Bringing Serendipity Methods to Computational Practice in Firedrake

CYRUS CHENG, Imperial College, United Kingdom
JUSTIN CRUM, University of Arizona
ANDREW GILLETTE, University of Arizona
DAVID HAM, Imperial College, United Kingdom
ROBERT KIRBY, Baylor University
JOSHUA A. LEVINE, University of Arizona
LAWRENCE MITCHELL, Durham University, United Kingdom

An abstract about Firedrake and FEM here.

ACM Reference Format:
Cyrus Cheng, Justin Crum, Andrew Gillette, David Ham, Robert Kirby, Joshua A. Levine, and Lawrence Mitchell. 2018. Bringing Serendipity Methods to Computational Practice in Firedrake. 1, 1 (December 2018), 9 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

2 BACKGROUND ON SERENDIPITY AND TRIMMED SERENDIPITY ELEMENTS

2.1 2D Elements

(1) Scalar (classical = Arnold-Awanou = $\mathcal{S}_r\Lambda^0(\mathbb{R}^2)$)
(2) Vector Serendipity (BDM = Arnold-Awanou = $\mathcal{S}_r\Lambda^1(\mathbb{R}^2)$)
(3) Vector Trimmed Serendipity (Arbogast-Correa = Gillette-Kloefkorn = $\mathcal{S}_r^-\Lambda^1(\mathbb{R}^2)$)
(4) Direct (Arbogast-Tao / Arbogast-Correa)

2.1.1 *Scalar (classical = Arnold-Awanou = $\mathcal{S}_r\Lambda^0(\mathbb{R}^2)$).*

2.1.2 *Vector Serendipity (BDM = Arnold-Awanou = $\mathcal{S}_r\Lambda^1(\mathbb{R}^2)$).*

2.1.3 *Vector Trimmed Serendipity (Arbogast-Correa = Gillette-Kloefkorn = $\mathcal{S}_r^-\Lambda^1(\mathbb{R}^2)$).*

2.1.4 *Direct (Arbogast-Tao / Arbogast-Correa).*

Authors' addresses: Cyrus Cheng, cyrus.cheng15@imperial.ac.uk, Imperial College, London, United Kingdom; Justin Crum, jcrum@math.arizona.edu, University of Arizona, Tucson, Arizona; Andrew Gillette, University of Arizona, Tucson, Arizona, agillette@math.arizona.edu; David Ham, Imperial College, London, United Kingdom, david.ham@imperial.ac.uk; Robert Kirby, Baylor University, Waco, Texas, robert_kirby@baylor.edu; Joshua A. Levine, University of Arizona, josh@email.arizona.edu; Lawrence Mitchell, Durham University, Durham, United Kingdom, wence@gmx.li.

2.2 3D Elements

(1) Scalar (classical = Arnold-Awanou = $\mathcal{S}_r\Lambda^0(\mathbb{R}^3)$)
(2) Vector serendipity (Arnold-Awanou = $\mathcal{S}_r\Lambda^1(\mathbb{R}^3)$ and $\mathcal{S}_r\Lambda^2(\mathbb{R}^3)$
(3) Vector trimmed serendipity (Gillette-Kloefkorn = $\mathcal{S}_r^-\Lambda^1(\mathbb{R}^3)$ and $\mathcal{S}_r^-\Lambda^2(\mathbb{R}^3)$)

3 BUILDING CAPACITY FOR SERENDIPITY ELEMENT TYPES IN FIREDRAKE

4 EXPERIMENTS

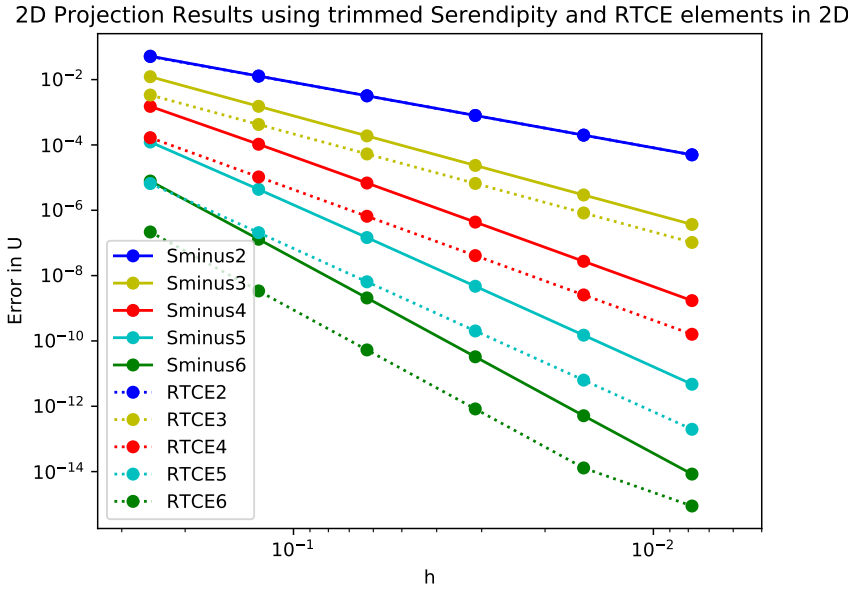
The following experiments were designed to show the benefits and costs to using trimmed Serendipity elements in comparison to traditional tensor product elements. We used a basic projection example to check implementation, as well as a primal Poisson problem (to test scalar elements), a mixed Poisson problem (to test H(div) elements), and a cavity resonator problem (to test H(curl) elements). Note that the cavity resonator problem was done only in 3D, as in 2D, the H(curl) elements are just a rotation of the H(div) elements.

4.1 Projection

We solve the projection problem to give a baseline of expectations for the elements.

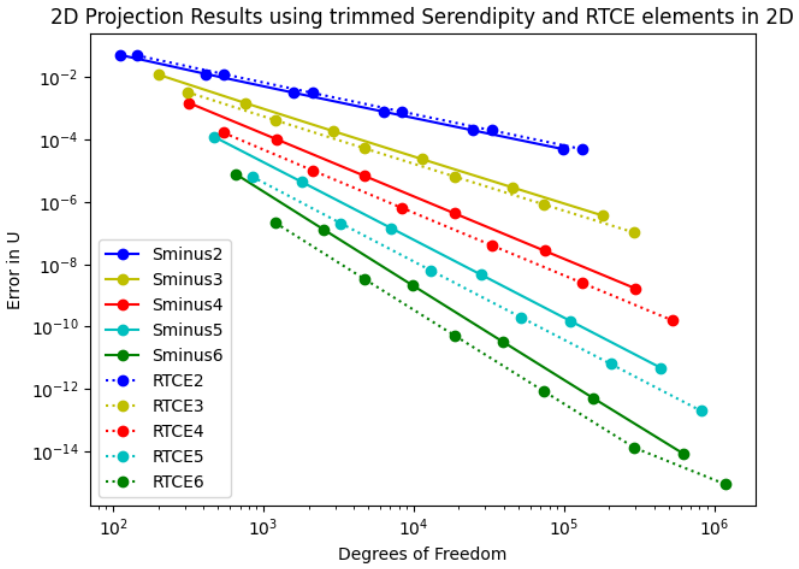
GET THE EXACT PROBLEM THAT THE PROJECTION PROBLEM IS SOLVING.

Fig. 1. Analysis of projection using S^-(Curl) and RTCE.



First, we graphed the errors from computing projections using trimmed Serendipity and tensor product elements in 2D, shown in 2 **AG:** Always say Figure [ref], or Table [ref] not just the reference. Since we expect that the order of convergence for both of these elements should be the same (only off by a constant factor), we look for the lines representing trimmed Serendipity and tensor product elements at a given order to be parallel.

Fig. 2. Degrees of Freedom vs Error analysis of projection using S^-(Curl) and RTCE.



After checking that it is converging at the right rate, we then want to analyze the data from a standpoint of memory usage. To do this, we plot degrees of freedom vs error, as seen in **AG:** Figure 2. Overall, the error given by trimmed Serendipity elements is higher than the error from using tensor product elements of the same order. However, the degrees of freedom comparison allows us to instead compare error per degree of freedom, which tends to favor trimmed Serendipity elements at higher degrees. To see this in a few more pratical (but still relatively simple problems), we move on to the next few experiments.

4.2 Primal Poisson

We solve the primal Poisson problem described below on a unit square domain Ω

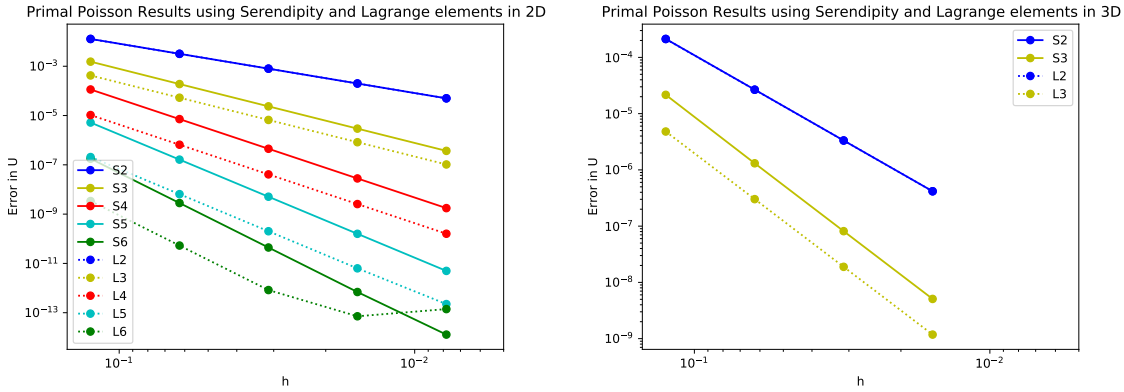
$$\begin{aligned} -\nabla^2 u &= 2\pi^2 \sin(\pi x)\sin(\pi y) \in \Omega \\ u|_{\partial\Omega} &= 0 \\ \nabla u \cdot n &= 0 \text{ on } \partial\Omega \end{aligned} \tag{1}$$

which yields the solution $u(x, y) = \sin(\pi x)\sin(\pi y)$. In 3D, we can extend this to

$$-\nabla^2 u = 2\pi^2 \sin(\pi x)\sin(\pi y)\sin(\pi z)$$

with the solution being $u(x, y) = \sin(\pi x)\sin(\pi y)\sin(\pi z)$. **AG:** Write the equation as $-\nabla^2 u = f$ then state what f is afterwards; this is the typical presentation style

As in the projection problem, we first give graphs **AG:** Call them Figures, with a capital F, not graphs, to be consistent 3b and 3a to illustrate the behavior of these elements with respect to mesh refinement. In general, Lagrange elements again



(a) 2D (b) 3D

Fig. 3. Analysing the error based off the mesh refinements for the primal Poisson problem in 2 and 3D.

give better error with respect to mesh refinement (ignoring order 6 in 2D when getting close to machine precision). In 3D, we focus on orders 2 and 3, as anything higher than this requires a prohibitive amount of time to run the Lagrange elements. **AG:** Be careful about claiming something *requires* a prohibitive amount of time - it would probably be easy to run on e.g. an LLNL supercomputer. Give a ballpark of how much time and/or memory it was taking on the machine you were using - was it maxing out one or the other? Qualify this a bit more.

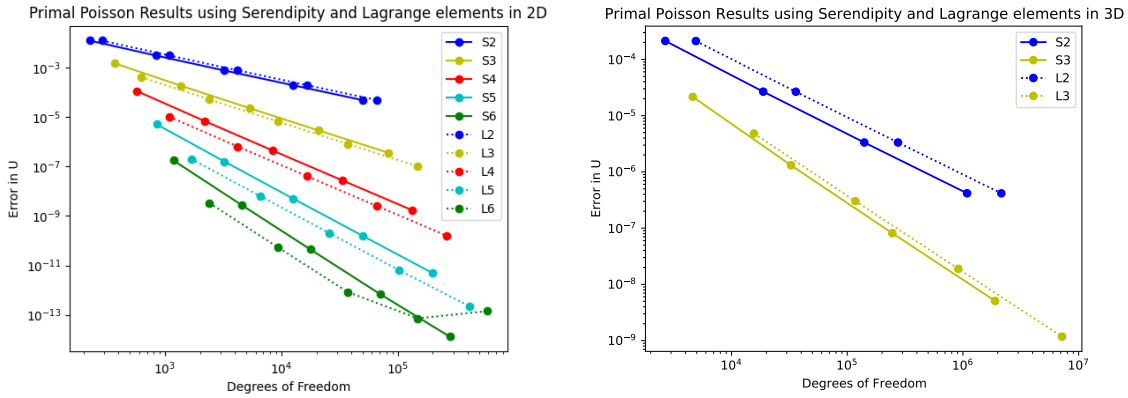
Though the Lagrange elements tend to give a better error approximation at each mesh refinement level, we also wish to investigate the cases of how much error can we get if we instead wish to limit the amount of memory or time that the machine uses. In **AG:** Figures 4a and 4b, we have graphs **AG:** trendlines? (instead of "graphs") illustrating the degrees of freedom in the mesh vs the error. What we find is that at the lower orders (order 2 for 2D and both orders 2 and 3 for 3D), the Serendipity curves show a better error at the same degrees of freedom, while the Lagrange elements overtake them at higher orders.

Finally, we look at the timing requirements for the primal Poisson problem using the Lagrange and Serendipity elements in 3D. In 5, we can see that the amount of time required for Serendipity elements tends to start off higher, but scales better as the problem grows in size.

4.3 Mixed Poisson

The mixed Poisson problem that we focus on is derived from the primal Poisson equation above. Specifically, we solve a discretization of **AG:** Put this as an `align` environment so it aligns the equals signs; you can use `\notag` at the end of a line if you don't want an equation number - I'll try to find an example for you if you haven't seen it before. Also prevents a group of equations like this from being split across a page break

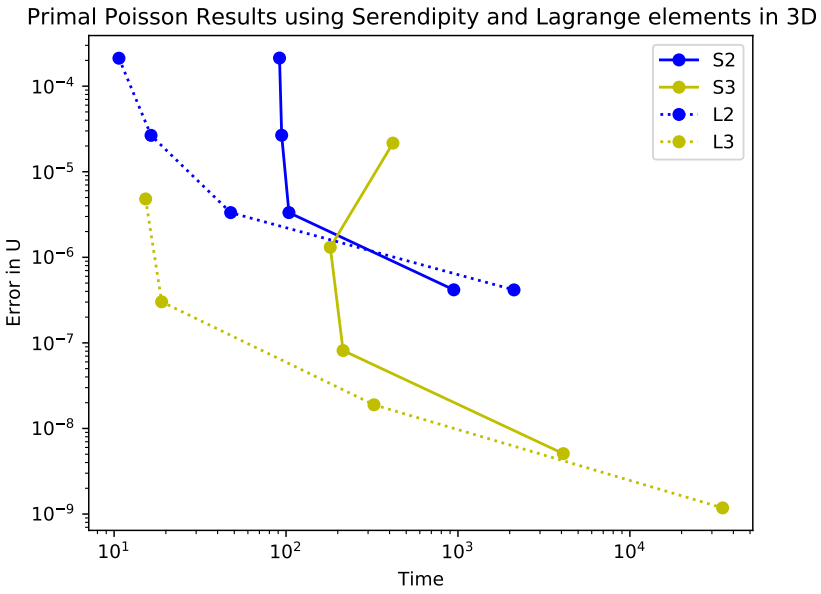
$$\begin{aligned} \sigma - \nabla u &= 0 \\ \nabla \cdot \sigma &= -f \\ u|_{\partial\Omega} &= 0 \end{aligned} \tag{2}$$



(a) 2D (b) 3D

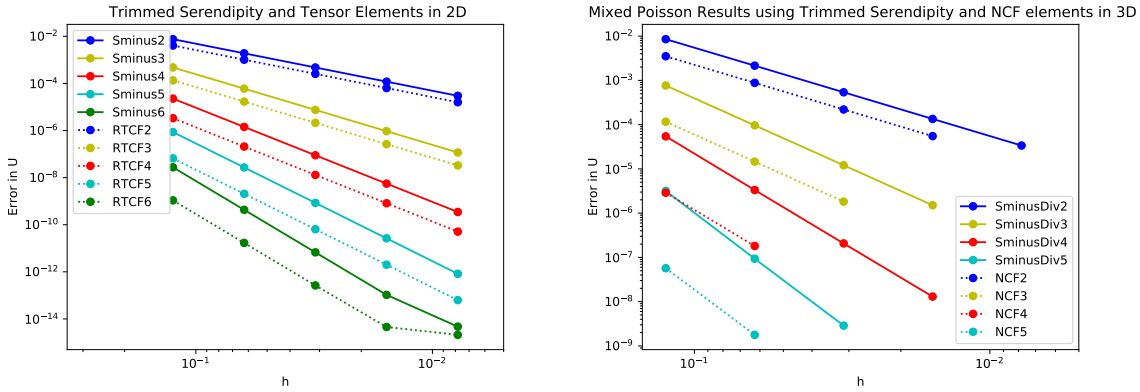
Fig. 4. Analyzing the primal Poisson problem in 2 and 3D based off degrees of freedom vs error.

Fig. 5. Timing data for the 3D primal Poisson problem, where each sequence of data represents using the respective element for a mesh of size $NxNxN$ for $N = 8, 16, 32, 64$.



which has the same analytic 2 and 3D solutions as the primal Poisson problem. **AG:** What is f? The error results based off mesh refinements can be seen in **AG:** Figures 6a and 6b. The 2D mixed Poisson graph is similar to the 2D primal Poisson graph. The 3D mixed Poisson graph again illustrates the proper rates of convergence for trimmed Serendipity

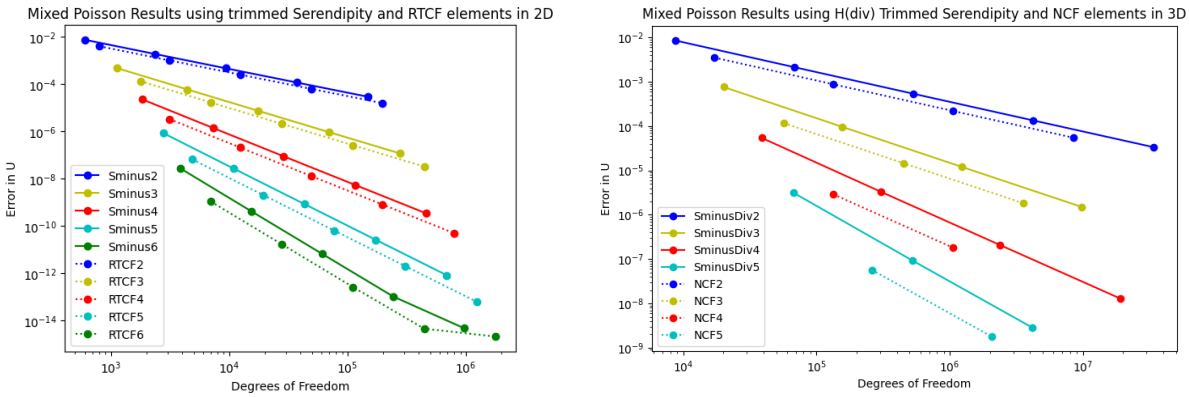
elements, but also illustrates the computational difficulty of scaling the tensor product elements to larger problems in 3D.



(a) 2D (b) 3D

Fig. 6. Analyzing the mixed Poisson problem in 2 and 3D based off edge length h vs error.

After the comparison based off edge length, we also want to focus on the error based on the degrees of freedom. In **AG:** Figures 7a and 7b, we see similar trends to our previous graphs.



(a) 2D (b) 3D

Fig. 7. Analyzing the mixed Poisson problem in 2 and 3D based off degrees of freedom vs error.

4.4 Cavity Resonator

AG: I rephrased slightly but good job on this: For testing the H(curl) elements in 3D, we use the cavity resonator problem. Here, the time-harmonic Maxwell equations are applied to a domain $\Omega = [0, 1]^3$ with perfectly conducting boundary conditions, yielding an eigenvalue problem where ω represents the resonances (i.e. eigenvalues) and E represents the electric field (i.e. eigenfunctions):

$$\langle \text{curl}(F), \text{curl}(E) \rangle = \omega^2 \langle F, E \rangle \text{ for all } F \in H_0(\text{curl}). \tag{3}$$

The exact eigenvalues should follow the formula

$$\omega^2 = m_1^2 + m_2^2 + m_3^2$$

where $m_i \in \mathbb{N} \cup 0$ and no more than one of m_1, m_2, m_3 may be equal to 0 at a time [?].

In Table 1, we look at the convergence rates of different eigenvalues based off solving the problem with tensor product (NCE) elements and trimmed Serendipity (S^-) elements in 3D. The table is split into two mirrored halves, the top half giving values from using NCE elements while the bottom half gives values from using S^- elements. The column labeled "Actual" represents the theoretical eigenvalue that eigenvalues in that row are converging towards. Each of the $N = 4, N = 8, N = 16, N = 32$ columns represents the approximate eigenvalues calculated on a mesh of size $NxNxN$. The number in parenthesis next to approximate eigenvalues is the rate of convergence of that eigenvalue. Finally, each half has a row giving the overall degrees of freedom in the mesh at each given mesh size and another row that gives the time that the eigenvalue solver needed to find the requested number of eigenvalues.
Note that the convergence rates are computed by doing

$$r = \frac{\log\left(\frac{\tilde{\lambda}_{i,N} - \lambda_{i,N}}{\tilde{\lambda}_{i,N+1} - \lambda_{i,N+1}}\right)}{\log\left(\frac{h_N}{h_{N+1}}\right)}$$

and are indicated in the chart by using parentheses. We use H(curl) to solve the problems, corresponding with edge elements in 3D. Based off earlier eigenvalue works [1], we expect that the rate of convergence be double the order of the finite element used to solve the problem. This is reflected in the table in most spots, except for one of the eigenvalues of 5, where it tends to oscillate a bit. This specific eigenvalue overall converges at a rate near 4.00 if we instead using the values at $N = 8$ and $N = 32$, ignoring the intermediate value at $N = 16$.

Based off the closed form formula, note that 4 should never be an eigenvalue in 3D. However, we see it consistently found by both the NCE and S^- elements. Any eigenvalue that has a – spot is to be interpreted as the eigenvalue solver did not find that specific eigenvalue in the number of iterations it required to find the first 15 requested eigenvalue-eigenvector pairs.

Knowing that both elements are solving this problem in a fashion that is expected theoretically, we can analyze the rest of the results shown in this table. Investigating the error in the eigenvalues in the chart compared to the exact values, we see that NCE elements are able to get results that are up to a magnitude better, though not for every eigenvalue.

NCE Elements				
Actual	N = 4	N = 8	N = 16	N = 32
2	2.0010243	2.0000655 (3.97)	2.0000041 (3.99)	2.0000003 (4.00)
2	2.0010243	2.0000655 (3.97)	2.0000041 (3.99)	2.0000003 (4.00)
2	2.0010243	2.0000655 (3.97)	2.0000041 (3.99)	2.0000003 (4.00)
3	3.0015364	3.0000983 (3.97)	3.0000062 (3.99)	3.0000004 (4.00)
3	3.0015364	3.0000983 (3.97)	3.0000062 (3.99)	3.0000004 (4.00)
! → 4	4.0300893	4.0020486 (3.88)	4.0001311 (3.97)	4.0000082 (3.99)
! → 4	4.0300893	4.0020486 (3.88)	4.0001311 (3.97)	4.0000082 (3.99)
5	5.0306014	5.0020813 (3.88)	5.0001331 (3.97)	5.0000084 (3.99)
5	5.0306014	5.0020813 (3.88)	5.0001331 (3.97)	5.0000084 (3.99)
5	5.0306014	5.0020813 (3.88)	5.0001331 (3.97)	5.0000084 (3.99)
5	-	-	5.0001331	5.0000084 (3.99)
6	-	6.0021141	6.0001352 (3.97)	-
6	-	6.0021141	6.0001352 (3.97)	-
DOF	1944	13872	104544	811200
EPS Solve Time (seconds)	0.0514	0.2811	3.2795	29.2620
S^- H(curl) Elements				
Actual	N = 4	N = 8	N = 16	N = 32
2	2.0010919	2.0000664 (4.04)	2.0000041 (4.01)	2.0000003 (4.00)
2	2.0010919	2.0000664 (4.04)	2.0000041 (4.01)	2.0000003 (4.00)
2	2.0059537	2.0003900 (3.93)	2.0000247 (3.98)	2.0000015 (4.00)
3	3.0090182	3.0005863 (3.94)	3.0000370 (3.98)	3.0000023 (4.00)
3	3.0090182	3.0005863 (3.94)	3.0000370 (3.98)	3.0000023 (4.00)
! → 4	4.0300893	4.0020486 (3.88)	4.0001311 (3.97)	4.0000082 (3.99)
! → 4	4.0300893	4.0020486 (3.88)	4.0001311 (3.97)	4.0000082 (3.99)
5	5.0320266	5.0020970 (3.93)	5.0001333 (3.98)	5.0000084 (3.99)
5	5.0320266	5.0020970 (3.93)	5.0001333 (3.98)	5.0000084 (3.99)
5	5.0736899	5.0052310 (3.82)	5.0001333 (5.29)	5.0000212 (2.65)
5	5.0736899	5.0052310 (3.82)	5.0003371 (3.96)	5.0000212 (3.99)
6	-	6.0049765	6.0003192 (3.96)	6.0000201 (3.99)
6	-	-	-	-
DOF	1080	7344	53856	411840
EPS Solve Time (seconds)	0.0367	0.1339	1.5125	17.6765

Table 1. A comparison of how order 2 NCE and S^- finite elements solve the Maxwell cavity resonator eigenvalue problem, $\langle \mathrm{curl}(F), \mathrm{curl}(E) \rangle = \omega^2 \langle F, E \rangle$.

However, this loss of accuracy from the trimmed Serendipity elements is made up for by the comparison of the DOFs and solve time required. At every mesh refinement level, trimmed Serendipity elements have nearly half the DOFs of NCE elements, and correspondingly, require about half the time to solve for the eigenvalues. At higher orders, we expect that this will be even more exaggerated.

The experiment was done by using SLEPc in Firedrake, computing an inverted shift to a target of 3.0, then asking SLEPc for 15 eigenvalue-eigenvector pairs. SLEPc was then give a tolerance level of 1e − 7, and then a couple of specific mumps parameters (icntl 14 set to 200 and icntl 13 set to 1). We ignored the eigenvalues of 1, as they correspond only to the boundary conditions.

5 CONCLUSION

Each finite element has a time and place where it could be considered beneficial to use. We explored the numerical properties of trimmed Serendipity elements to refine our understanding of how they act. From the theory, we knew that trimmed Serendipity elements should be able to converge at the same rate as tensor product and Serendipity elements, while using fewer degrees of freedom overall. The plots here demonstrate that the rate of convergence is consistent with what we expect at many orders for H(curl), H(div), and L^2 elements in both 2 and 3D.

Beyond the convergence rates hitting what we expect, we were able to analyze memory usage on these problems by studying the degrees of freedom required. Trimmed Serendipity, while generally have a worse error at a given order k, also uses significantly fewer degrees of freedom. This is illustrated well in the Maxwell Cavity Eigenvalue problem in 1, where the degrees of freedom required were nearly half of what the tensor product elements used.

Another example of this is the 3D mixed Poisson problem, where we see that the trimmed Serendipity elements are able to be used at more refined meshes while the tensor product elements would need to be allotted more time on a high memory computer to be able to get results on the same sized mesh.

In general, it is clear from these results that trimmed Serendipity is not always a better choice compared to tensor product elements. However, these examples illustrates the benefit of trimmed Serendipity elements–in a setting where the mesh is fixed, the option to use a trimmed Serendipity element might give an extra way to refine a problem to increase the accuracy of a solution.

REFERENCES

[1] Daniele Boffi. 2010. Finite element approximation of eigenvalue problems. *Acta Numer.* 19 (2010), 1–120.

[] Marie E Rognes, Robert C Kirby, and Anders Logg. 2010. Efficient assembly of H(div) and H(curl) conforming finite elements. *SIAM Journal on Scientific Computing* 31, 6 (2010), 4130–4151.