Basis construction techniques for serendipity-type spaces

Andrew Gillette
University of Arizona

joint work with
Tyler Kloefkorn, National Academy of Sciences
Victoria Sanders, University of Arizona
Table of Contents

1  Serendipity methods: a review of their potential
2  Recent mathematical advances in serendipity theory
3  POEMS techniques for generic quad and hex elements
4  Explicit bases and implementation plans
Outline

1. Serendipity methods: a review of their potential
2. Recent mathematical advances in serendipity theory
3. POEMS techniques for generic quad and hex elements
4. Explicit bases and implementation plans
The original “serendipity phenomenon”

Finite element method for $\Delta u = 0$.
Boundary data: $\sin(x) \ e^y$
Domain: $[0, 3]^2$, with $\ell \times \ell$ square grid.
Code: MATLAB

| $\ell$ | DoFs | $||u - u_h||_2$ | ratio | order | $||\nabla u - \nabla u_h||_2$ | ratio | order |
|-------|------|-----------------|-------|-------|----------------|-------|-------|
| 2     | 25   | 4.2029e-01      | 1.000 | 0.000 | 2.4006e+00      | 1.000 | 0.000 |
| 4     | 81   | 5.7476e-02      | 9.376 | 3.229 | 5.3156e-01      | 4.516 | 2.175 |
| 8     | 289  | 7.3802e-03      | 8.155 | 3.028 | 1.2947e-01      | 4.106 | 2.038 |
| 16    | 1089 | 9.2909e-04      | 8.002 | 3.000 | 3.2221e-02      | 4.018 | 2.007 |
| 32    | 4225 | 1.1635e-04      | 7.986 | 2.997 | 8.0455e-03      | 3.997 | 1.999 |

| $\ell$ | DoFs | $||u - u_h||_2$ | ratio | order | $||\nabla u - \nabla u_h||_2$ | ratio | order |
|-------|------|-----------------|-------|-------|----------------|-------|-------|
| 2     | 21   | 5.6921e-01      | 0.000 | 0.000 | 2.4006e+00      | 0.000 | 0.000 |
| 4     | 65   | 6.0711e-02      | 9.376 | 3.229 | 5.3156e-01      | 4.516 | 2.175 |
| 8     | 225  | 7.4447e-03      | 8.155 | 3.028 | 1.2947e-01      | 4.106 | 2.038 |
| 16    | 833  | 9.3040e-04      | 8.002 | 3.000 | 3.2221e-02      | 4.018 | 2.007 |
| 32    | 3201 | 1.1637e-04      | 7.995 | 2.999 | 8.0491e-03      | 4.003 | 2.001 |
The original “serendipity” phenomenon

Finite element method for $\Delta u = 0$.
Boundary data: $\sin(x) e^y$
Domain: $[0, 3]^2$, with $\ell \times \ell$ square grid.
Code: MATLAB

How much of a savings in DoFs can we get for large $r$?
Serendipity per-element DoF savings grow with $r$

→ DoFs per $Q_r$ (scalar) element in dim $n = (r + 1)^n$
→ DoFs per $S_r$ (scalar) element in dim $n = \mathcal{O}(r^n / n!)$
→ In 2D, for large $r$, $Q_r$ has $\approx 2$ times as many DoFs per element as $S_r$
→ In 3D, for large $r$, $Q_r$ has $\approx 5.8$ times as many DoFs per element as $S_r$, including more than 2 times as many DoFs shared between elements!
Additional potential savings for solvers

Patch-based solvers depend on a stencil of DoFs around each vertex in a mesh. Stencils for $\mathcal{P}_3$ on a triangular mesh and $\mathcal{S}_3$ on a quad mesh are shown.

↪ from a proposal with Rob Kirby (Baylor U.); currently under review

**Ex:** In 3D, a $\mathcal{Q}_5$ patch has $\approx 12$ times the number of DoFs as a $\mathcal{S}_5$ patch

$\implies$ a quadratic order complexity solver with $\mathcal{Q}_5$ patches would have $\approx 144$ times longer run times than one with $\mathcal{S}_5$ patches!

*Takeaway:* robustly implementing serendipity elements should allow significant reduction in computational cost with no loss in order of accuracy.
1. Serendipity methods: a review of their potential

2. Recent mathematical advances in serendipity theory

3. POEMS techniques for generic quad and hex elements

4. Explicit bases and implementation plans
Two key insights from Arnold and Awanou

→ Scalar serendipity elements exist for any order \( r \geq 1 \) in any dimension \( n \geq 2 \).


\[
\begin{align*}
\text{r = 2} & \quad \text{r = 3} & \quad \text{r = 4} & \quad \text{r = 5} & \quad \text{r = 6} & \quad \ldots \\
\end{align*}
\]

→ Scalar serendipity elements are part of a family of finite element differential forms.


Ex: \( S_1 \Lambda^2(\Box_3) \) is an element for

- \( r = 1 \rightarrow \) linear order of error decay
- \( k = 2 \rightarrow \) conformity in \( \Lambda^2(\mathbb{R}^3) \sim H(\text{div}) \)
- \( n = 3 \rightarrow \) domains in \( \mathbb{R}^3 \)
Classification of many common conforming finite element types.

\[ n \rightarrow \text{Domains in } \mathbb{R}^2 \text{ (top half) and in } \mathbb{R}^3 \text{ (bottom half)} \]

\[ r \rightarrow \text{Order } 1, 2, 3 \text{ of error decay (going down columns)} \]

\[ k \rightarrow \text{Conformity type } k = 0, \ldots, n \text{ (going across a row)} \]

Geometry types: Simplices (left half) and cubes (right half).
Method selection and cochain complexes

\[ \Rightarrow \]

\[ \text{Provably stable method converges to } u = x(1 - x)y(1 - y) \]

Stable pairs of elements for mixed Hodge-Laplacian problems are found by choosing consecutive spaces in compatible discretizations of the \( L^2 \) deRham Diagram.

\[ \begin{align*} H^1 \xrightarrow{\nabla \text{ grad}} & H(\text{curl}) \xrightarrow{\nabla \times \text{ curl}} H(\text{div}) \xrightarrow{\nabla \cdot \text{ div}} L^2 \\ \text{vector Poisson} & \sigma \mu \text{ Maxwell’s eqn’s} \ h \ b \text{ Darcy / Poisson} \ u \ p \end{align*} \]

Stable pairs are found from consecutive entries in a cochain complex.
Cochain complexes occur either horizontally or diagonally in the table as shown.

Methods can be chosen from $\mathcal{P}$ or $\mathcal{P}^-$ (simplices) and $\mathcal{Q}^-$ or $\mathcal{S}$ (cubes).

Mysteriously, the DoF count for mixed methods from the $\mathcal{P}^-$ spaces is smaller than those from the $\mathcal{P}$ spaces, while the opposite is true for $\mathcal{Q}^-$ and $\mathcal{S}$ spaces.
The 5th column: Trimmed serendipity spaces

A new column for the PToFE: the **trimmed serendipity** elements.

\[ S_r^{-} \Lambda^k(\square^n) \]

denotes approximation order \( r \),
subset of \( k \)-form space \( \Lambda^k(\Omega) \),
use on meshes of \( n \)-dim’l cubes.

Defined for any \( n \geq 1, 0 \leq k \leq n, r \geq 1 \)

Identical or analogous properties to all the other columns in the table.

**Computational advantage:**
Fewer DoFs for mixed methods than both tensor product and serendipity counterparts.

Outline

1. Serendipity methods: a review of their potential
2. Recent mathematical advances in serendipity theory
3. POEMS techniques for generic quad and hex elements
4. Explicit bases and implementation plans
Correct usage on unstructured quad/hex meshes

Quadratic serendipity elements, mapped non-affinely, are only expected to converge at the rate of linear elements.

Similar problems for all elements in the serendipity families!

One way out: use VEM serendipity!

→ The VEM serendipity spaces $VEMS_{r,r,r-1}$ on quads have the same degree of freedom counts as the trimmed serendipity spaces $S_{r+1}^{-} \Lambda^1(\Box_2)$

→ Similar equivalences hold between other VEM serendipity spaces and other (trimmed) serendipity spaces.

→ Going the VEM route means giving up on local basis functions.

Beirão da Veiga, Brezzi, Marini, Russo “Serendipity face and edge VEM spaces”
Another way out: basis functions on physical elements

→ Define basis functions $\psi_{ij}$ on physical elements:

$$ u_h = I_q u := \sum_{i=1}^{n} u(v_i) \psi_{ii} + u \left( \frac{v_i + v_{i+1}}{2} \right) \psi_{i(i+1)} $$

→ Hard to generalize and compute beyond quadratic order

Non-affine bilinear mapping

| n  | $||u - u_h||_{L^2}$ error | $||\nabla (u - u_h)||_{L^2}$ error |
|----|----------------------------|----------------------------------|
| 2  | 5.0e-2                     | 6.2e-1                           |
| 4  | 6.7e-3 2.9                 | 1.8e-1 1.8                       |
| 8  | 9.7e-4 2.8                 | 5.9e-2 1.6                       |
| 16 | 1.6e-4 2.6                 | 2.3e-2 1.4                       |
| 32 | 3.3e-5 2.3                 | 1.0e-2 1.2                       |
| 64 | 7.4e-6 2.1                 | 4.96e-3 1.1                      |

Physical element basis functions:

| n  | $||u - u_h||_{L^2}$ error | $||\nabla (u - u_h)||_{L^2}$ error |
|----|----------------------------|----------------------------------|
| 2  | 2.34e-3                    | 2.22e-2                          |
| 4  | 3.03e-4 2.95              | 6.10e-3 1.87                     |
| 8  | 3.87e-5 2.97              | 1.59e-3 1.94                     |
| 16 | 4.88e-6 2.99              | 4.04e-4 1.97                     |
| 32 | 6.13e-7 3.00              | 1.02e-4 1.99                     |
| 64 | 7.67e-8 3.00              | 2.56e-5 1.99                     |

A finite element space on a general quadrilateral is built in two parts:

- Apply Piola mapping to functions associated to boundary of reference element.
- Define functions on the physical element corresponding to interior degrees of freedom in a way that ensures relevant polynomial approximation properties.

1. Serendipity methods: a review of their potential

2. Recent mathematical advances in serendipity theory

3. POEMS techniques for generic quad and hex elements

4. Explicit bases and implementation plans
Goal: find a computational basis for $S_1 \Lambda^1(\square_3)$:

- Must be $H(\text{curl})$-conforming
- Must have 24 functions, 2 associated to each edge of cube
- Must recover constant and linear approx. on each edge
- The approximation space contains:
  
  1. Any polynomial coefficient of at most linear order:
     \[
     \{1, x, y, z\} \times \{dx, dy, dz\} \rightarrow 12 \text{ forms}
     \]
  2. Certain forms with quadratic or cubic order coefficients shown in table at left \rightarrow 12 forms

- For constants, use “obvious” functions:

\[
\{(y \pm 1)(z \pm 1)dx, (x \pm 1)(z \pm 1)dy, (x \pm 1)(y \pm 1)dz\}
\]

  e.g. $(y + 1)(z + 1)dx$ evaluates to zero on every edge except $\{y = 1, z = 1\}$ where it is $\equiv 4 \rightarrow$ constant approx.

Also, $(y + 1)(z + 1)dx$ can be written as a linear combo, by using the first three forms at left to get the $yz \ dx$ term.
Building a computational basis

- For constant approx on edges, we used:
  \{ (y \pm 1)(z \pm 1)dx, (x \pm 1)(z \pm 1)dy, (x \pm 1)(y \pm 1)dz \}

- Guess for linear approx on edges:
  \{ x(y \pm 1)(z \pm 1)dx, y(x \pm 1)(z \pm 1)dy, z(x \pm 1)(y \pm 1)dz \}
  
  e.g. \( x(y + 1)(z + 1)dx \) evaluates to \( 4x \) on \( \{ y = 1, z = 1 \} \).

- Unfortunately:
  \( x(y + 1)(z + 1)dx \not\in S_1 \Lambda(\Box_3)! \)
  
  Why? \( x(y + 1)(z + 1)dx = (xyz + xy + xz + x)dx \)
  
  but \( xyz \, dx \) only appears with other cubic order coefficients!

- Remedy: add \( dy \) and \( dz \) terms that vanish on all edges.
Building a computational basis

Computational basis element associated to \(\{y = 1, z = 1\}\):

\[
2x(y + 1)(z + 1) \, dx + (z + 1)(x^2 - 1) \, dy + (y + 1)(x^2 - 1) \, dz
\]

✓ Evaluates to 4\(x\) on \(\{y = 1, z = 1\}\) (linear approx.)
✓ Evaluates to 0 on all other edges
✓ Belongs to the space \(S_1 \Lambda(\Box_3)\):

\[
\begin{align*}
2xyz \, dx & + x^2z \, dy & + x^2y \, dz \\
2xy \, dx & + x^2 \, dy & + 0 \, dz \\
x2z \, dx & + 0 \, dy & + x^2 \, dz \\
2x \, dx & + (z - 1)dy & + (y - 1)dz
\end{align*}
\]

linear order

\[\Rightarrow\] summation and factoring yields the desired form)

There are 11 other such functions, one per edge. We have:

\[
S_1 \Lambda(\Box_3) = E_0 \Lambda^1(\Box_3) \oplus \tilde{E}_1 \Lambda^1(\Box_3)
\]

“obvious” basis for constant approx
modified basis for linear approx

\[
dim 24 = 12 + 12
\]
## A complete table of computational bases

<table>
<thead>
<tr>
<th>( n = 3 )</th>
<th>( k = 0 )</th>
<th>( k = 1 )</th>
<th>( k = 2 )</th>
<th>( k = 3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>( S_r \Lambda^k )</strong></td>
<td>( V \Lambda^0(\square_3) )</td>
<td>( \emptyset )</td>
<td>( \emptyset )</td>
<td>( \emptyset )</td>
</tr>
<tr>
<td>( r-2 )</td>
<td>( \bigoplus E_i \Lambda^0(\square_3) )</td>
<td>( \bigoplus E_i \Lambda^1(\square_3) )</td>
<td>( \emptyset )</td>
<td>( \emptyset )</td>
</tr>
<tr>
<td>( i=0 )</td>
<td>( \bigoplus F_i \Lambda^0(\square_3) )</td>
<td>( \bigoplus F_i \Lambda^1(\square_3) )</td>
<td>( \bigoplus F_i \Lambda^2(\square_3) )</td>
<td>( \emptyset )</td>
</tr>
<tr>
<td>( r )</td>
<td>( \bigoplus l_i \Lambda^0(\square_3) )</td>
<td>( \bigoplus l_i \Lambda^1(\square_3) )</td>
<td>( \bigoplus l_i \Lambda^2(\square_3) )</td>
<td>( \bigoplus l_i \Lambda^3(\square_3) )</td>
</tr>
<tr>
<td><strong>( S_r^- \Lambda^k )</strong></td>
<td>( V \Lambda^0(\square_3) )</td>
<td>( \emptyset )</td>
<td>( \emptyset )</td>
<td>( \emptyset )</td>
</tr>
<tr>
<td>( r-2 )</td>
<td>( \bigoplus E_i \Lambda^0(\square_3) )</td>
<td>( \bigoplus E_i \Lambda^1(\square_3) )</td>
<td>( \emptyset )</td>
<td>( \emptyset )</td>
</tr>
<tr>
<td>( i=0 )</td>
<td>( \bigoplus F_i \Lambda^0(\square_3) )</td>
<td>( \bigoplus F_i \Lambda^1(\square_3) )</td>
<td>( \bigoplus F_i \Lambda^2(\square_3) )</td>
<td>( \emptyset )</td>
</tr>
<tr>
<td>( r )</td>
<td>( \bigoplus l_i \Lambda^0(\square_3) )</td>
<td>( \bigoplus l_i \Lambda^1(\square_3) \oplus \tilde{\tau}_r \Lambda^1(\square_3) )</td>
<td>( \bigoplus l_i \Lambda^2(\square_3) \oplus \tilde{\tau}_r \Lambda^2(\square_3) )</td>
<td>( \bigoplus l_i \Lambda^3(\square_3) )</td>
</tr>
</tbody>
</table>

Open source finite element software packages I

→ open source C++ program library for adaptive FEM, in development since 1998
→ designed to support quad/hex meshes and $h/p$ adaptivity
→ data structures are well-documented but not easy to introduce new element types without in-depth knowledge of the code

→ FEM toolkits that use Unified Form Language to define a weak form and create local assembly kernels
→ FEniCS passes kernels to DOLFIN’s C++ libraries and PETSc to do solves
→ Firedrake creates intermediate data structures that are then passed to parallel schedulers, including notions like “dofs” and “interior facet” that more easily accommodate extensibility

First pass at Firedrake implementation

→ Scalar-valued, 2D square elements only (so far!)
→ Replaced “monomial” parts of basis with Legendre polynomials.
→ Laplace problem with boundary data: \( \cos(\pi x) \cos(\pi y) \)
→ Domain: \([0, 1]^2\), with \(\ell \times \ell\) square grid.
→ Code: Firedrake, with Krylov solver options

![Graph showing convergence of solution error with increasing global DoFs](image)

<table>
<thead>
<tr>
<th>Global DoFs</th>
<th>(|u - u_h|_2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10^{10}</td>
<td>10^{-3}</td>
</tr>
<tr>
<td>10^{11}</td>
<td>10^{-6}</td>
</tr>
<tr>
<td>10^{12}</td>
<td>10^{-9}</td>
</tr>
<tr>
<td>10^{13}</td>
<td>10^{-12}</td>
</tr>
<tr>
<td>10^{14}</td>
<td>10^{-15}</td>
</tr>
</tbody>
</table>

Andrew Gillette - U. Arizona
Bases for serendipity spaces
POEMs 2019

25 / 27
**MFEM:**

Modular Finite Element Methods library

→ “free, lightweight, scalable C++ library for FE methods,” developed at Lawrence Livermore National Labs since 2010

→ emphasis placed on high-order methods, parallelizability, and support for variety of techniques

→ supports lab missions in studies of hydrodynamics, magnetostatics, fusion, turbulence, etc.

I will be working with the MFEM team at LLNL this summer to (begin to) implement serendipity elements in their package!

**Pictures from** [mfem.org/gallery](http://mfem.org/gallery)

---

Acknowledgments

*Merci to the organizers for the invitation!*

Collaborators on this work

- Rob Kirby  
  Baylor University
- Tyler Kloefkorn  
  National Academies Program Officer, Math  
  *(former postdoc)*
- Victoria Sanders  
  U. Arizona  
  *(undergrad math major)*

Research Funding

Supported in part by the National Science Foundation grant DMS-1522289.

Slides and Pre-prints

[http://math.arizona.edu/~agillette/](http://math.arizona.edu/~agillette/)