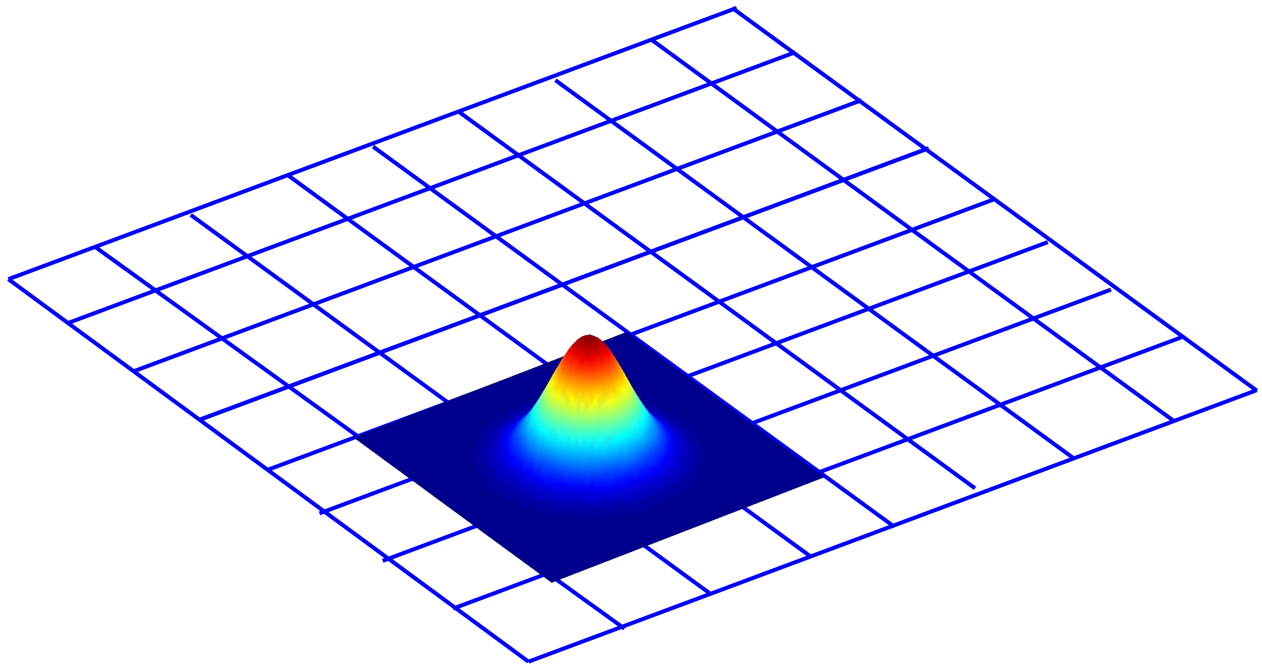
Instead of values *at* points, we now have values locally *around* points

Basis Functions with Local Support



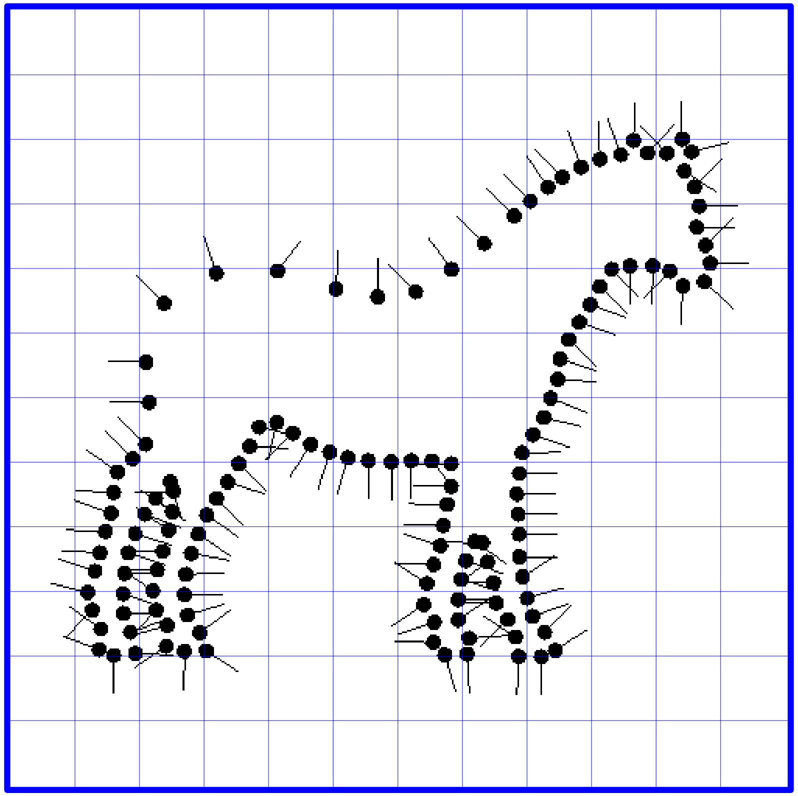
A potential grid of cells.

Basis Functions with Local Support



A single basis function, centered at a grid cell but overlapping adjacent cells

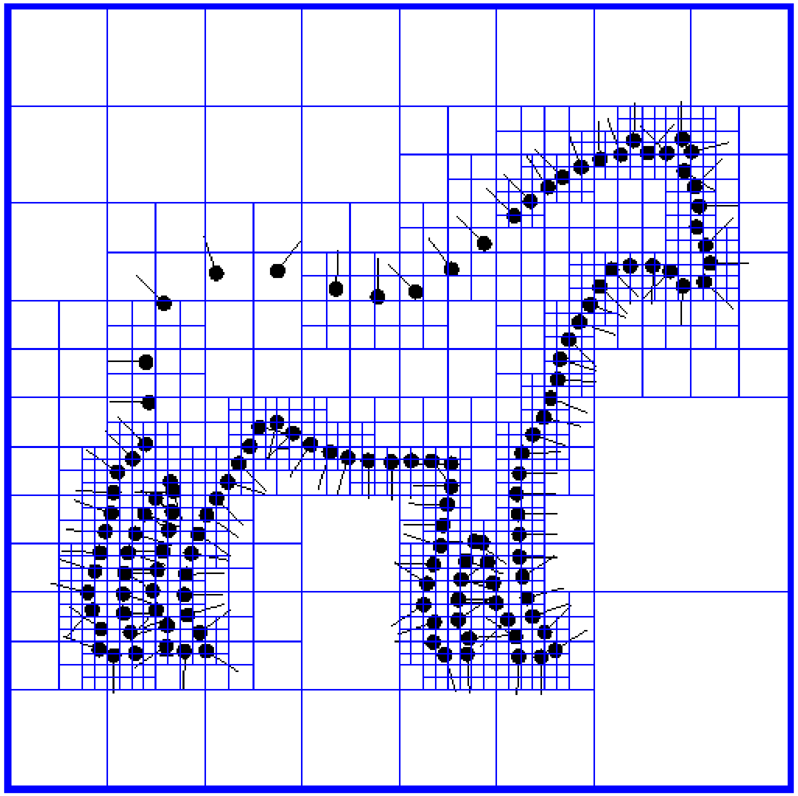
Basis Functions with Local Support



A potential grid of cells.

Problem: Not enough detail where it's needed (boundary), too much detail where it's not (empty space or interior)

Basis Functions with Local Support



A hierarchical, adaptive grid (octree).

Puts resolution where it matters. One basis function per octree cell.

Projecting to the Finite Basis

- Assume we want to reconstruct the function over range Ω (e.g. $[0, 1]$ in 1D, or $[0, 1]^3$ in 3D)
- The original Poisson problem is $\Delta\chi = \nabla \cdot V$
- **BUT:** since we've now restricted our solutions to the space spanned by $\{B_i\}$, this equation may not have an exact solution!
  - **Solution:** Least squares to the rescue again!

Projecting to the Finite Basis

- **Solve:** $\Delta\chi = \nabla \cdot V$ for $\chi \in F$
- To find the best solution within the space spanned by the basis, we minimize the sum of squared projections onto the basis functions

$$\sum_{i=1}^{m} \left\langle \Delta \chi - \nabla \cdot V , B_i \right\rangle_{\Omega}^{2}$$

where $\langle f , B_i \rangle = \int_{\Omega} f(x) B_i(x) \, \mathrm{d}\sigma$ measures the projection of function f onto basis function B_i

Projecting to the Finite Basis

- **Minimize:** $\sum_{i=1}^{m} \langle \Delta\chi - \nabla\cdot V, B_i \rangle_\Omega^2$

$$= \sum_{i=1}^{m} \left| \langle \Delta\chi, B_i \rangle - \langle \nabla\cdot V, B_i \rangle_\Omega \right|^2$$

- (skipping some algebra) This amounts to minimizing $||L\mathbf{w} - \mathbf{v}||^2$, where

$$L_{ij} = \underbrace{\left\langle \frac{\partial^2 B_i}{\partial x^2}, B_j \right\rangle + \left\langle \frac{\partial^2 B_i}{\partial y^2}, B_j \right\rangle + \left\langle \frac{\partial^2 B_i}{\partial z^2}, B_j \right\rangle}_{\text{Mostly zero, since most pairs of basis functions don't overlap}} \qquad \mathbf{w} = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_m \end{pmatrix}$$

$$v_i = \langle \nabla\cdot V, B_i \rangle$$

Assignment 1: Point Clouds

- **Given:**
  - Point cloud class + display functions
  - Utility toolkit with lots of useful code
- **Todo:**
  - Estimate the normal at each point
    - Construct a kd-tree for range queries
    - Apply regression or any other suitable method over local neighborhoods
    - Extra credit: Ensure they are consistently oriented
    - Extra credit: Handle sharp edges correctly
    - Extra credit: Adaptively downsample the point cloud: reduce #points in flat regions with similar normals