Multiscale biological modeling: Using every area of mathematics that you can imagine

Andrew Gillette

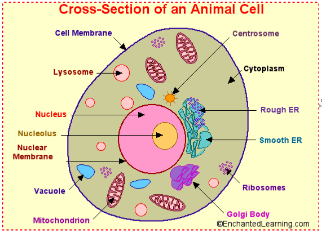
Department of Mathematics
University of Arizona

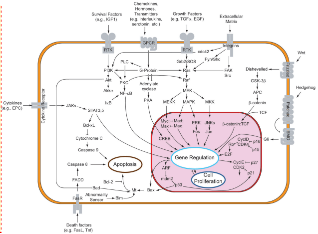
Recruitment Workshop Presentation

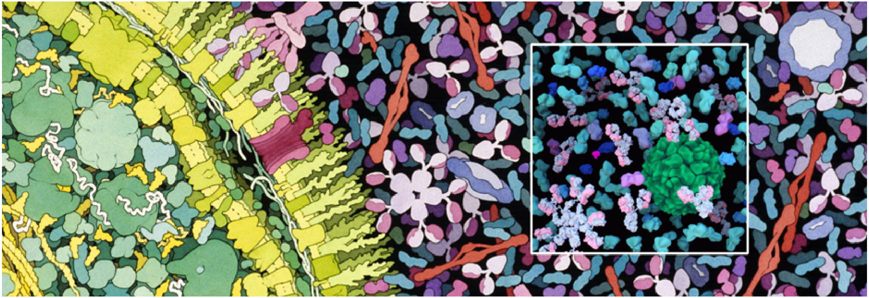
Slides and more info at:

http://math.arizona.edu/∼agillette/

What’s relevant in molecular modeling?

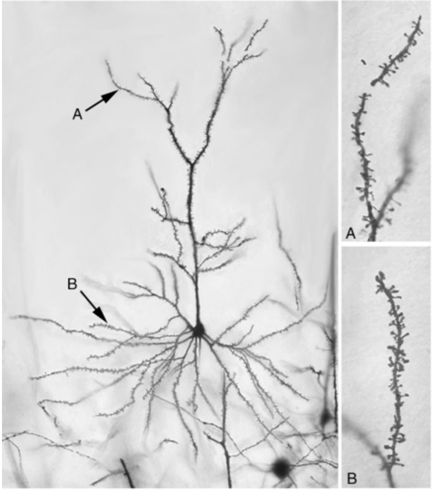


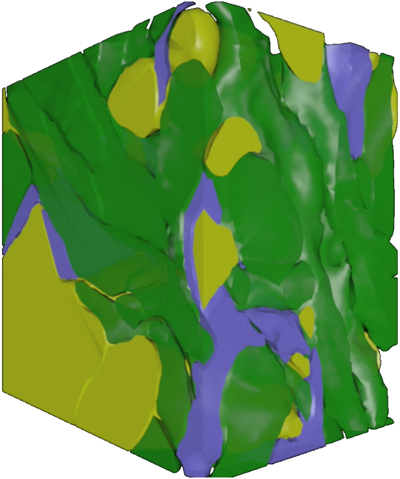




(bottom image: David Goodsell)

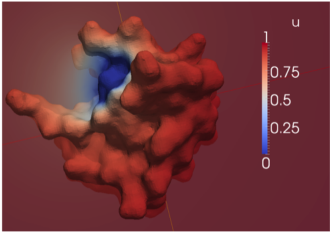
What's relevant in neuronal modeling?

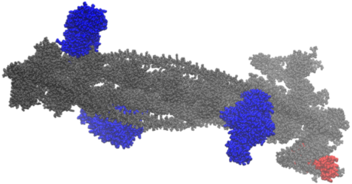


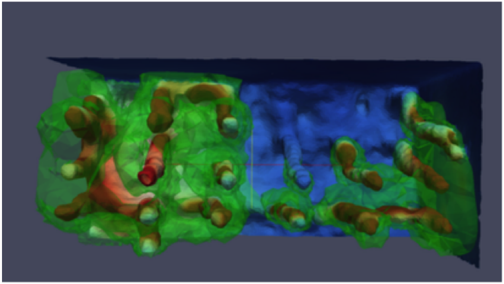


(right image: Chandrajit Bajaj)

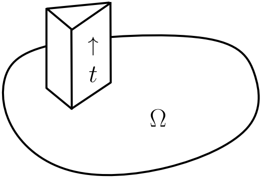
What's relevant in diffusion modeling?

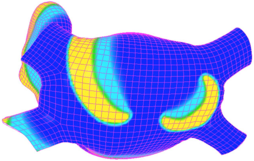






Mathematics used in multiscale biological models





$$\begin{bmatrix} & & \\ & \mathbb{A} & \\ & & \end{bmatrix} \begin{bmatrix} \\ \mathbf{x} \\ \\ \end{bmatrix} = \begin{bmatrix} \\ \mathbf{b} \\ \\ \end{bmatrix}$$

Real analysis

PDEs

Geometry

Topology

Combinatorics

Linear algebra

Numerical analysis

Mathematics helps answer distinguish relevant and irrelevant features of a model:

- Does the PDE have a unique solution, bounded in some norm?
- Does the domain discretization affect the quality of the approximate solution?
- Is the solution method optimally efficient? (e.g. Why isn't my code working?)

Focus of my research in these areas: the **Finite Element Method**

The rest of this talk

1. A very brief introduction to the finite element method

2. A mathematical challenge for efficient computation

   Employing ideas from analysis, linear algebra, and combinatorics

3. A quick historical quiz. . .

The Finite Element Method: 1D

The **finite element method** is a way to numerically approximate the solution to PDEs.

Ex: The 1D Laplace equation: find $u(x) \in U$ s.t.

$$\begin{cases} -u''(x) = f(x) & \text{on } [a, b] \\ u(a) = 0, \\ u(b) = 0 \end{cases}$$

Make the problem easier by making it (seemingly) harder ...

Weak form: find $u(x) \in U$ ($\dim U = \infty$) s.t.

$$\int_a^b u'(x)v'(x)\,dx = \int_a^b f(x)v(x)\,dx, \quad \forall v \in V \quad (\dim V = \infty)$$

... but we can now search a finite-dimensional space...

Discrete form: find $u_h(x) \in U_h$ ($\dim U_h < \infty$) s.t.

$$\int_a^b u_h'(x)v_h'(x)\,dx = \int_a^b f(x)v_h(x)\,dx, \quad \forall v_h \in V_h \quad (\dim V_h < \infty)$$

The Finite Element Method: 1D

Suppose $u_h(x)$ can be written as linear combination of V_h elements:

$$u_h(x) = \sum_{v_i \in V_h} u_i v_i(x)$$

The discrete form becomes: find coefficients $u_i \in \mathbb{R}$ such that

$$\sum_i \int_a^b u_i v_i'(x) v_j'(x)\, dx = \int_a^b f(x) v_j(x)\; dx, \quad \forall v_h \in V_h \quad (\dim V_h < \infty)$$

Written as a linear system:

$$[\, \mathbb{A} \,]_{ji}\ [\, u \,]_i = [\, f \,]_j\,, \quad \forall v_j \in V_h$$

With some functional analysis we can prove: $\exists\ C > 0$, independent of h, s.t.

$$\underbrace{||u - u_h||_{H^1(\Omega)}}_{\text{error between cnts and discrete solution}} \leq \underbrace{C\, h\, |u|_{H^2(\Omega)}}_{\text{bound in terms of 2nd order osc. of } u}, \quad \underbrace{\forall u \in H^2(\Omega)}_{\text{holds for any } u \text{ with bounded 2nd derivs.}}.$$

where h = maximum width of elements use in discretization.

Tensor product finite element methods

Generalizing the 1st order, 1D method

Goal: Efficient, accurate approximation of the solution to a PDE over $\Omega \subset \mathbb{R}^n$ for arbitrary dimension n and arbitrary rate of convergence r.

Standard $O(h^r)$ **tensor product** finite element method in $\mathbb{R}^n$:

$\rightarrow$ Mesh Ω by n-dimensional cubes of side length h.

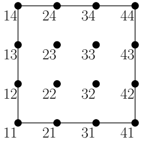
$\rightarrow$ Set up a linear system involving $(r+1)^n$ degrees of freedom (DoFs) per cube.

$\rightarrow$ For unknown continuous solution u and computed discrete approximation u_h:

$$\underbrace{||u - u_h||_{H^1(\Omega)}}_{\text{approximation error}} \leq \underbrace{C\, h^r\, |u|_{H^{r+1}(\Omega)}}_{\text{optimal error bound}}, \qquad \forall u \in H^{r+1}(\Omega).$$

Implementation requires a clear characterization of the isomorphisms:

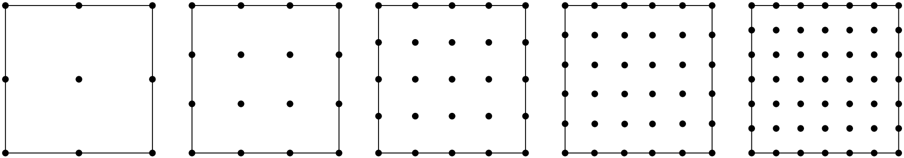
$$\left\{ \begin{array}{c} x^r y^s \\ 0 \leq r, s \leq 3 \end{array} \right\} \longleftrightarrow \left\{ \begin{array}{c} \psi_i(x)\psi_j(y) \\ 1 \leq i, j \leq 4 \end{array} \right\} \longleftrightarrow$$



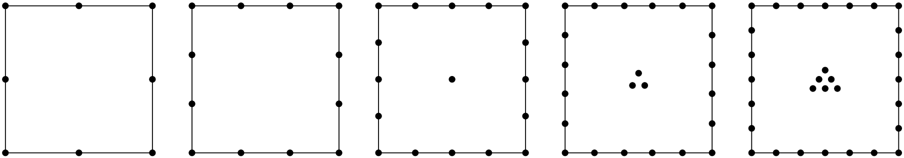
monomials $\longleftrightarrow$ **basis functions** $\longleftrightarrow$ **domain points**

Serendipity Elements

9 | 16 | 25 | 36 | 49



$O(h^2)$ | $O(h^3)$ | $O(h^4)$ | $O(h^5)$ | $O(h^6)$



8 | 12 | 17 | 23 | 30

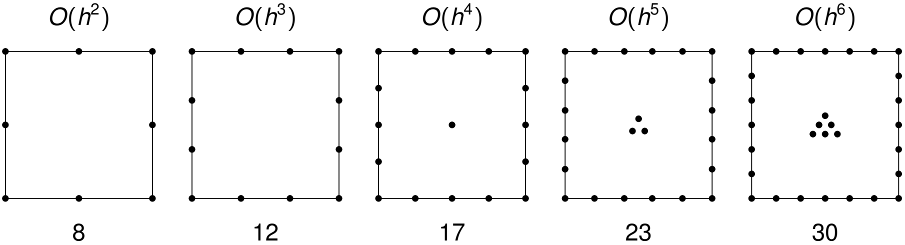
For $r \geq 4$ on squares:

$O(h^r)$ tensor product method : $r^2 + 2r + 1$ dots

$O(h^r)$ serendipity method: $\frac{1}{2}(r^2 + 3r + 6)$ dots

$$\underbrace{||u - u_h||_{H^1(\Omega)}}_{\text{approximation error}} \leq \underbrace{C\, h^r\, |u|_{H^{r+1}(\Omega)}}_{\text{optimal error bound}}, \quad \forall u \in H^{r+1}(\Omega).$$

Serendipity Elements

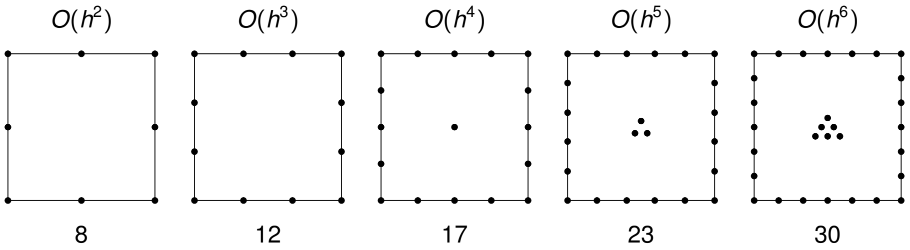


$\rightarrow$ Why $r+1$ dots per edge?
Ensures continuity between adjacent elements.

$\rightarrow$ Why interior dots only for $r \geq 4$?
Consider, e.g. $p(x, y) := (1+x)(1-x)(1-y)(1+y)$
Observe p is a degree 4 polynomial but $p \equiv 0$ on $\partial([-1, 1]^2)$.

$\rightarrow$ How can we recover tensor product-like structure...
...without a tensor product structure?

Mathematical Challenges More Precisely

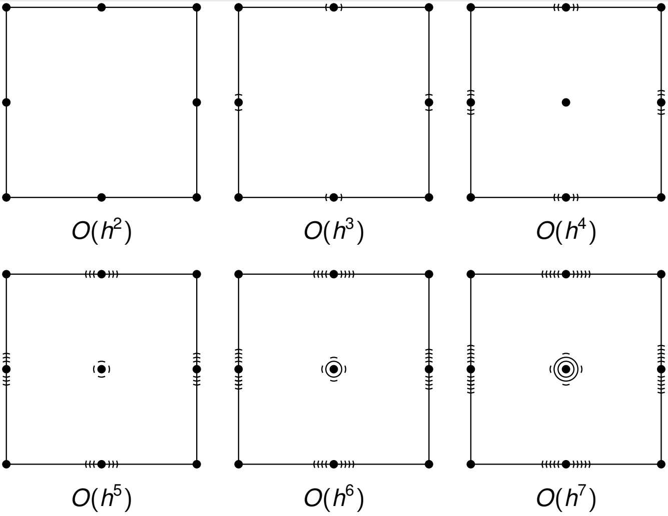


Goal: Construct basis functions for serendipity elements satisfying the following:

- **Symmetry:** Accommodate interior degrees of freedom that grow according to triangular numbers on square-shaped elements.
- **Hierarchical:** Generalize to methods on n-cubes for any $n \geq 2$, allowing restrictions to lower-dimensional faces.
- **Tensor product structure:** Write as linear combinations of standard tensor product functions.

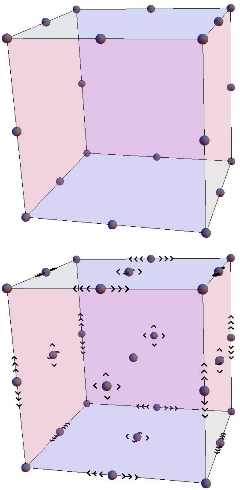
2D symmetric serendipity elements

Symmetry: Accommodate interior degrees of freedom that grow according to triangular numbers on square-shaped elements.



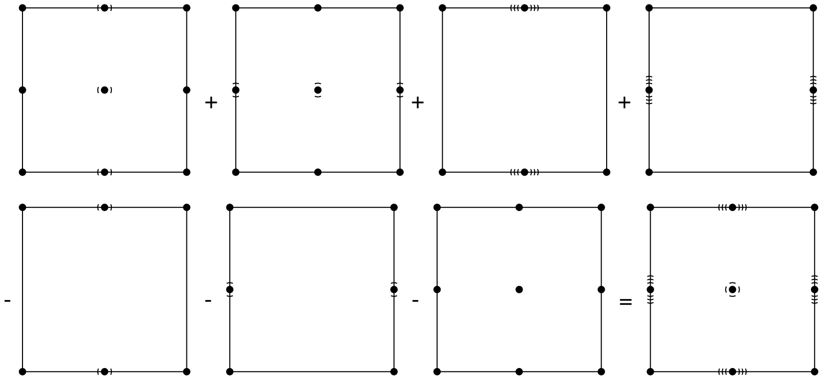
3D elements

Hierarchical: Generalize to methods on *n*-cubes for any $n \geq 2$, allowing restrictions to lower-dimensional faces.



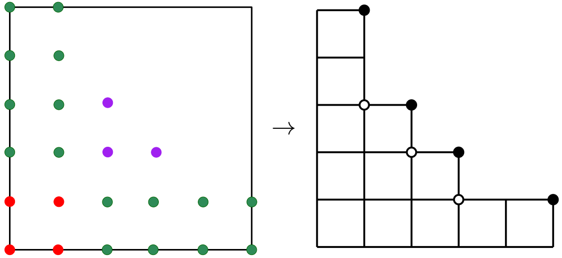
Linear combination of tensor products

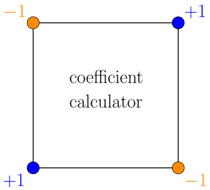
Tensor product structure: Write basis functions as linear combinations of standard tensor product functions.



Tensor product decomposition

The coefficients in the tensor product decomposition in 2D can be determined from a ‘staircase’ called a **lower set**.



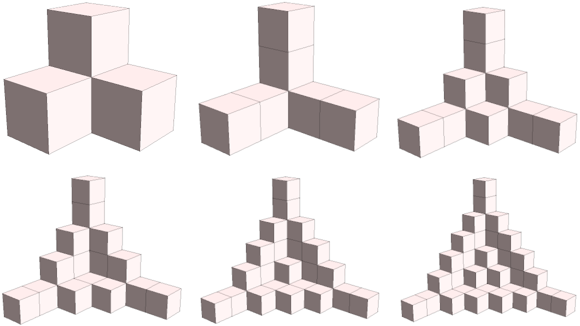


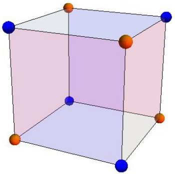
→ Place the coefficient calculator at the corners of each step.
→ Add up all values appearing in the staircase.
→ The coefficient for the block is the value of the sum.

Hence: black dots → +1; white dots → -1; others → 0.

3d coefficient computation

Lower sets for superlinear polynomials in 3 variables:





Decomposition into a linear combination of tensor product interpolants works the same as in 2D, using the 3D coefficient calculator at left. (Blue $\rightarrow$ +1; Orange $\rightarrow$ -1).

Historical Quiz

What video game is shown on the right?