High-Order Serendipity Finite Elements for Gkeyll

Eric Shi, Ammar Hakim, Greg Hammett

Graduate Seminar Talk, 7 Jan 2013

Outline

Major Motivations

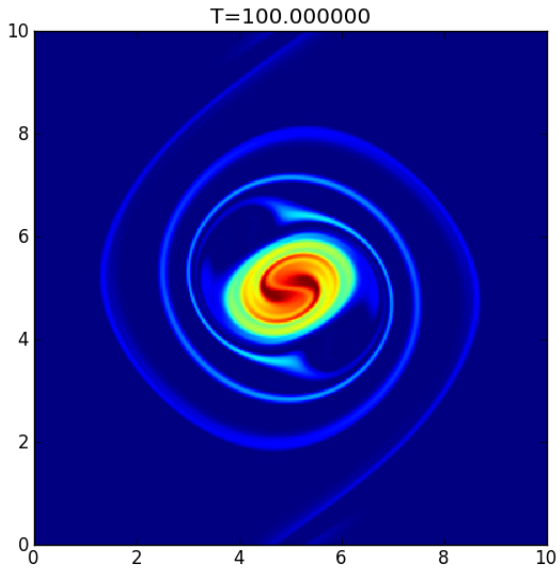
- Edge region is important, but complicated
- Tokamak edge physics relatively unexplored: no complete model of self-consistent cross-field transport in open-field line region, very little study of neutral transport, wall effects, etc.
- Large density/amplitude variations, large relative banana width, wide range of collisionalities

Major Motivations

- Need comprehensive simulations of edge turbulence because predicted fusion performance strongly dependent on edge temperature
  - ELM suppression/mitigation, spontaneous flow, Li walls
- Need new code or major extensions to existing codes to handle edge region
- Advanced algorithms can help with these additional challenges

Gkeyll Overview

- Prototype code to explore advanced algorithms for continuum edge gyrokinetic simulation (e.g. edge plasma turbulence)
  - Hybrid discontinuous/continuous Galerkin methods augmented with reconstruction techniques from finite volume schemes
- Main code is written in C++
- Lua scripts for simulations



Goal

A robust code capable of running very quickly at coarse velocity space resolution while preserving all conservation laws of gyro-fluid/fluid equations and giving fairly good results.

Objective

Explore basis functions that reduce computational effort, yet retain the formal high-order accuracy for 4-D/5-D gyrokinetic simulations

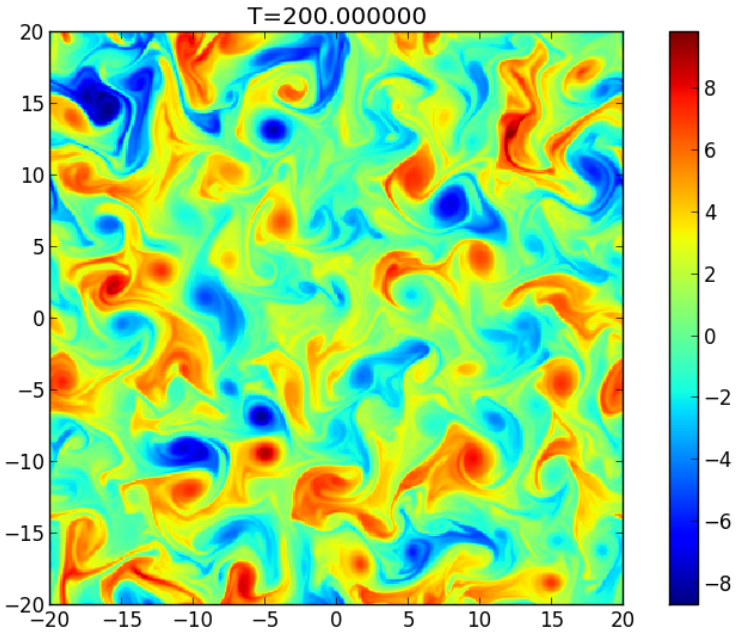
- 1024 unknowns per element using standard 3rd order element in 5-D
- Investigate serendipity basis functions

Hyperbolic PDEs

- Conservation laws

$$\frac{\partial Q}{\partial t} + \nabla \cdot \boldsymbol{F}(Q) = \psi(Q)$$

- Wave equations
- Euler equations
- Navier-Stokes equations
- Two-Fluid MHD
- Vlasov Equation
- Hasegawa-Wakatani equations
- Gyrokinetic equations



Finite Element Methods

- Technique for solving systems of PDEs
- Example: Consider a differential equation with the exact solution in blue
- Seek approximate solution (red) as a sum of piecewise linear functions

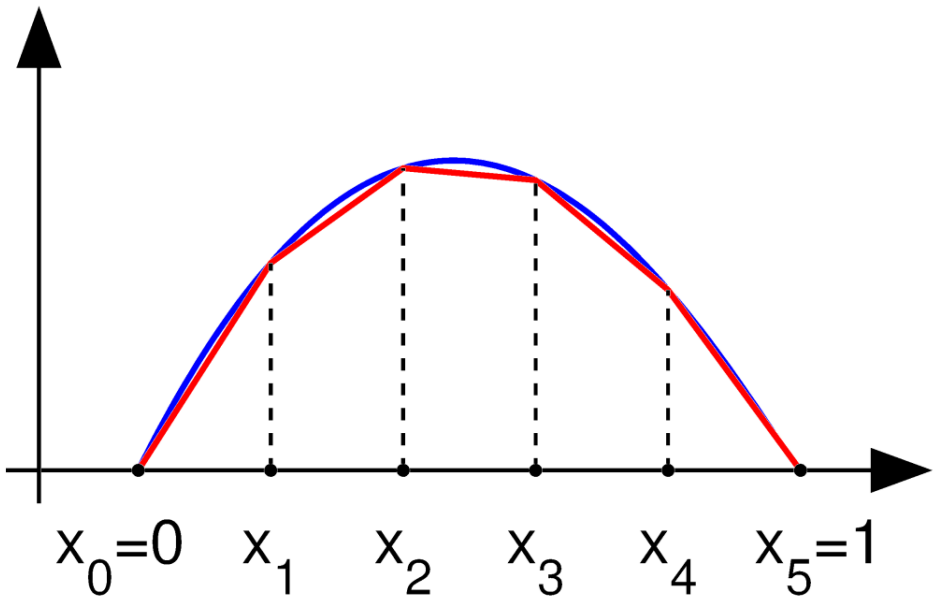


Image from Wikipedia

Finite Element Methods

- Partition solution domain into elements of simple geometrical shape (mesh)
  - Each element contains a number of nodes
- Use finite element analysis to find the 'optimal' linear combination of *basis functions* for approximate solution

$$u(x) \approx \tilde{u}(x) = \sum_{k=1}^{K} w_k N_k(x), \; N_k(x_j) = \delta_{kj}$$

- The N_k(x) (blue) are basis functions, x_k are nodes

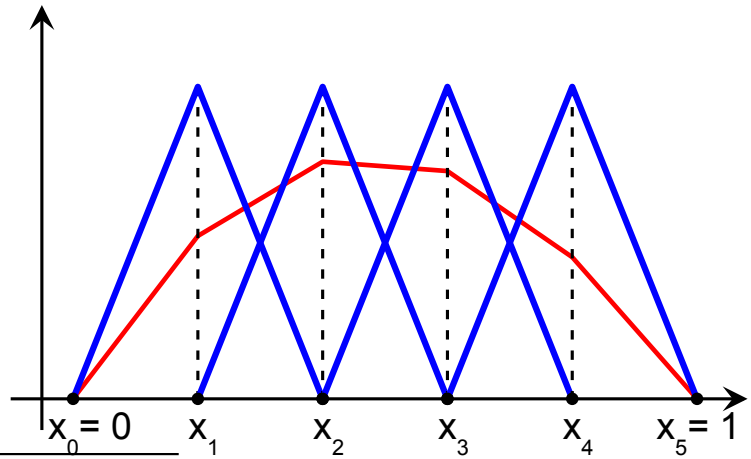


Image from Wikipedia

Finite Element Methods

- Due to basis function properties, approximate solution can be expressed in terms of the *function values* at the nodes

$$u(x) \approx \tilde{u}(x) = \sum_{k=1}^{K} \tilde{u}(x_k) N_k(x), \; N_k(x_j) = \delta_{kj}$$

- Solve K algebraic equations to find unknown weights $w_k = \tilde{u}(x_k, t)$

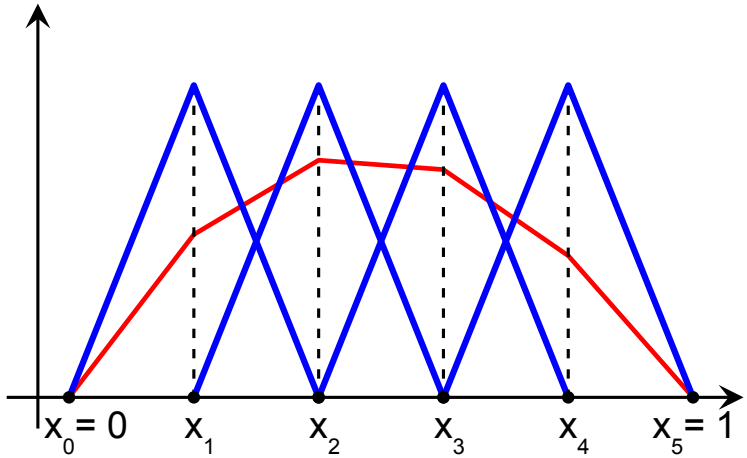


Image from Wikipedia

Finite Element Methods

- *Nodal* basis functions $N_k(x)$ evaluate to 1 at node x_k and 0 at every $x_{j \neq k}$
- There is flexibility in choosing basis functions–don't need to use tent functions
- Two strategies for increasing accuracy:
  - Use smaller elements (more elements to discretize domain)
  - Use basis functions composed of higher degree polynomials
- Cost of using higher-degree basis functions is more degrees of freedom (unknowns) per element
  - Use higher-degree basis functions with larger elements?

Polynomial Completeness

- If basis functions span a complete Nth degree polynomial and finite elements have length h, error behaves as

$$\|u - \tilde{u}\| \leq Ch^{N+1}$$

- Useful to recall Pascal triangle:

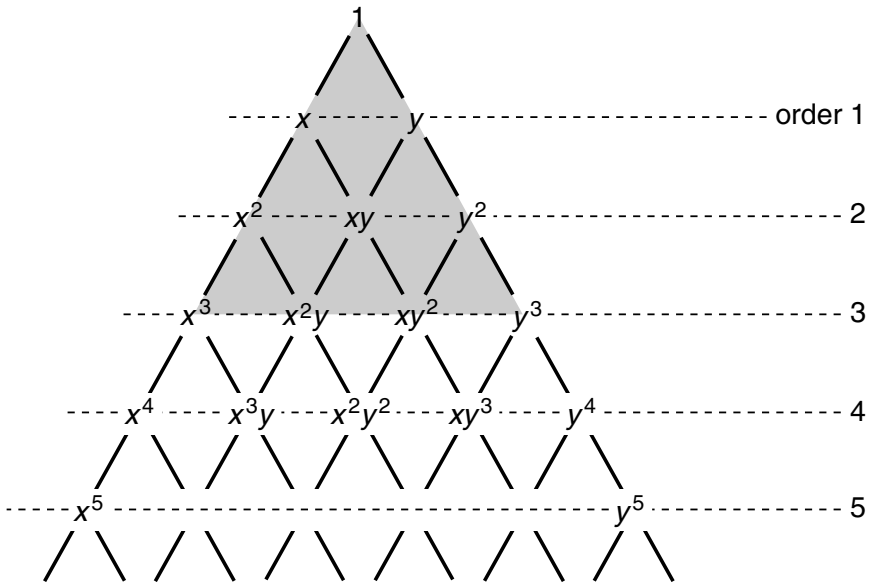


Fig. 4.5 The Pascal triangle. (Cubic expansion shaded – 10 terms.)

Image from *The Finite Element Method: Its Basis and Fundamentals*

Degrees of Freedom (DOF)

- In the FEM, solution is expressed in terms of a finite number of DOF + basis functions
- DOF are the unknown basis function weights, e.g.

$$u(x,t) \approx \tilde{u}(x,t) = \sum_{k=1}^{K} \tilde{u}(x_k,t) N_k(x)$$

- When using *nodal* elements, the DOF are characterized by the value(s) of a function at the nodes of each element
  - # unknowns = # DOF
  - Function can be $\tilde{u}$ and/or $\tilde{u}'$

Finite Elements in Higher Dimensions

- 1-D: elements of finite interval
- 2-D: elements of finite area (e.g. rectangles, triangles, curvilinear polygons)
- 3-D: elements of finite volumes (e.g. cubes, tetrahedra)
- All finite elements are transformations of reference elements (e.g. square, triangle, cube)

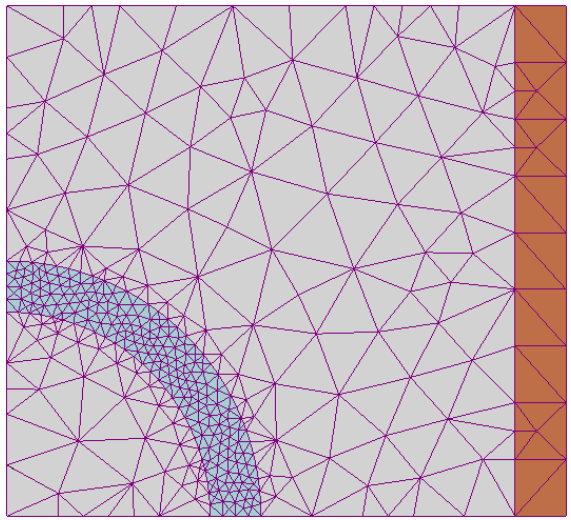


Figure: Example 2-D FEM mesh using triangles

Discontinuous Galerkin Background

- State-of-art for solution of hyperbolic partial differential equations
- First introduced by Reed and Hill for steady-state 2-D neutron transport in 1973
- Runge-Kutta DG for time-dependent problems by Cockburn and Shu (1998)
  - Finite element space discretization, Runge-Kutta time discretization
- Widely used in computational fluid dynamics, finding use in more applications (e.g. atmospheric modeling, MHD)

Discontinuous Galerkin Solutions

Discontinuous Galerkin schemes use function spaces that allow *discontinuities* across cell boundaries.

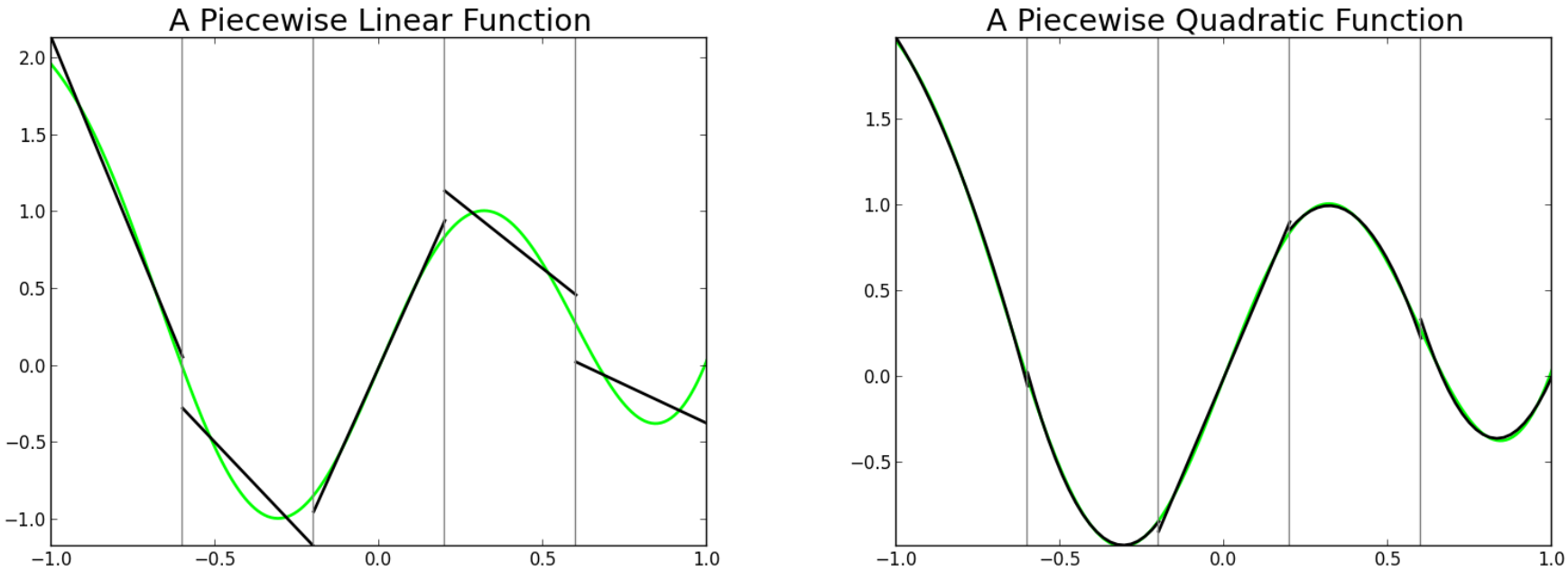


Figure: The best L^2 fit of $x^4 + \sin(5x)$ (green) with piecewise linear (left) and quadratic (right) basis functions.

Discontinuous Galerkin Features

- Numerical solution is discontinuous between elements
- Hybrid approach: exploit merits of classical finite element and finite volume methods
  - FV: slope limiters to control spurious oscillations, locality
  - FE: high-order accuracy, complex geometries
- Information only needs to be shared between neighboring elements–adapts well to massively parallel architectures

DG combined with FV schemes can lead to best-in-class explicit algorithms for *hyperbolic PDEs*.

Finite Elements in DG

- Finite elements are non-overlapping
- Basis functions are zero outside the element
- Solution within an element defined only in terms of that element's basis functions and DOF

$$u(x,t) \approx \tilde{u}(x,t) = \sum_{k=1}^{K} \tilde{u}(x_k,t) N_k(x)$$

Finite Element Spaces

- Set of all functions that can be written as linear combinations of the basis functions
- Specified for each degree and dimension by:
  1. Element shape (e.g. triangle, quadrilateral, hexahedron, etc.)
  2. Set of basis functions that span the function space
     - # basis functions per element = # DOF per element
     - (Nodal only): DOF on each face of dimension d (vertex, edge, interior)
- Example finite element spaces:
  - Lagrange (C^0)
  - Serendipity (C^0)
  - Hermite Cubic (C^1)

Lagrange Family

- Basis functions are tensor products of Lagrange polynomials

$$\ell_k^n(x) = \prod_{\substack{i=0 \\ i \neq k}}^{n} \frac{x - x_i}{x_k - x_i}, \; N_a(x, y) \equiv l_a^n(x) l_a^m(y)$$

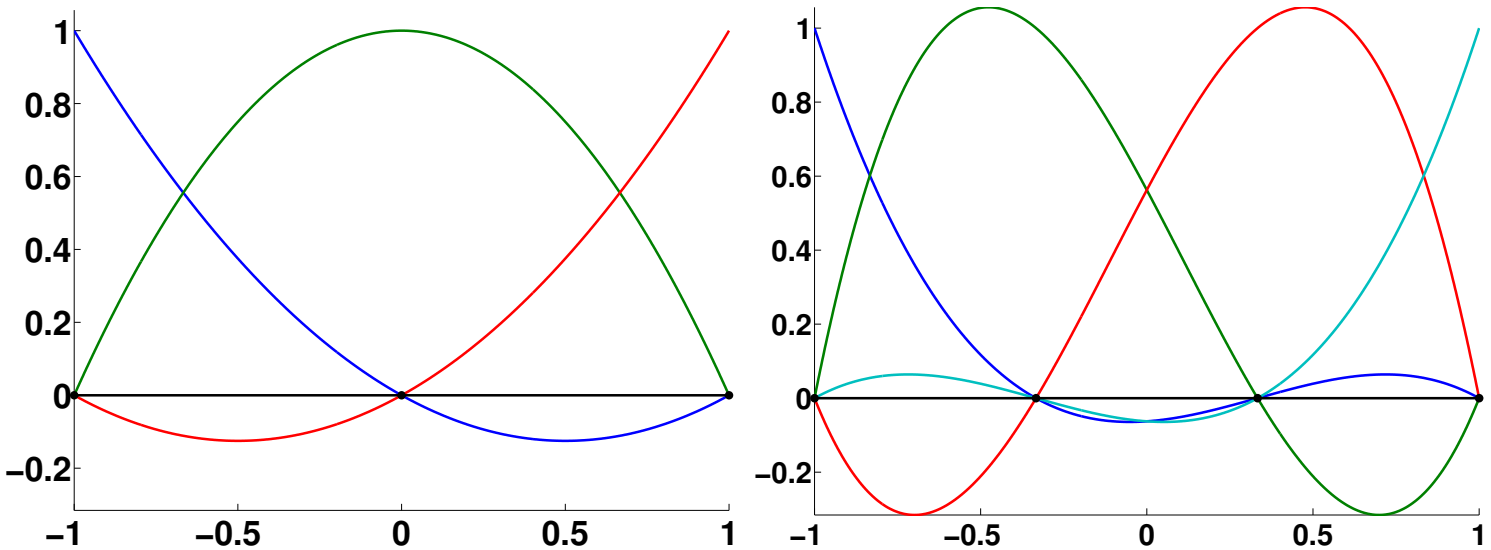


Figure: Basis functions for 2nd (right) and 3rd (left) order elements.

Lagrange Family

- Pros: Easy to implement, generalizes to arbitrary dimension and order
- Cons: Large number of DOF in higher dimension and order
  - In 5-D, need $(3 + 1)^5 = \mathbf{1024}$ DOF per element for 3rd order
- Large number of terms above those needed for complete expansion present!

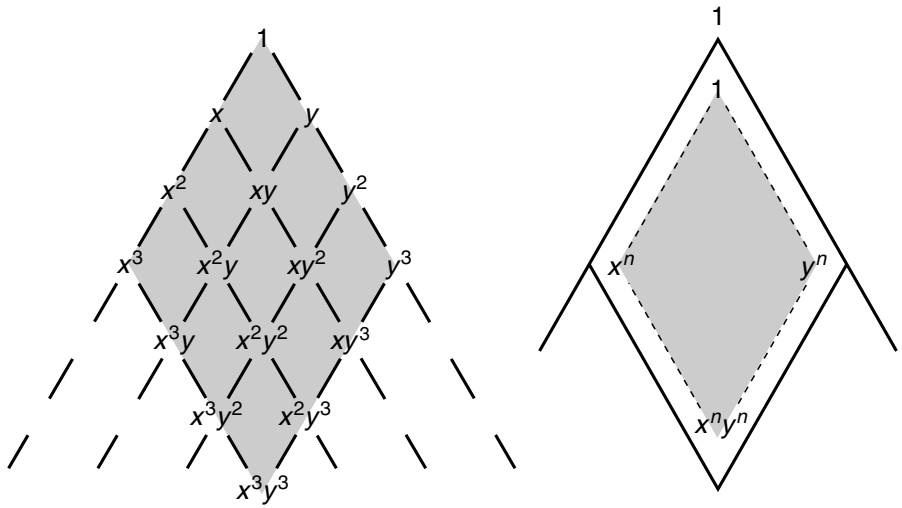


Fig. 4.8 Terms generated by a lagrangian expansion of order 3 × 3 (or $m \times n$). Complete polynomials of order 3 (or n).

Image from *The Finite Element Method: Its Basis and Fundamentals*

Serendipity Family

- Basis functions originally derived by inspection ('serendipity')
- Fewer interior nodes compared to Lagrange element of same order
  - Fewer DOF per element
  - Smaller dimension function space
- Expect lower accuracy, but faster than Lagrange elements of the same order

Key Question

Can serendipity elements deliver the same error at a lower computational cost (by choice of element size and order)?

Lagrange and Serendipity Compared in 2-D

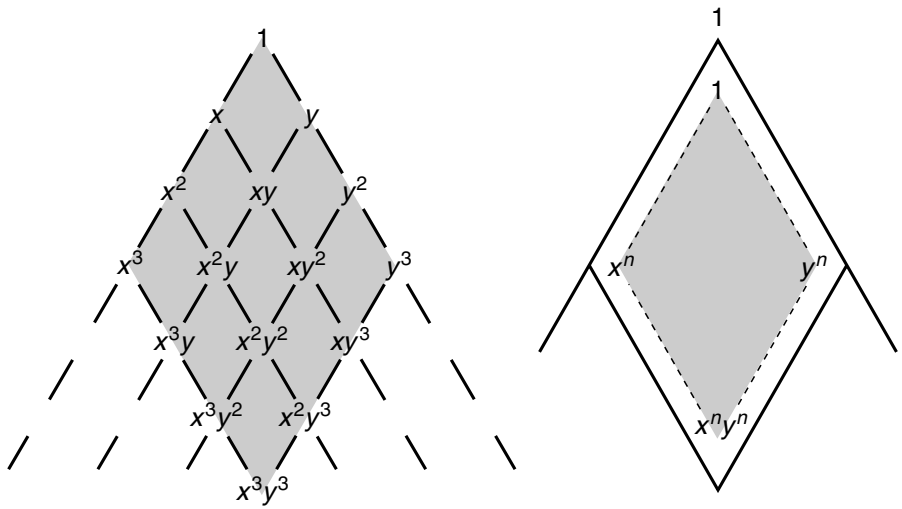


Fig. 4.8 Terms generated by a lagrangian expansion of order 3 × 3 (or $m \times n$). Complete polynomials of order 3 (or n).

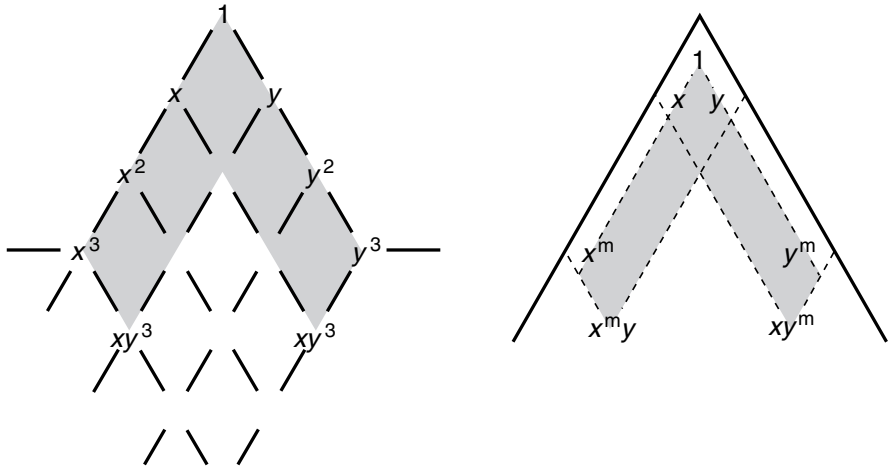


Fig. 4.12 Terms generated by edge shape functions in serendipity-type elements (3 × 3 and $m \times m$).

Image from *The Finite Element Method: Its Basis and Fundamentals*

Lagrange and Serendipity Compared in 2-D

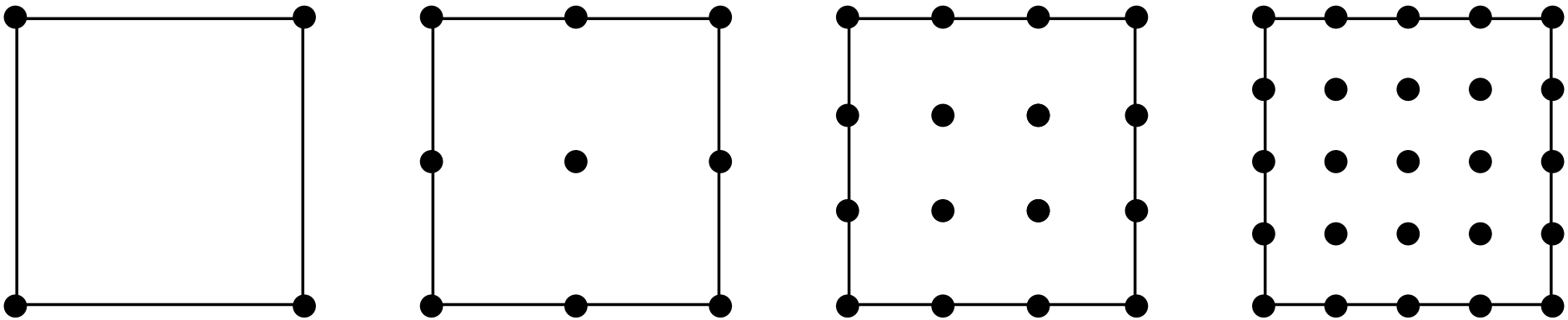


Figure: Node placement of 1st to 4th order Lagrange elements.

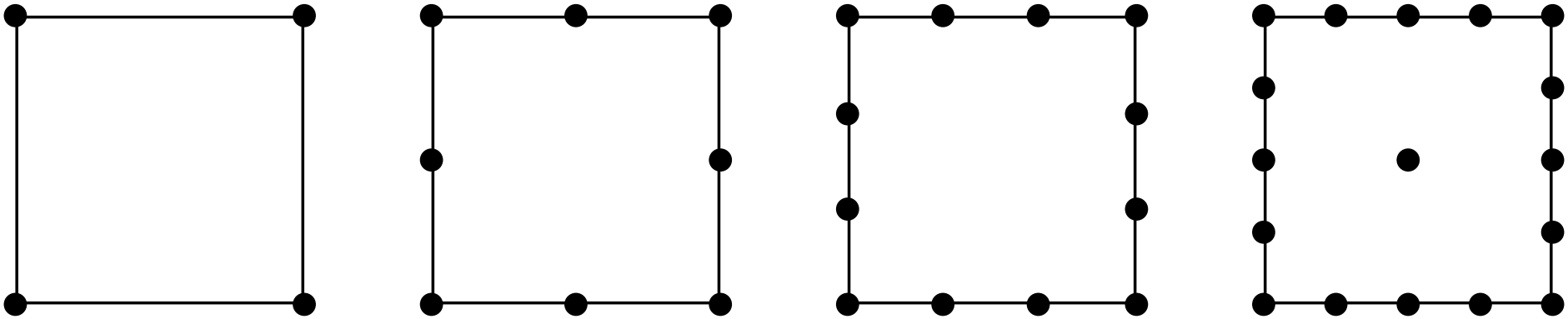


Figure: Node placement of 1st to 4th order Serendipity elements.

Larger savings in higher dimensions and order

Three Dimensions

Order	Lagrange	Serendipity
1	8	8
2	27	20
3	64	32
4	125	50

Five Dimensions

Order	Lagrange	Serendipity
1	32	32
2	243	112
3	1024	192
4	3125	352

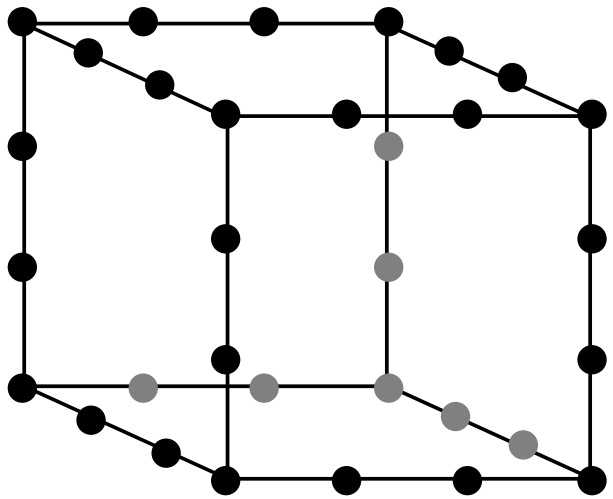


Figure: 3rd order Serendipity element in 3-D

Serendipity Family Definition

- Typically used with low order (r<3) in 2-D and to a lesser extent in 3-D
- Pattern for progression to higher orders, higher dimensions not evident ('serendipity')
- Simple and new dimension-independent definition given by Arnold[1] for serendipity elements

Definition

The serendipity space $\mathcal{S}_r(I^n)$ is the space of all polynomials in n variables with superlinear degree (total degree with respect to variables entering at least quadratically) at most r.

[1] **serendipity**.

Example: Degree 3 Polynomial Spaces (2-D)

$\mathcal{P}_r(I^n)$:=span{monomials in n variables with degree $\leq r$}
$\mathcal{S}_r(I^n)$:=span{monomials in n variables with superlinear degree $\leq r$}
$\mathcal{Q}_r(I^n)$:=span{monomials in n variables with each variable degree $\leq r$}

$$\mathcal{P}_3(I^2) = \text{span}\{1, x, y, x^2, y^2, xy, x^3, y^3, x^2y, xy^2\}$$
$$\mathcal{S}_3(I^2) = \mathcal{P}_3(I^2) \cup \text{span}\{x^3y, xy^3\}$$
$$\mathcal{Q}_3(I^2) = \mathcal{S}_3(I^2) \cup \text{span}\{x^2y^2, x^3y^2, x^2y^3, x^3y^3\}$$

- Note that $\mathcal{P}_r(I^n) \subset \mathcal{S}_r(I^n) \subset \mathcal{Q}_r(I^n)$

Example: $\mathcal{S}_2(I^2)$ Basis Functions

$$\begin{pmatrix} N_0 \\ N_1 \\ N_2 \\ N_3 \\ N_4 \\ N_5 \\ N_6 \\ N_7 \end{pmatrix} = \begin{pmatrix} -\frac{(x-1)(y-1)(x+y+1)}{4} \\ \frac{(x^2-1)(y-1)}{2} \\ \frac{(x+1)(y-1)(y-x+1)}{4} \\ -\frac{(y^2-1)(x+1)}{2} \\ \frac{(x+1)(y+1)(x+y-1)}{4} \\ -\frac{(x^2-1)(y+1)}{2} \\ \frac{(x-1)(y+1)(x-y+1)}{4} \\ \frac{(y^2-1)(x-1)}{2} \end{pmatrix}$$

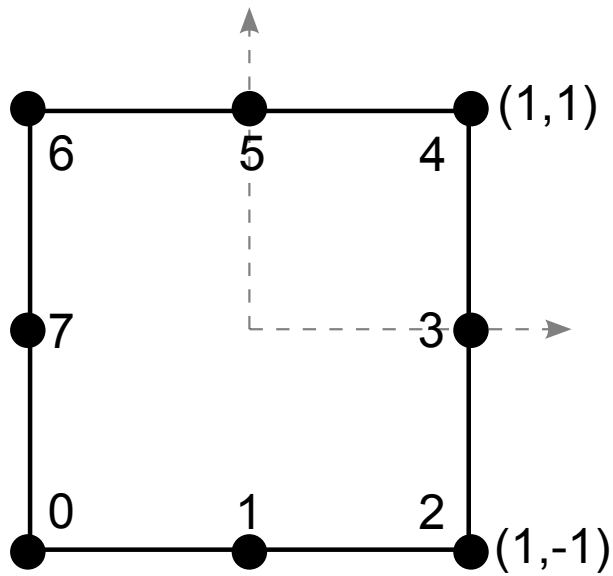


Figure: $\mathcal{S}_2(I^2)$ element centered at origin

Generating Serendipity Basis Functions

1. Given an order and dimension, determine the monomials v_k that span the serendipity space
2. Determine location of nodes $\mathbf{x_k}$ on reference element
3. Set up matrix equation for basis functions

$$N_j(\mathbf{x}) = \sum_k c_j^{(k)} v_k(\mathbf{x})$$

$$N_j(\mathbf{x_k}) = \delta_{kj}$$

4. Solve system for weights $c_j^{(k)}$ by a matrix inversion
5. Map reference space basis functions to physical space

3-D Gaussian Pulse Advection

- Advection at constant speed in 3-D

$$\frac{\partial f}{\partial t} + \nabla \cdot (f\mathbf{u}) = 0$$

- Solution is given by

$$f(\mathbf{x}, t) = f_0(\mathbf{x} - \mathbf{u}t)$$

- Periodic boundary conditions
- Design simulation to end when pulse has completed one period of its motion
- Compute error *per node* by summing over all nodes in domain:

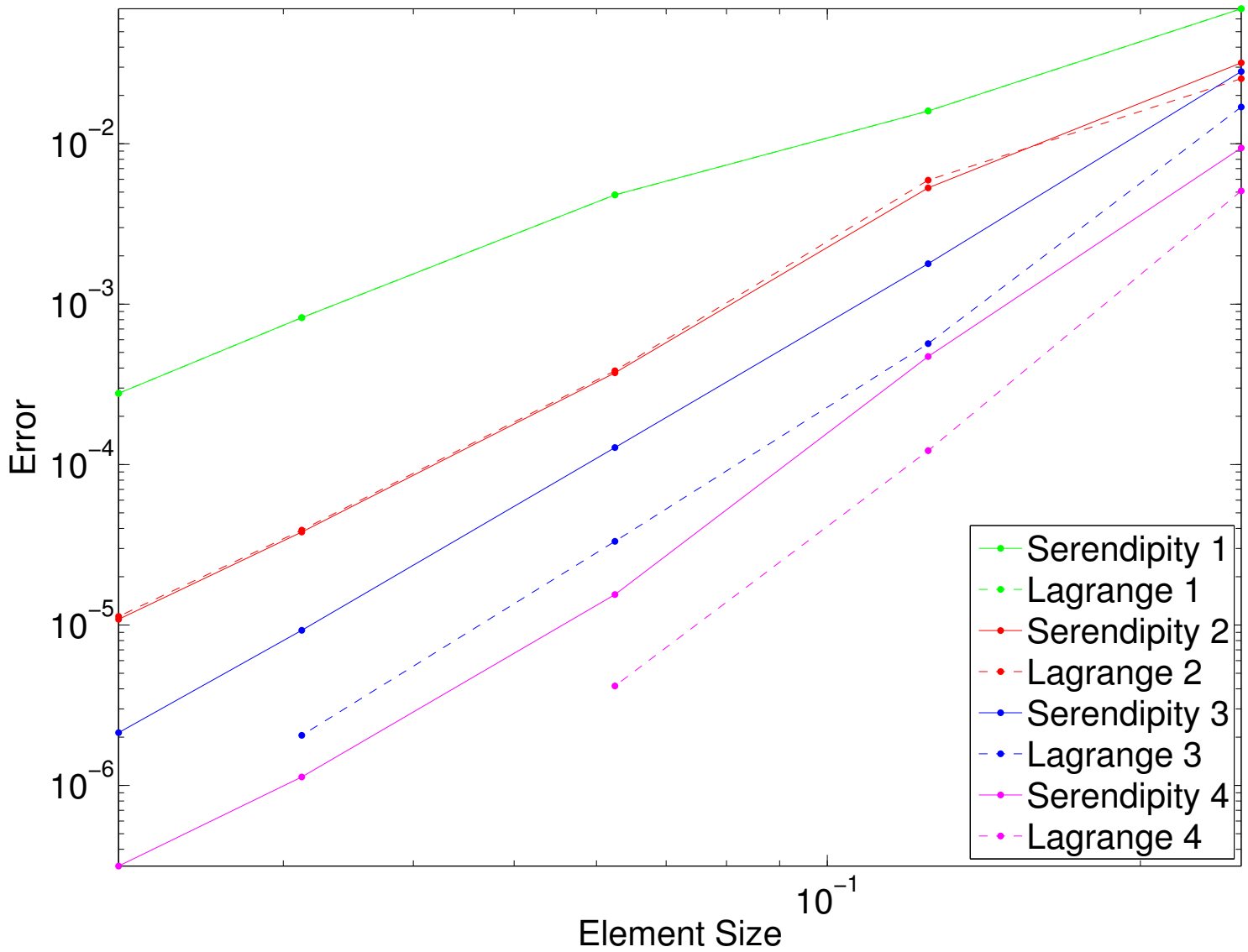
$$E = \sum_{k}^{N_{nodes}} \frac{|u(t_f, \vec{x_k}) - u(t_0, \vec{x_k})|}{N_{nodes}}$$

Convergence Study

- Order of serendipity element and total number of elements varied ($n \times n \times n$ for $n = 4, 8, 16, 32$)
- Since error E scales as $O(h^c)$ with element size h, plot $\log E$ vs. $\log h$ and find slope of best-fit line

Order of Element	Order of Scheme (S)	Order of Scheme (L)
1	2.09	2.09
2	3.30	3.20
3	3.85	4.31
4	4.40	5.13

Error vs. Element Size



Computation Time

- Execution time results for a $32 \times 32 \times 32$ element grid:

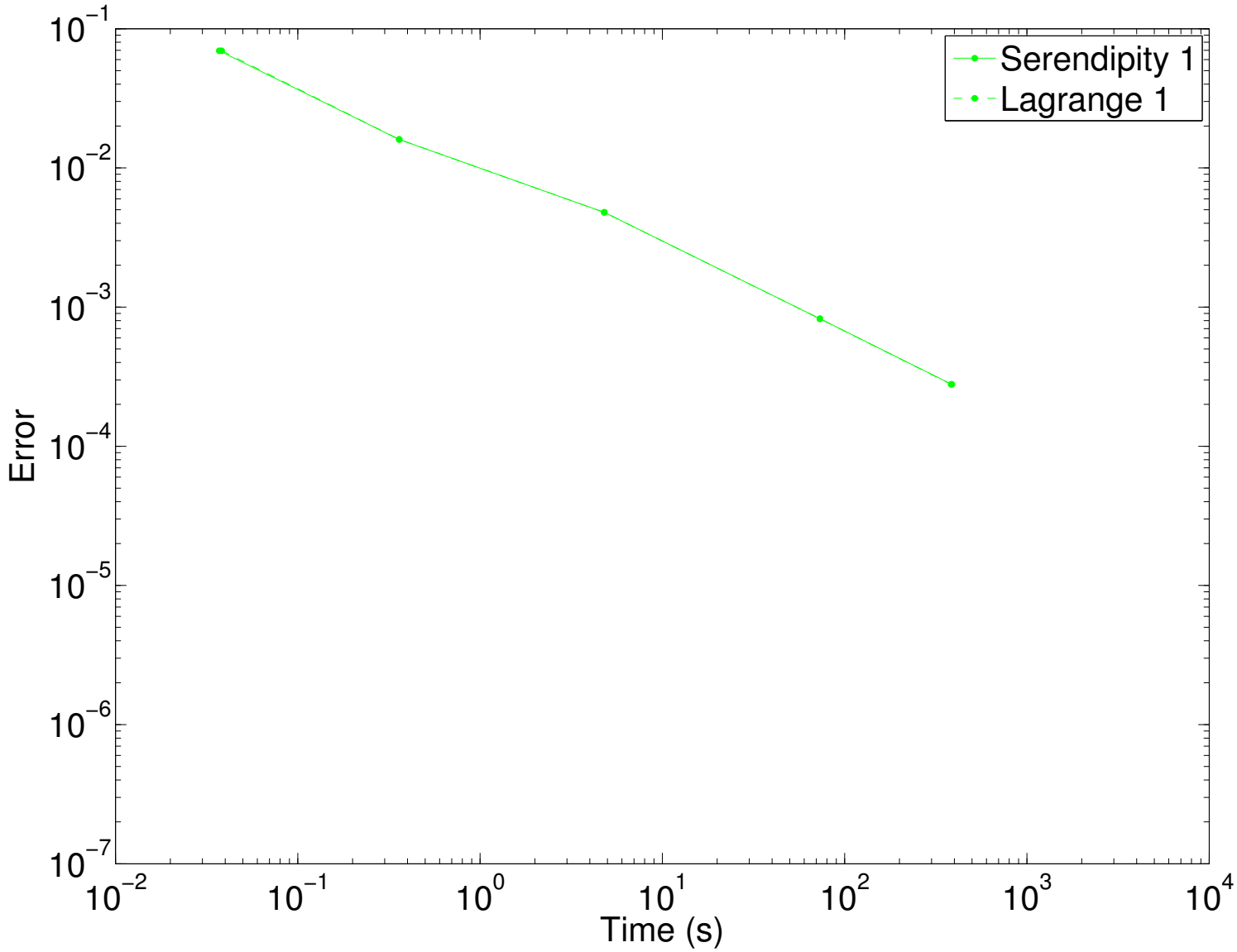
Order of Element	Serendipity	Lagrange	Ratio
1	73.2 s	73.2 s	1.00
2	220 s	663 s	0.331
3	654 s	5350 s	0.122

- DOF per element comparison:

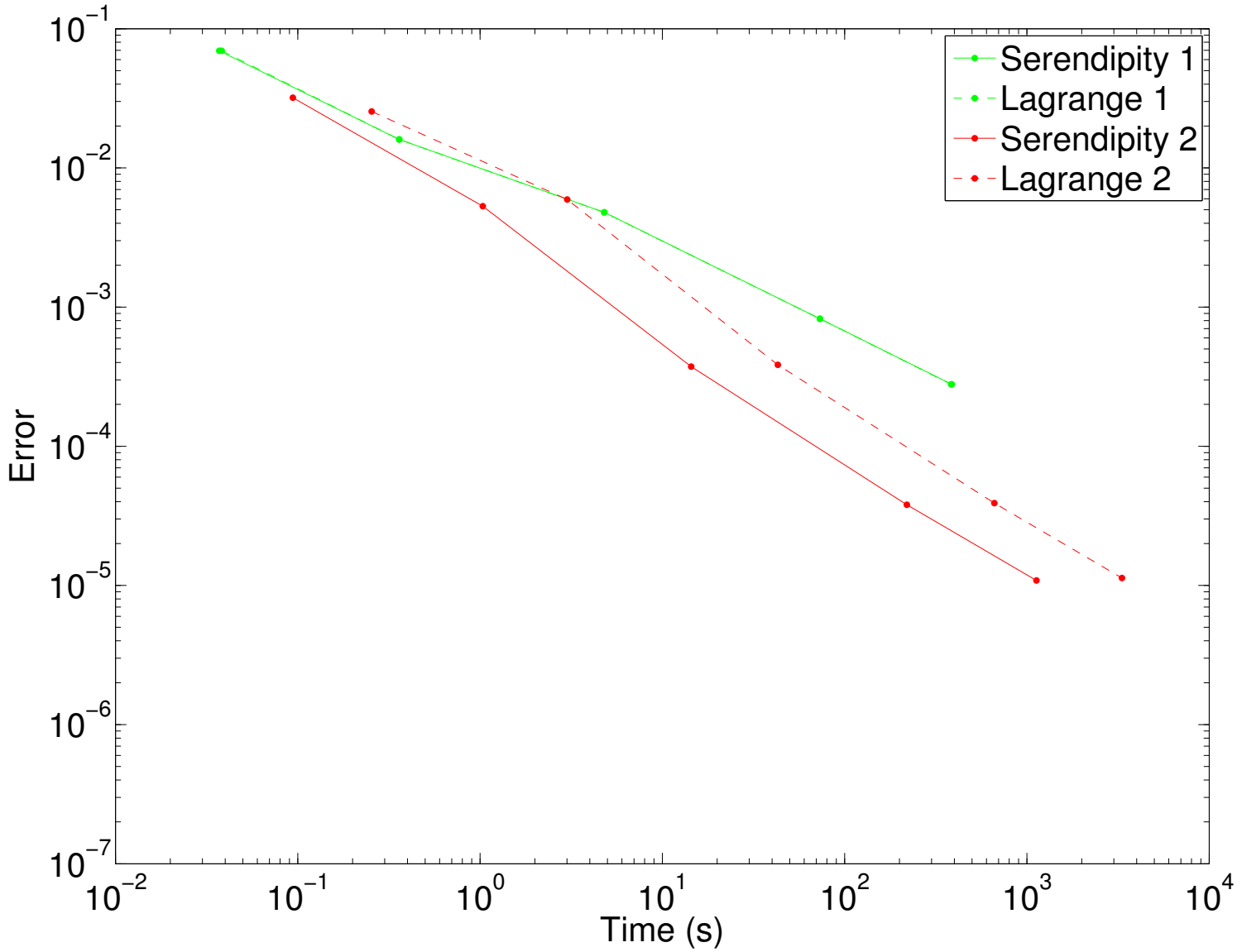
Order of Element	Serendipity	Lagrange	Ratio
1	8	8	1.00
2	20	27	0.741
3	32	64	0.500

Additional savings due to larger permissible time step for serendipity elements

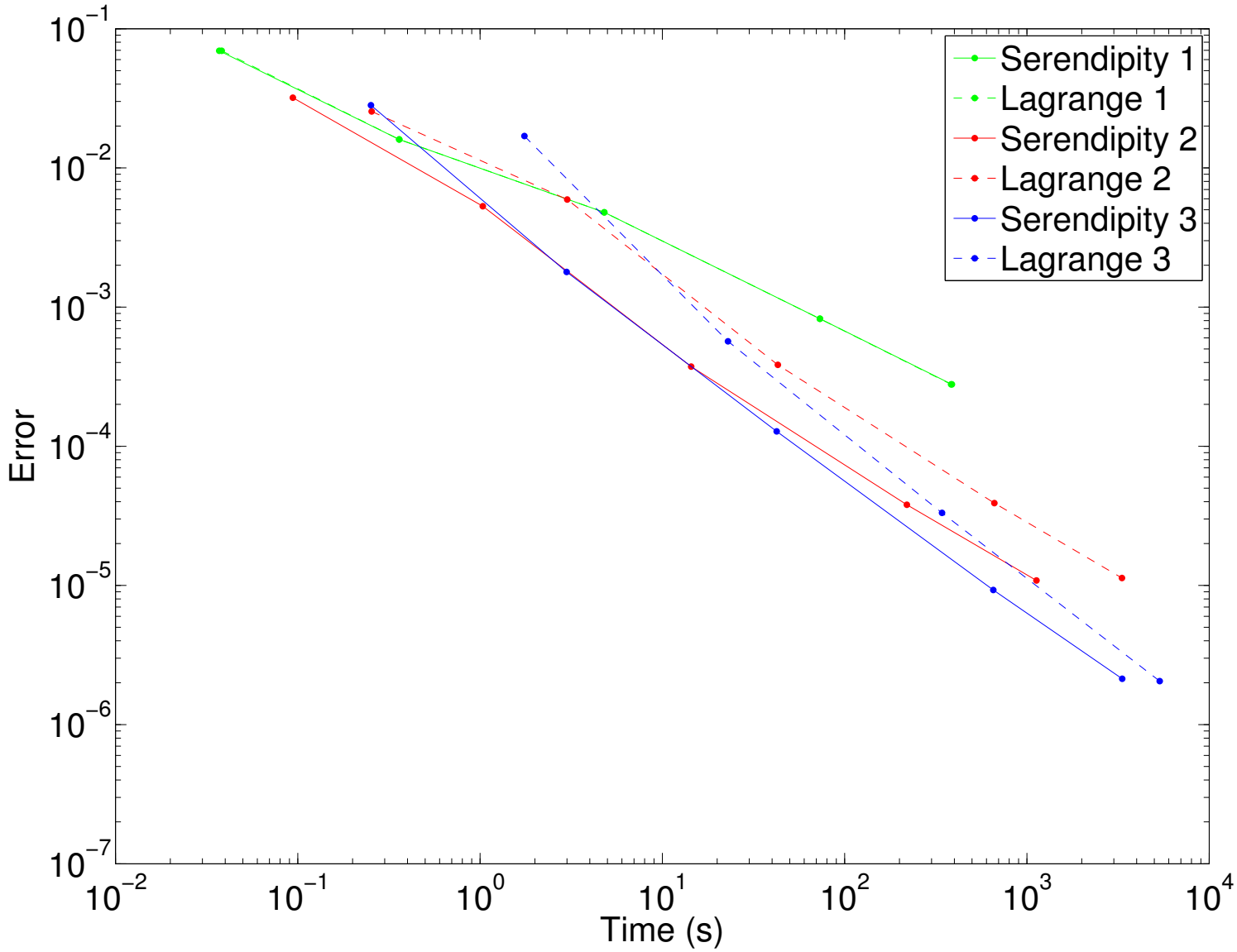
Error vs. Computation Time



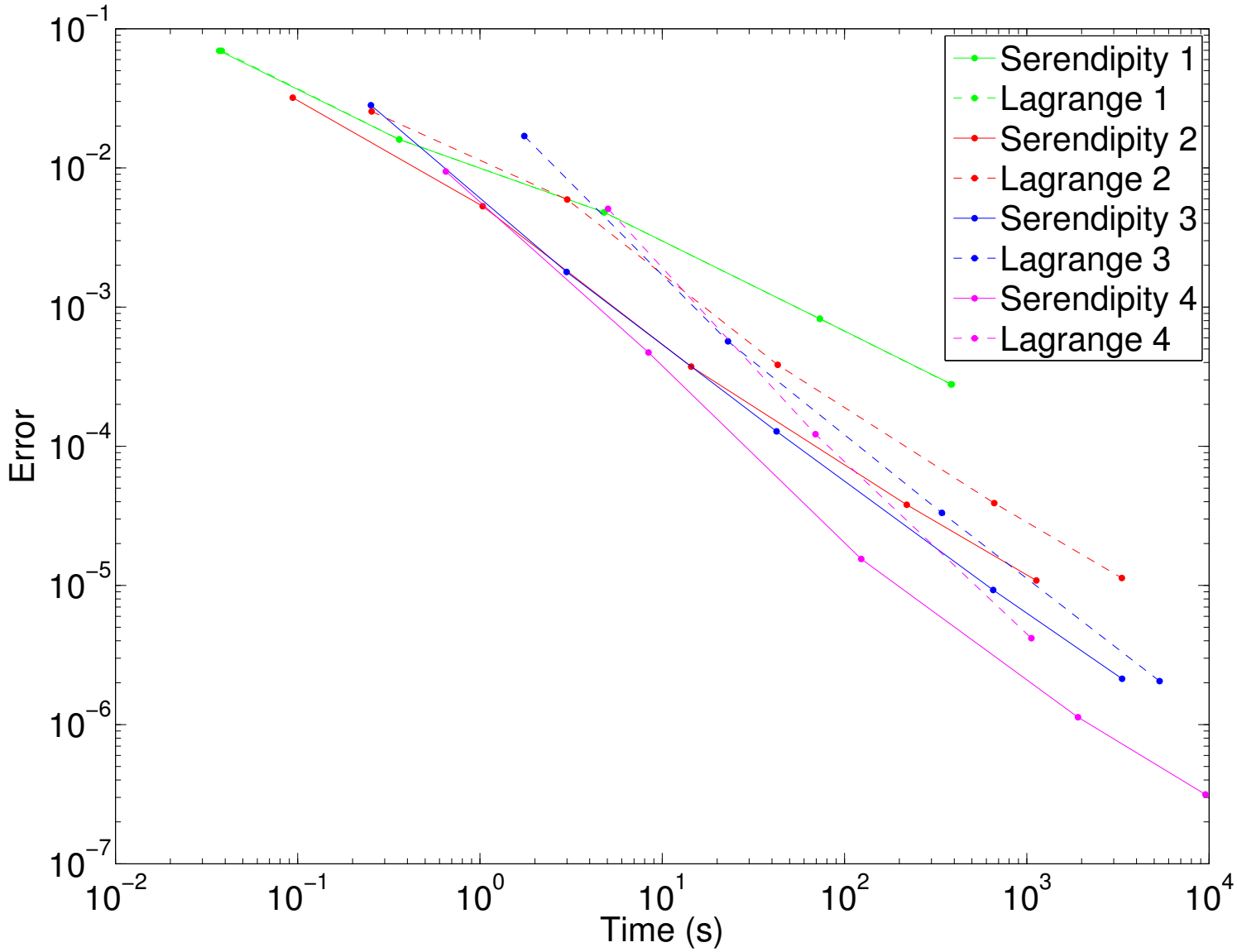
Error vs. Computation Time



Error vs. Computation Time



Error vs. Computation Time



Conclusions

Initial results indicate that:

- Serendipity elements are more efficient than Lagrange elements for obtaining the same level of error
- Additional savings in computational time from larger maximum permissible time step for convergence (CFL condition)

Future Work

- Investigate performance of serendipity elements in multi-block structured grids
- Use serendipity elements in 1D2V simulations
  - Scrape-off layer model
- Add full Lenard-Bernstein collision operator
- Extend serendipity elements in Gkeyll to 4-D and 5-D
- Investigate Maxwellian-weighted basis functions for velocity space discretization

Acknowledgements

Thanks to:

- Jeffrey Parker
- Chang Liu

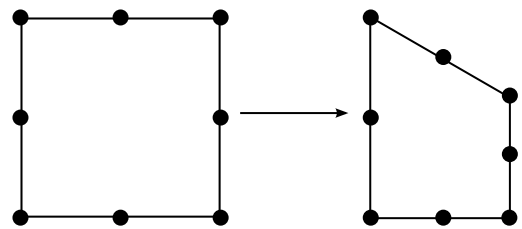
Alpha testing:

- Sam Cohen
- Allan Reiman

Caveats

- Reference element in results was mapped to the physical space by scaling each dimension by a constant
  - This is an example of an **affine** map

- Unfortunately, serendipity elements do not attain the same optimal rate of convergence on non-affine meshes



- Can avoid this problem by defining basis functions in physical space