Replicated Computational Results (RCR) Report for “Code Generation for Generally Mapped Finite Elements”

NEIL LINDQUIST, Saint John’s University, USA

“Code Generation for Generally Mapped Finite Elements” includes performance results for the finite element methods discussed in that manuscript. The authors provided a Zenodo archive with the Firedrake components and dependencies used, as well as the scripts that generated the results. The software was installed on two similar platforms; then, new results were gathered and compared to the original results. After completing this process, the results have been deemed replicable by the reviewer.

CCS Concepts: • General and reference → Verification; • Mathematics of computing → Mathematical software performance; Solvers;

Additional Key Words and Phrases: Finite element, partial differential equations, replicated computational results

ACM Reference format:
Neil Lindquist. 2019. Replicated Computational Results (RCR) Report for “Code Generation for Generally Mapped Finite Elements”. ACM Trans. Math. Softw. 45, 4, Article 42 (December 2019), 7 pages. https://doi.org/10.1145/3360984

1 INTRODUCTION

The results replication effort for “Code Generation for Generally Mapped Finite Elements,” by Robert Kirby and Lawrence Mitchell, focused on Section 4 of the manuscript, which provides example results and comparisons between the newly implemented finite element transformations and the existing Lagrange transformation. The original results were obtained using an Intel E5-2640v3 processor; however, the replication was done using two different machines, an Intel E5-2698v4 and an Intel E5-2609v4, designated node0 and node1, respectively.

2 REPLICATING THE RESULTS

The exact version of the software used to generate the results was archived on Zenodo [Firedrake-Zenodo 2019]. This archive contains the versions of almost all of the Firedrake components used, the scripts that generate the data, and the resulting data files. The one dependency not included in the archive is SciPy, which is required to generate the sparsity graphs. The original archive had a few configuration issues, but the authors worked with the reviewer to correct the issues, which resulted in the aforementioned version of the archive.

Author’s address: N. Lindquist, Saint John’s University, Collegeville, MN; email: nslindquist@csbsju.edu.
Author’s current address: N. Lindquist, Innovative Computing Laboratory, University of Tennessee, Department of Electrical Engineering and Computer Science, 1122 Volunteer Blvd, Suite 203 Claxton, Knoxville, TN 37996.
Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2019 Association for Computing Machinery.
0098-3500/2019/12-ART42 $15.00
https://doi.org/10.1145/3360984
2.1 Installation
Because Firedrake is able to configure an installation based on the Zenodo archive, installation was straightforward. The machines used to replicate the results shared a file system and operating system image, and so, Firedrake was installed from node0 then reused from node1. The installation script for Firedrake normally ensures the correct system dependencies are present using `apt-get`. However, `apt-get` was not present on the replication machines. So, the system dependencies had to be manually verified, including

- GNU make 3.82
- cmake 2.8.12.2
- gcc and gfortran 4.8.5
- MPICH 3.2
- Python 3.7.3

Installation was then accomplished by running the following commands from a bash shell. Note that the `--no-package-manager` and `--disable-ssh` flags were added by the reviewer to the recommended installation. The `--no-package-manager` was required, since `apt-get` was not present on the replication machines. Git was not configured correctly to clone repositories over ssh. So, the `--disable-ssh` flag was added to use https from the start, instead of attempting to use ssh first for every repository.

```
$ export PETSC_CONFIGURE_OPTIONS="--download-pastix --download-ptsscotch"
$ python3 firedrake --install --doi 10.5281/zenodo.3234784 --slepc
   --no-package-manager --disable-ssh --install git+https://github.com/ThomasGibson/scpc.git@3a11792bb3610dcdf5090088294a542274bb73a
$ chmod +x firedrake/bin/activate
$ firedrake/bin/activate
$ firedrake/bin/pip install scipy
```

2.2 Execution
The various results were generated from the 13 Python scripts in the Zenodo archive. On each machine, first `firedrake/bin/activate` was run in the bash shell to ensure the correct virtual environment was used. Then, each of the scripts was run using the Firedrake installation’s Python. Note that both `chladni.py` and `guitar.py` both have command line configurable options. The default values matched those described in the article, except for the number of eigenvalues to generate. So, `-eps_nev 30` was passed to those scripts.

The output was a mixture of pdf and csv files. The correspondence between the output files and the results presented in the manuscript is obvious except for the timing results. For the timing results, each discretization has its own csv file containing the solution error and various execution times for each mesh size. The timing results presented in the manuscript are the time to assemble the discretization (in column SNESJacobianEval), the time to factor and solve the linear system (in column KSPSolve), and the total solver time (in column SNESSolver).

3 EVALUATION OF REPLICATED RESULTS
There were two problems used to compare the newly implemented transformations with the existing transformation, the second-degree Poisson’s equation and the fourth-degree biharmonic equation. Because different hardware was used for the original results and the replicated results,
the timing results were only judged on whether the relative performances are similar. For the other results, only differences that can be explained by round-off error were considered acceptable. Additionally, there were two example problems, the Cahn-Hilliard equation and the Chladni plate problem. These example problems each generate a set of images. These images were expected to look similar, with minor differences considered acceptable.

3.1 Comparison of Model Problems

The FLOP counts and sparsity for assembling the linear systems should both be deterministic computations with limited floating point operations, so both should be highly replicable. The FLOP counts generated by each machine are exactly equal to those provided in the archived data and match up with the graphs in the manuscript. Similarly, the generated sparsity patterns were visually indistinguishable with the ones in the archive and the manuscript.

Figure 9(a) in the manuscript contains the $L^2$ error of the Poisson equation. The $L^2$ errors in the generated results were all within $1.75 \times 10^{-14}$ of the archived data. Similarly, for the $L^2$ error of the biharmonic equation, presented in Figure 9(b) in the manuscript, the errors of the generated results were all within $3.25 \times 10^{-15}$ of the archived data, except for the error of $P^5$ when $N = 2^7$, which was off by $6.87 \times 10^{-13}$. These errors are reasonable for accumulated floating point round-off error, and so are considered replicated.

Figure 11 in the manuscript shows the time to assemble the Laplace and biharmonic operators. The manuscript notes that for the Poisson system “Hermite and $P^3$, and Argyris and $P^5$ elements require very similar assembly time, with Bell somewhat higher than $P^4$.” Figure 1 shows the replication performance of assembling the Laplace operator. For the biharmonic operator, it is noted that a “clear win for $H^2$ elements” is shown, with the replication performance shown in Figure 2. Both sets of replication results agree with the manuscript’s claims of relative performance. Additionally, the graphs look similar with the only difference is that the time to assembly of the $H^2$ elements is slightly lower, which is acceptable.

In the manuscript, Figure 14 shows the time to factor and solve the linear system and the overall solver time for the Poisson equation. For the Poisson equation, the manuscript notes that “Hermite and Bell systems are actually cheaper to solve than any of the $P^k$ systems, and Argyris appears to have a comparable cost to $P^4$ rather than $P^5$” and that “condensation improves the total run-time
For the Poisson problem, the time to factor and solve the linear system in Figure 3, and the overall solve time in Figure 4. As can be seen in the figures, the claims about the performance of the Argyris and Bell systems for the Poisson equation do not hold in the replicated data. However, the rest of the performance claims do hold, and the graphs otherwise look similar. Note that the time to factor and solve the linear equation makes up almost the entirety of the total solve time. Thus, this difference in performance likely comes from differences in either the linear systems or the configuration and runtime of the linear solver. Because the previous parts of the comparison were replicated successfully, a difference in the runtime of the linear solver is more likely the cause of this issue. Furthermore, the conclusion
only states that the "numerical results indicate that these elements are viable" and "of comparable cost to deploy as any Lagrange elements," meaning that the replicated performance still supports those conclusions. Hence, the performance results were considered to be successfully replicated. Similarly, Figure 15 in the manuscript shows the time to factor and solve the linear system, as well as the overall solver time, for the biharmonic equation. The manuscript merely states that "solution of the Argyris element typically being cheaper than even the $P^3$ interior penalty method, not to mention $P^5$." The time to factor and solve the linear system is shown in Figure 5, and the overall solver time is shown in Figure 6. Unlike the performance of the Laplacian operator, the biharmonic operator performed close to the reported results.
3.2 Example Problems

The Cahn-Hilliard example produced only the images of the initial and final states. Additionally, the initial state is generated with a fixed randomization seed, so directly comparing the resulting images is possible. The initial state and final state images generated by each machine are indistinguishable from the respective images in the Zenodo archive and in the manuscript.

The Chladni plate problems each provide a series of images that depict various Chladni figures for the respective plate shapes. For the square plate, both machines generated almost the same set of eigenpairs with a few minor differences from the versions in the archive and manuscript. First, there are three eigenfunctions with eigenvalues of 0. These eigenfunctions are present in the archived results but were not generated by either replication machine. However, these eigenfunctions are not shown in the manuscript, so this issue was not considered to affect the replicability. Second, two of the eigenfunctions rendered slightly differently than shown in the manuscript and archived data. These eigenfunctions correspond to the same eigenvalue and are rotations of each other, so only one is discussed here. Figure 7 shows both of the generated images and the archived image for one of those eigenfunctions. Note that the exact value of this eigenfunction is similar to the images presented in Figures 7(b) and 7(c), except the diagonal is continuous [Ritz 1909]. Given that the replicated result is more accurate than the image in the manuscript, this difference was not considered to affect the replicability of the manuscript.
In addition to computing the Chladni figures on a square plate, figures were computed for a guitar-shaped region. The Zenodo archive only contains images for the first 28 eigenfunctions, which are shown in the manuscript. However, with the \texttt{-epsnev 30} flag, there are an extra 6 eigenfunctions computed that were not present in the archive; these images match between the two replication machines but were otherwise ignored for this review. For the other 28 eigenfunctions, both sets of generated images match the images in the archive and manuscript.

4 CONCLUDING REMARKS

The reviewer attempted to replicate the figures and results in Section 4 of the manuscript. The installation and execution of the software was straightforward using the Zenodo archive listed in the manuscript. Furthermore, the results generated were reasonably close to those in the manuscript. Thus, the reviewer deemed the published results to be replicable.

ACKNOWLEDGMENTS

The reviewer would like to thank Lawrence Mitchell for providing guidance on installing and using the software, and Michael Heroux for providing guidance on the RCR review process.

REFERENCES

Firedrake-Zenodo. 2019. Software used in “Code generation for generally mapped finite elements.” DOI: https://doi.org/10.5281/zenodo.3381901

Walter Ritz. 1909. Theorie der transversalschwingungen einer quadratischen platte mit freien Rändern. Ann. Der Physik 333, 4 (1909), 737–786. DOI: https://doi.org/10.1002/andp.19093330403

Received June 2019; accepted September 2019