FEniCSx Preconditioning Tools (FEniCSx-pctools)

Martin Řehoř*   Jack S. Hale†

November 28, 2023

Abstract

FEniCSx Preconditioning Tools (FEniCSx-pctools) is a software package for easing the specification of PETSc-based block preconditioning strategies in the DOLFINx finite element solver of the FEniCS Project. It attaches all of the necessary metadata to the block-structured linear systems in order that block-structured preconditioners can be applied straightforwardly via PETSc's options-based configuration system. Fast prototyping is facilitated thanks to the implementation in Python, and all intensive operations are executed in C/C++. FEniCSx-pctools is available under the LGPLv3 or later license.

Introduction

Overview

FEniCSx Preconditioning Tools (FEniCSx-pctools) is a Python package for easing the specification of complex PETSc-based block-structured linear solvers in the DOLFINx finite element software of the FEniCS Project. FEniCSx-pctools was produced as a supporting tool in a larger research project focused on fluidic topology optimisation.

A classic example of a block-structured system is the symmetric saddle point problem stemming from the $H(\mathrm{div}) \times L^2$ flux-potential mixed finite element discretisation of the Poisson problem

$$\begin{bmatrix} A & B^T \\ B & O \end{bmatrix} \begin{bmatrix} q \\ p \end{bmatrix} = \begin{bmatrix} 0 \\ g \end{bmatrix}, \tag{1}$$

where A is a square matrix arising from the bilinear form $(q_h, \tilde{q})$, B is a non-square matrix arising from the bilinear form $(\mathrm{div}(q_h), \tilde{p})$, O is a square matrix

*`martin.rehor@rafinex.com` Rafinex S.à r.l., Luxembourg.

†`jack.hale@uni.lu` Institute of Computational Engineering, Department of Engineering, Faculty of Science, Technology and Medicine, University of Luxembourg.

of zeros, q is a vector of unknowns related to the flux, p is a vector of unknowns related to the pressure, 0 is a vector of zeros and g is a vector arising from the linear form $-(f, \tilde{p})$.

It is now well established that such block-structured systems can only be solved scalably (ideally, in linear time in the number of unknowns on high-performance computing systems) using Krylov subspace methods in conjunction with suitably designed preconditioning strategies [1, 8].

FEniCSx-pctools consists of a set of algorithms that can analyse the high-level Unified Form Language (UFL) representation of a block-structured finite element problem and subsequently create the necessary PETSc (Portable, Extensible Toolkit for Scientific Computation) data structures describing the same block structure on the algebraic level. With this information, advanced block preconditioning strategies can be specified straightforwardly using PETSc's standard options-based configuration system.

We include three fully documented demos. The first one is a motivational demo that introduces FEniCSx-pctools as a useful complement to DOLFINx; an elementary system of algebraic equations is used to illustrate the ability to change the solver configuration at runtime independently of the model formulation. The other two demos are slightly more involved; one of them sets up a Schur complement preconditioner for the mixed Poisson problem in eq. (1) based on a design proposed in [15], whereas the other one sets up a Schur complement preconditioner of the velocity-pressure Navier-Stokes equations using the pressure-convection-diffusion (PCD) approach proposed in [8].

Related work

We focus on software that provides a high-level interface to block preconditioning strategies exposed by lower-level sparse linear algebra libraries such as Trilinos and PETSc.

1. CBC.Block [13] provides block preconditioning tools within the legacy DOLFIN library using the Trilinos linear algebra backend. A particularly strong aspect of CBC.Block is its domain specific language for specifying block linear algebra preconditioners.

2. The Firedrake Project [14] builds directly on top of the PETSc DM package [12] allowing for the straightforward specification of block-structured algebraic systems and composable physics-based preconditioners. By contrast, DOLFINx is backend agnostic and contains only low-level routines for assembly into PETSc sparse data structures. FEniCSx-pctools bridges this gap, bringing advanced preconditioning specification from PETSc to users of DOLFINx.

3. FENaPack [3] is a preconditioning package for the legacy DOLFIN library using the PETSc linear algebra backend. It is particularly focused on providing an implementation of the PCD approach for preconditioning

the Navier-Stokes equations. An implementation of PCD is also provided in FEniCSx-pctools.

4. PFIBS [6], like FENaPack, is a parallel preconditioning package for the legacy DOLFIN library using the PETSc linear algebra backend. It contains a class BlockProblem that can split a monolithic PETSc matrix into blocks. It also contains an example of PCD preconditioning of the Navier-Stokes equations.

Use in research

In this section we show scaling results demonstrating the relevance of the software to solving real-world problems involving the solution of partial differential equations (PDE) at scale on high-performance computers (HPC).

We perform weak scalability tests up to 8192 MPI processes on a three-field (temperature, pressure and velocity) Rayleigh-Bénard convection problem preconditioned with algebraic multigrid and a pressure-convection-diffusion (PCD) approach [8]. The PDE and problem data (domain, boundary conditions etc.) are exactly the same as the ones described in [10] and solved with Firedrake [14], so for brevity's sake we do not repeat the details. The solution is visualised in fig. 1. We use a slightly different design for the PCD aspect of our preconditioner [2], which results in fewer outer Newton-Raphson iterations and fewer inner Krylov solver iterations than the design proposed in [10].

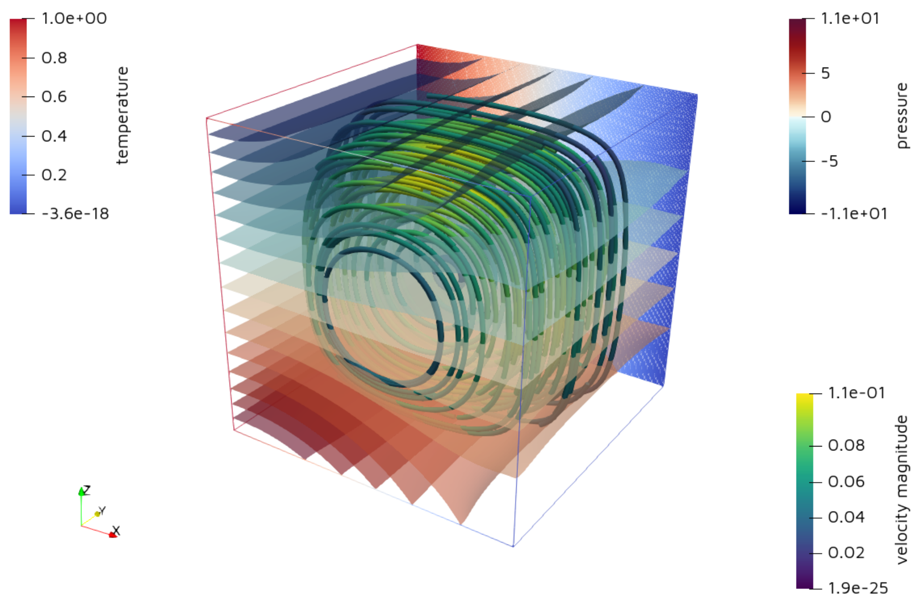

Figure 1: Visualisation of the Rayleigh-Bénard solution.

The computational experiments were performed on the University of Luxembourg Aion cluster [16]. The Aion cluster consists of 354 nodes each with two 64 core AMD Epyc ROME 7H12 processors attached to 256 GB RAM. The nodes are connected with an Infiniband HDR network in a 'fat-tree' topology. We built DOLFINx 0.7.0 against PETSc 3.20.0, OpenMPI 4.0.5 and Python 3.8.6 using the GCC 10.2.0 compiler suite with `-march=znver2 -O3` optimisation flags. All experiments were performed at 50% core utilisation per node, i.e. 64 cores per node as these low-order finite element problems are typically memory-bandwidth constrained. The node allocation was taken in exclusive mode, i.e. with no other jobs running.

In the weak scaling test we solve the problem on 1 node and double the number of nodes until we reach 128 nodes, for a total of 8 192 MPI processes. We simultaneously increase the problem size so that the number of degrees of freedom (DOF) per MPI rank remains fixed at around 100 000.

A breakdown of timings for the weak scaling study are shown in table 1. The code to execute these experiments and our raw timing data is available in the supplementary material. In summary, the time to solution stays roughly constant, demonstrating the excellent parallel performance of DOLFINx and PETSc for this problem when using a scalable preconditioner design implemented via the FEniCSx-pctools package.

DOF ($\times 10^6$)	MPI processes	Nonlinear iterations	Linear iterations	Navier-Stokes iterations	Temperature iterations	Time to solution (s)
6.359	64	2	8	115 (14.4)	49 (6.1)	26.7
12.6	128	2	8	117 (14.6)	49 (6.1)	27.2
25.64	256	2	9	133 (14.8)	56 (6.2)	31.2
101.7	1024	2	7	103 (14.7)	43 (6.1)	25.9
203.5	2048	2	7	102 (14.6)	44 (6.3)	26.1
408.9	4096	2	5	82 (16.4)	31 (6.2)	22.4
816.8	8192	2	6	102 (17)	41 (6.8)	27.1

Table 1: Performance metrics for Rayleigh-Bénard problem with PCD-AMG preconditioning, weak scaling at 100k (DOF) per process. Aion Cluster, 50% utilisation. The number in brackets is the average iterations per outer linear solve.

Implementation and architecture

Existing low-level assembly routines in DOLFINx can be used to build up a block-structured matrix in two consecutive steps:

1. Create PETSc sparse data structures based on an array of bilinear forms; this involves creation of a merged sparsity pattern with the block layout dictated by the arrangement of forms in the array.

2. Assemble the matrix object initialised in the first step in a blockwise manner; this involves creation of index sets that can be used to extract and

assemble each individual submatrix.

The extraction of a submatrix is the key mechanism exploited by PETSc's fieldsplit-based preconditioners [4]. The core of FEniCSx-pctools is a set of functions that attaches the necessary metadata to the PETSc block-structured matrix produced by DOLFINx so that PETSc's options-based preconditioning system can be applied straightforwardly, see fig. 2.

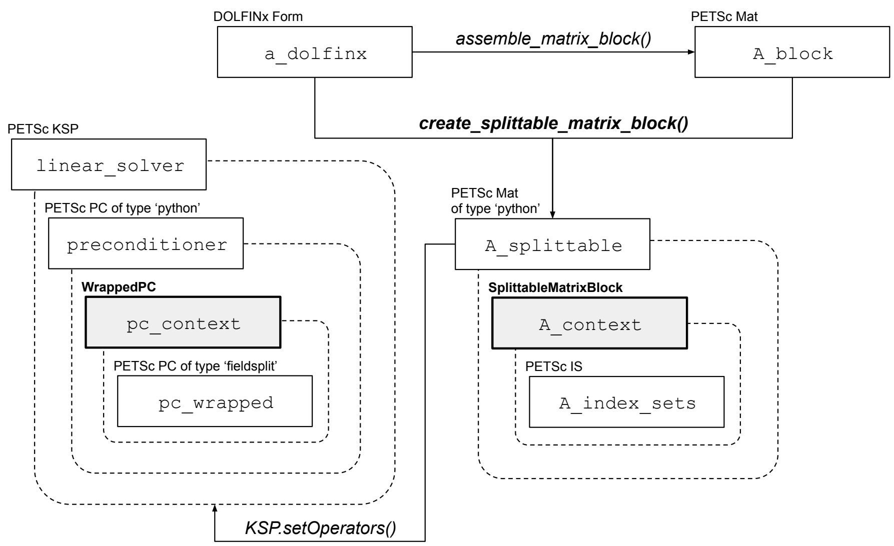

Figure 2: Architecture diagram with the FEniCSx-pctools interface shown in bold.

Let us emphasize that the same configuration of the linear solver can be achieved using the alternative DOLFINx's assembly routines that produce the PETSc block-structured matrix of type 'nest' [4], while it is not necessary to use any high-level wrappers. The main advantage offered by FEniCSx-pctools is the possibility to change the preconditioner setup at runtime without the need to modify the model specification (typically, the arrangement of finite element function spaces that determines the block layout of the system matrix). This is best illustrated/explained in the motivational demo already mentioned above.

Quality control

FEniCSx-pctools contains unit tests that assert that the package functions correctly. In addition, there are three demo problems with checks for correctness. These tests are run as part of a continuous integration pipeline. Users can run these tests themselves by following the instructions in the `README.rst` file. The package is fully documented.

Reuse potential

The design of parameter and discretisation robust block preconditioning strategies is an active research topic in numerical analysis and computational sciences. We can point to recent developments in designs for the Navier–Stokes [9] equations, poroelasticity equations [7], magnetohydrodynamic equations [11] and multiphysics interface problems [5]. Together, FEniCSx-pctools, DOLFINx and PETSc support the straightforward expression and testing of these preconditioning strategies in code, and therefore are useful for researchers in who wish to quickly verify the performance of their preconditioning designs. In addition, block preconditioning strategies are an important tool for solving large real-world problems in computational sciences and engineering.

Users can post issues on our GitLab issue tracker at the above repository.

Availability

Operating system

DOLFINx, and consequently FEniCSx-pctools, can be built on any modern POSIX-like system, e.g. macOS, Linux, FreeBSD etc. Windows is not currently supported.

Programming language

FEniCSx-pctools is written in Python and is compatible with the CPython intepreter version 3.8 and above.

Additional system requirements

FEniCSx-pctools and its main dependencies DOLFINx and PETSc are designed with scalability on parallel distributed memory systems in mind. Consequently, they can run on laptops through to large HPC systems.

Dependencies

FEniCSx-pctools primarily depends on the Python interface to DOLFINx which in turn depends on petsc4py. These dependencies are specified in the standard Python packaging configuration file. We aim to make tagged releases of FEniCSx-pctools that are compatible with DOLFINx releases along with the latest stable release of PETSc. FEniCSx-pctools can be installed straightforwardly alongside any build of the FEniCS Project components.

Supplementary material

Permanent archive:

- *Name:* Figshare [17]
- *Persistent identifier:* `https://doi.org/10.6084/m9.figshare.21408294`
- *Licence:* LGPLv3 or later
- *Publisher:* Martin Řehoř on behalf of Rafinex S.à r.l.
- *Version published:* git tag: v0.7.2
- *Date published:* 27/10/2022 (ongoing)

Development repository:

- *Name:* gitlab.com
- *Persistent identifier:* `https://gitlab.com/rafinex-external-rifle/fenicsx-pctools`
- *Licence:* LGPLv3 or later
- *Date published:* 27/10/2022 (ongoing)

Acknowledgements

The experiments presented in this work were carried out using the HPC facilities of the University of Luxembourg [16] (see `https://hpc.uni.lu`).

Jack S. Hale has a family member that works at Rafinex S.à r.l. This family member was not involved in this research project. Martin Řehoř declares no competing interests.

Funding statement

This research was funded in whole, or in part, by the Luxembourg National Research Fund (FNR), grant reference RIFLE/13754363. For the purpose of open access, and in fulfilment of the obligations arising from the grant agreement, the author has applied a Creative Commons Attribution 4.0 International (CC BY 4.0) license to any Author Accepted Manuscript version arising from this submission.

References

[1] Michele Benzi, Gene H. Golub, and Jörg Liesen. “Numerical solution of saddle point problems”. en. In: *Acta Numerica* 14 (May 2005). Publisher: Cambridge University Press, pp. 1–137. ISSN: 1474-0508, 0962-4929. DOI: `10.1017/S0962492904000212`. (Visited on 10/28/2022).

[2] Jan Blechta. "Towards efficient numerical computation of flows of non-Newtonian fluids". PhD thesis. Univerzita Karlova, June 2019. URL: https://dspace.cuni.cz/handle/20.500.11956/108384 (visited on 10/13/2023).

[3] Jan Blechta and Martin Řehoř. "FENaPack – FEniCS Navier-Stokes preconditioning package". In: *Proceedings of the FEniCS Conference 2017.* (University of Luxembourg, Luxembourg.). Ed. by Jack S. Hale. June 2017, pp. 14–15. DOI: 10.6084/m9.figshare.5086369.

[4] Jed Brown et al. "Composable Linear Solvers for Multiphysics". In: *2012 11th International Symposium on Parallel and Distributed Computing.* 2012 11th International Symposium on Parallel and Distributed Computing (ISPDC). Munich, Germany: IEEE, June 2012, pp. 55–62. ISBN: 978-1-4673-2599-8. DOI: 10.1109/ISPDC.2012.16. URL: http://ieeexplore.ieee.org/document/6341494/ (visited on 03/18/2021).

[5] Ana Budiša et al. "Rational Approximation Preconditioners for Multiphysics Problems". en. In: *Numerical Methods and Applications.* Ed. by Ivan Georgiev et al. Lecture Notes in Computer Science. Cham: Springer Nature Switzerland, 2023, pp. 100–113. ISBN: 978-3-031-32412-3. DOI: 10.1007/978-3-031-32412-3_9.

[6] Justin Chang, Ken Protasov, and Jeffery M. Allen. *PFIBS: a Parallel FEniCS Implementation of Block Solvers.* 2022. URL: https://github.com/NREL/pfibs.

[7] Shuangshuang Chen et al. "Robust block preconditioners for poroelasticity". In: *Computer Methods in Applied Mechanics and Engineering* 369 (Sept. 2020), p. 113229. ISSN: 0045-7825. DOI: 10.1016/j.cma.2020.113229.

[8] Howard Elman, David Silvester, and Andy Wathen. *Finite Elements and Fast Iterative Solvers: with Applications in Incompressible Fluid Dynamics.* Oxford University Press, June 2014. ISBN: 978-0-19-967879-2. DOI: 10.1093/acprof:oso/9780199678792.001.0001.

[9] Patrick E. Farrell et al. "A Reynolds-robust preconditioner for the Scott-Vogelius discretization of the stationary incompressible Navier-Stokes equations". In: *The SMAI Journal of Computational Mathematics* 7 (2021), pp. 75–96. ISSN: 2426-8399. DOI: 10.5802/smai-jcm.72.

[10] Robert C. Kirby and Lawrence Mitchell. "Solver Composition Across the PDE/Linear Algebra Barrier". en. In: *SIAM Journal on Scientific Computing* 40.1 (Jan. 2018), pp. C76–C98. ISSN: 1064-8275, 1095-7197. DOI: 10.1137/17M1133208.

[11] Fabian Laakmann, Patrick E. Farrell, and Lawrence Mitchell. "An Augmented Lagrangian Preconditioner for the Magnetohydrodynamics Equations at High Reynolds and Coupling Numbers". In: *SIAM Journal on Scientific Computing* 44.4 (Aug. 2022). Publisher: Society for Industrial and Applied Mathematics, B1018–B1044. ISSN: 1064-8275. DOI: 10.1137/21M1416539. (Visited on 11/17/2023).

[12] Michael Lange et al. “Efficient Mesh Management in Firedrake Using PETSc DMPlex”. In: *SIAM Journal on Scientific Computing* 38.5 (Jan. 2016). Publisher: Society for Industrial and Applied Mathematics, S143–S155. ISSN: 1064-8275. DOI: 10.1137/15M1026092.

[13] Kent-Andre Mardal and Joachim Berdal Haga. “Block preconditioning of systems of PDEs”. en. In: *Automated Solution of Differential Equations by the Finite Element Method*. Ed. by Anders Logg, Kent-Andre Mardal, and Garth Wells. Lecture Notes in Computational Science and Engineering 84. Springer Berlin Heidelberg, Jan. 2012, pp. 643–655. ISBN: 978-3-642-23098-1 978-3-642-23099-8. DOI: 10.1007/978-3-642-23099-8_35.

[14] Florian Rathgeber et al. “Firedrake: Automating the Finite Element Method by Composing Abstractions”. In: *ACM Transactions on Mathematical Software (TOMS)* 43.3 (Dec. 2016), 24:1–24:27. ISSN: 0098-3500. DOI: 10.1145/2998441.

[15] Torgeir Rusten, Panayot Vassilevski, and Ragnar Winther. “Interior penalty preconditioners for mixed finite element approximations of elliptic problems”. en. In: *Mathematics of Computation* 65.214 (1996), pp. 447–466. ISSN: 0025-5718, 1088-6842. DOI: 10.1090/S0025-5718-96-00720-X.

[16] Sebastien Varrette et al. “Management of an Academic HPC & Research Computing Facility: The ULHPC Experience 2.0”. In: *Proceedings of the 2022 6th High Performance Computing and Cluster Technologies Conference*. HPCCT ’22. New York, NY, USA: Association for Computing Machinery, Oct. 2022, pp. 14–24. DOI: 10.1145/3560442.3560445.

[17] Martin Řehoř and Jack S. Hale. “FEniCSx Preconditioning Tools”. In: (Oct. 2022). DOI: 10.6084/m9.figshare.21408294.