The Journal of Open Source Software

ngsPETSc: A coupling between NETGEN/NGSolve and PETSc

Patrick E. Farrell [1*], **Joachim Schöberl** [2*], **Stefano Zampini** [3*], and **Umberto Zerbinati** [1*¶]

1 University of Oxford, United Kingdom **2** TU Wien, Austria **3** King Abdullah University of Science and Technology, Saudi Arabia ¶ Corresponding author * These authors contributed equally.

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor:

Submitted: 19 July 2024
Published: unpublished

License
Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License (CC BY 4.0).

DRAFT

Summary

Combining advanced meshing techniques with robust solver capabilities is essential for solving difficult problems in computational science and engineering. This paper introduces ngsPETSc, software built with petsc4py (Dalcin et al., 2011) that seamlessly integrates the NETGEN mesher (Schöberl, 1997), the NGSolve finite element library (Schöberl, 2014), and the PETSc toolkit (Balay et al., 2023). ngsPETSc enables the use of NETGEN meshes and geometries in PETSc-based solvers, and provides NGSolve users access to the wide array of linear, nonlinear solvers, and time-steppers available in PETSc.

Statement of Need

Efficiently solving large-scale partial differential equations (PDEs) on complex geometries is vital in scientific computing. PETSc, NETGEN, and NGSolve offer distinct functionalities: PETSc handles linear and nonlinear problems in a discretisation agnostic manner, NETGEN constructs meshes from constructive solid geometry (CSG) described with OpenCASCADE (OpenCASCADE Technology, 2023), and NGSolve offers a wide range of finite element discretisations. Integrating these tools with ngsPETSc promises to streamline simulation workflows and to enhance large-scale computing capabilities for challenging problems. This integration also facilitates seamless mesh exports from NETGEN to PETSc DMPlex, enabling simulations of complex geometries and supporting advanced meshing techniques in other PETSc-based solvers, like Firedrake (Ham et al., 2023).

In particular, by combining PETSc, NETGEN, and NGSolve within ngsPETSc the following new features are available:

- PETSc Krylov solvers, including flexible and pipelined variants, are available in NGSolve. They can be used both with NGSolve matrix-free operators and NGSolve block matrices;
- PETSc preconditioners can be used as components within the NGSolve preconditioning infrastructure;
- PETSc nonlinear solvers are available in NGSolve, including advanced line search and trust region Newton-based methods;
- high order meshes constructed in NETGEN are now available in Firedrake (Ham et al., 2023), enabling adaptive mesh refinement and geometric multigrid on hierarchies of curved meshes.

In conclusion, ngsPETSc is a lightweight, user-friendly interface that bridges the gap between NETGEN, NGSolve, and PETSc, building on top of petsc4py. ngsPETSc aims to assist with the solution of challenging PDEs on complex geometries, enriching the already powerful capabilities of NETGEN, NGSolve, PETSc, and Firedrake.

Examples

In this section we provide a few examples of results that can be obtained using ngsPETSc. We begin by considering a simple primal Poisson problem on a unit square domain discretised with conforming P_2 finite elements and compare the performance of different solvers newly available in NGSolve via ngsPETSc. In particular, we consider PETSc's algebraic multigrid algorithm GAMG ("Parallel Multigrid Smoothing," 2003), PETSc's domain decomposition BDDC algorithm (Zampini, 2016), NGSolve's own implementation of element-wise BDDC, the Hypre algebraic multigrid algorithm (Falgout & Yang, 2002) and the ML algebraic multigrid algorithm (Sala et al., 2004), each combined with the conjugate gradient method. Other than the elementwise BDDC preconditioner, these preconditioners were not previously available in NGSolve. The results are shown in Table 1 and the full example, with more details, can be found in the ngsPETSc documentation. All the preconditioners considered exhibit robust conjugate gradient iteration counts as we refine the mesh for a P_1 discretisation, but out-of-the-box only BDDC type preconditioners are robust as we refine the mesh for a P_2 discretisation. A possible remedy for this issue is discussed in the ngsPETSc documentation.

# DoFs	PETSc GAMG	HYPRE	ML	PETSc BDDC*	Element-wise BDDC**
116716	35	36	31	9	10
464858	69	74	63	8	9
1855428	142	148	127	9	10

Table 1: The number of degrees of freedom (DoFs) and the number of iterations required to solve the Poisson problem with different solvers. Each row corresponds to a level of uniform refinement. The conjugate gradient solve was terminated when the residual norm decreased by six orders of magnitude. *We choose to use PETSc BDDC with six subdomains. **Element-wise BDDC is a custom implementation of BDDC in NGSolve.

We next consider the Oseen problem, i.e.

$$\nu\Delta\vec{u} + \vec{b}\cdot\nabla\vec{u} - \nabla p = \vec{f}, \quad \nabla\cdot\vec{u} = 0,$$

We discretise this problem using high-order Hood-Taylor elements (P_4-P_3) on a unit square domain (Boffi, 1994; Taylor & Hood, 1973). We employ an augmented Lagrangian formulation to better enforce the incompressibility constraint. We present the performance of GMRES (Saad & Schultz, 1986) preconditioned with a two level additive Schwarz preconditioner with vertex-patch smoothing as fine level correction (Benzi & Olshanskii, 2006; Farrell et al., 2021). This preconditioner was built using ngsPETSc. The result for different viscosities ν are shown in Table 2, exhibiting reasonable robustness as the viscosity (and hence Reynolds number) changes. The full example, with more details, can be found in the ngsPETSc documentation.

# refinements (# DoFs)	$\nu = 10^{-2}$	$\nu = 10^{-3}$	$\nu = 10^{-4}$
1 (83842)	3	4	6
2 (334082)	3	4	6
3 (1333762)	3	4	6

Table 2: The number of iterations required to solve the Oseen problem with different viscosities and different refinement levels. In parentheses we report the number of degrees of freedom (DoFs) on the finest level. The GMRES iteration was terminated when the residual norm decreased by eight orders of magnitude.

Figure 1 shows a simulation of a hyperelastic beam, solved with PETSc nonlinear solvers; the line search algorithms in PETSc solve this straightforwardly, but an undamped Newton

DRAFT

iteration does not converge. Figures 2 and 3 show simulations in Firedrake that were not previously possible. Figure 2 shows a high-order NETGEN mesh employed for the simulation of a Navier-Stokes flow past a cylinder, while Figure 3 shows adaptive mesh refinement for a Poisson problem on an L-shaped domain. The adaptive procedure achieves the optimal complexity of error with degree of freedom count, as expected (Stevenson, 2006).

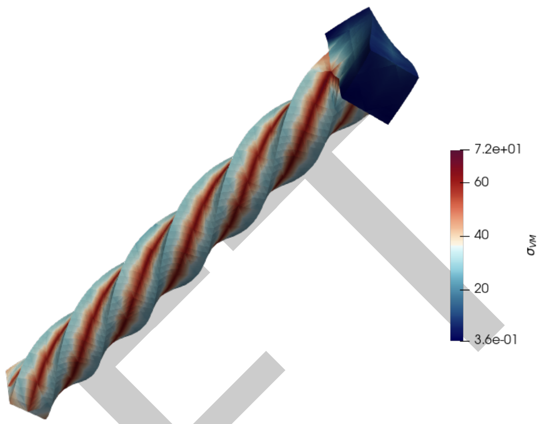

Figure 1: A hyperelastic beam deformed by fixing one end and applying a twist at the other end. The colouring corresponds to the deviatoric von Mises stress experienced by the beam. The beam is discretised with P_3 finite elements and the nonlinear problem is solved using PETSc SNES. The full example, with more details, can be found in the ngsPETSc documentation.

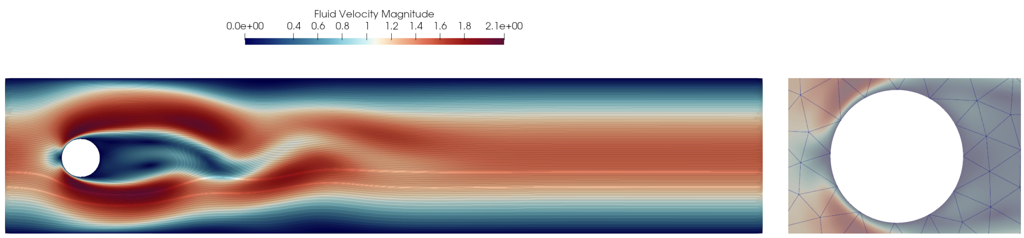

Figure 2: Flow past a cylinder. The Navier-Stokes equations are discretised on a NETGEN high-order mesh with Firedrake. We use high-order Taylor-Hood elements (P_4-P_3) and a vertex-patch smoother as fine level correction in a two-level additive Schwarz preconditioner, (Benzi & Olshanskii, 2006; Farrell et al., 2021). The full example, with more details, can be found in ngsPETSc documentation. On the right a zoom near the cylinder shows the curvature of the mesh.

DRAFT

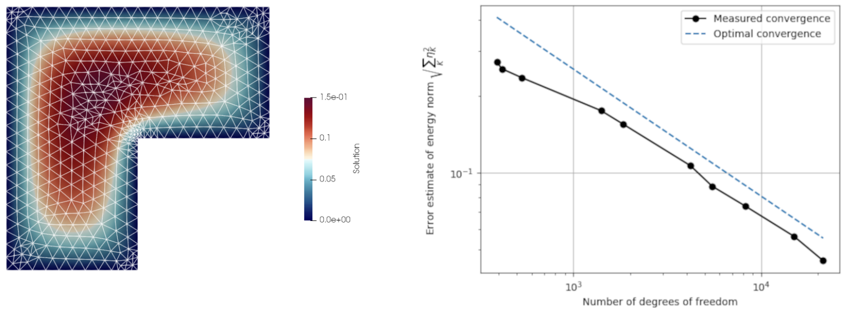

Figure 3: An adaptive scheme applied to the Poisson problem on an L-shaped domain. The domain is discretised using P_1 finite elements and the adaptive mesh refinement is driven by a Babuška-Rheinboldt error estimator (Babuška & Rheinboldt, 1978). The adaptive procedure delivers optimal scaling of the energy norm of the error in terms of the number of degrees of freedom. The full example, with more details, can be found in the ngsPETSc documentation.

More examples can be found in the documentation of ngsPETSc manual (Zerbinati, 2024).

Babuška, I., & Rheinboldt, W. C. (1978). A-posteriori error estimates for the finite element method. *International Journal for Numerical Methods in Engineering*, *12*(10), 1597–1615. https://doi.org/10.1002/nme.1620121010

Balay, S., Abhyankar, S., Adams, M. F., Benson, S., Brown, J., Brune, P., Buschelman, K., Constantinescu, E., Dalcin, L., Dener, A., Eijkhout, V., Faibussowitsch, J., Gropp, W. D., Hapla, V., Isaac, T., Jolivet, P., Karpeev, D., Kaushik, D., Knepley, M. G., … Zhang, J. (2023). *PETSc/TAO users manual* (ANL-21/39 - Revision 3.20). Argonne National Laboratory. https://doi.org/10.2172/2205494

Benzi, M., & Olshanskii, M. A. (2006). An augmented lagrangian–based approach to the oseen problem. *SIAM Journal on Scientific Computing*, *28*(6), 2095–2113. https://doi.org/10.1137/050646421

Boffi, D. (1994). Stability of higher order triangular Hood–Taylor methods for the stationary Stokes equations. *Mathematical Models and Methods in Applied Sciences*, *04*(02), 223–235. https://doi.org/10.1142/S0218202594000133

Dalcin, L. D., Paz, R. R., Kler, P. A., & Cosimo, A. (2011). Parallel distributed computing using python. *Advances in Water Resources*, *34*(9), 1124–1139. https://doi.org/10.1016/j.advwatres.2011.04.013

Falgout, R. D., & Yang, U. M. (2002). Hypre: A library of high performance preconditioners. In P. M. A. Sloot, A. G. Hoekstra, C. J. K. Tan, & J. J. Dongarra (Eds.), *Computational science — ICCS 2002* (pp. 632–641). Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-47789-6_66

Farrell, P. E., Mitchell, L., Scott, L. R., & Wechsung, F. (2021). A Reynolds-robust preconditioner for the Scott-Vogelius discretization of the stationary incompressible Navier-Stokes equations. *The SMAI Journal of Computational Mathematics*, *7*, 75–96. https://doi.org/10.5802/smai-jcm.72

Ham, D. A., Kelly, P. H., Mitchell, L., Cotter, C. J., Kirby, R. C., Sagiyama, K., Bouziani, N., Vorderwuelbecke, S., Gregory, T. J., Betteridge, J., & others. (2023). Firedrake user manual. *Imperial College London and University of Oxford and Baylor University and University of Washington,*. https://doi.org/10.25561/104839

OpenCASCADE Technology. (2023). *OpenCASCADE*. https://www.opencascade.com/

Parallel multigrid smoothing: Polynomial versus Gauss–Seidel. (2003). *Journal of Computational Physics*, *188*(2), 593–610. https://doi.org/10.1016/S0021-9991(03)00194-3

Saad, Y., & Schultz, M. H. (1986). GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, *7*(3), 856–869. https://doi.org/10.1137/0907058

Sala, M., Hu, J. J., & Tuminaro, R. S. (2004). *ML3.1 Smoothed Aggregation User's Guide* (No. SAND2004-4821). Sandia National Laboratories.

Schöberl, J. (1997). NETGEN an advancing front 2D/3D-mesh generator based on abstract rules. *Computing and Visualization in Science*, *1*(1), 41–52. https://doi.org/10.1007/s007910050004

Schöberl, J. (2014). C++ 11 implementation of finite elements in NGSolve. *Institute for Analysis and Scientific Computing, Vienna University of Technology*, *30*.

Stevenson, R. (2006). Optimality of a standard adaptive finite element method. *Foundations of Computational Mathematics*, *7*(2), 245–269. https://doi.org/10.1007/s10208-005-0183-0

Taylor, C., & Hood, P. (1973). A numerical solution of the Navier–Stokes equations using the finite element technique. *Computers & Fluids*, *1*(1), 73–100. https://doi.org/10.1016/0045-7930(73)90027-3

Zampini, S. (2016). PCBDDC: A class of robust dual-primal methods in PETSc. *SIAM Journal on Scientific Computing*, *38*(5), S282–S306. https://doi.org/10.1137/15M1025785

Zerbinati, U. (2024). *ngsPETSc user manual* (Version 0.0.5). Zenodo. https://doi.org/10.5281/zenodo.12650574

DRAFT