



HAL
open science

KSPHPDDM and PCHPDDM: Extending PETSc with advanced Krylov methods and robust multilevel overlapping Schwarz preconditioners

Pierre Jolivet, Jose E. Roman, Stefano Zampini

► **To cite this version:**

Pierre Jolivet, Jose E. Roman, Stefano Zampini. KSPHPDDM and PCHPDDM: Extending PETSc with advanced Krylov methods and robust multilevel overlapping Schwarz preconditioners. *Computers & Mathematics with Applications*, 2021, 84, pp.277-295. 10.1016/j.camwa.2021.01.003 . hal-04751926

HAL Id: hal-04751926

<https://hal.science/hal-04751926v1>

Submitted on 13 Nov 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons Attribution - NonCommercial 4.0 International License

KSPHPDDM and PCHPDDM: extending PETSc with advanced Krylov methods and robust multilevel overlapping Schwarz preconditioners

Pierre Jolivet^{a,*}, Jose E. Roman^b, Stefano Zampini^c

^a*CNRS, IRIT-ENSEEIH, Toulouse, France*

^b*Universitat Politècnica de València, València, Spain*

^c*King Abdullah University of Science and Technology, Thuwal, Saudi Arabia*

Abstract

Contemporary applications in computational science and engineering often require the solution of linear systems which may be of different sizes, shapes, and structures. The goal of this paper is to explain how two libraries, PETSc and HPDDM, have been interfaced in order to offer end-users robust overlapping Schwarz preconditioners and advanced Krylov methods featuring recycling and the ability to deal with multiple right-hand sides. The flexibility of the implementation is showcased and explained with minimalist, easy-to-run, and reproducible examples, to ease the integration of these algorithms into more advanced frameworks. The examples provided cover applications from eigenanalysis, elasticity, combustion, and electromagnetism.

Keywords: Krylov methods, domain decomposition preconditioners, distributed-memory parallel computing

1. Introduction

Computational science and engineering today enjoys unprecedented opportunities to transform the way society solves many of its most urgent technological problems through predictive simulations. At the heart of simulations are robust and scalable solution algorithms. The efficient production of applications requires a rich ecosystem of state-of-the-art reusable libraries from which domain specialists can benefit [1]. In this paper, we focus on the interoperability of two such libraries.

On the one hand, PETSc [2, 3], the Portable and Extensible Toolkit for Scientific computation, is a well-established, actively developed software from the community. It may be used to efficiently discretize partial differential equations and solve algebraic linear or nonlinear systems of time-dependent equations. Among its strengths, it offers many advanced features regarding preconditioning and tailored matrix formats, and it can interact with third-party libraries, such as *hypra* [4] for linear solvers, TetGen [5] for mesh generation, p4est [6] for adaptive mesh refinement, to cite a few. The extensibility and robustness of the framework convinced computational scientists to use PETSc as one of the discretization and/or algebraic backend in many different higher-level projects, see <https://www.mcs.anl.gov/petsc> for a comprehensive list.

*Corresponding author

Email addresses: pierre.jolivet@enseeiht.fr (Pierre Jolivet), jroman@dsic.upv.es (Jose E. Roman), stefano.zampini@kaust.edu.sa (Stefano Zampini)

17 On the other hand, HPDDM [7], the High-Performance unified framework for Domain Decom-
18 position Methods (HPDDM) is a much smaller project focusing on robust and scalable domain
19 decomposition preconditioners and advanced iterative methods [8] which can efficiently deal with
20 linear systems with multiple right-hand sides, i.e., block iterative methods, and when there is a
21 recurrence of varying coefficient matrices and right-hand sides, i.e., recycling iterative methods.

22 Here, we describe the integration of the HPDDM solvers suite consisting of advanced iterative
23 methods and robust domain decomposition preconditioners into PETSc. The design principles
24 of the interface are showcased considering minimalist and easy-to-run examples, with a focus on
25 fulfilling reproducibility requirements as advocated by the Association for Computing Machinery
26 <https://www.acm.org/publications/policies/artifact-review-badging>, as well as to ease
27 the integration of these algorithms into more advanced frameworks and facilitate the analysis and
28 comparison of performance results [9].

29 Fruits of the proposed enhancement are discussed for applications from SLEPc [10], the Scalable
30 Library for Eigenvalue Problem computations, an add-on library that extends PETSc with classes
31 for linear and nonlinear eigenvalue problems, as well as with tools to facilitate the use of shift-and-
32 invert spectral transformations.

33 The paper is divided in two main parts. Section 2 introduces the interface to recycling and
34 block Krylov methods (KSPHPDDM) and discusses the readiness of the PETSc library with respect to
35 block Krylov methods. Different applications from eigenanalysis are also provided. Section 3 intro-
36 duces the suite of robust multilevel overlapping Schwarz methods (PCHPDDM) with examples from
37 linear and nonlinear partial differential equations. Eventually, concluding remarks are provided in
38 section 4.

39 All PETSc keywords and options are typeset in typewriter font, these are public and docu-
40 mented at <https://www.mcs.anl.gov/petsc/petsc-master/docs>. Codes not already available
41 in other public repositories are available at <https://github.com/prj-/jolivet2020petsc>. In
42 **Appendix A, a short guide to build the required set of libraries for running these codes is pro-**
43 **vided.**

44 **2. Advanced Krylov methods in PETSc: KSPHPDDM**

45 *2.1. Related work*

46 Krylov subspace methods are widely used in numerical linear algebra for solving linear sys-
47 tems of equations because of their memory efficiency [11, 12]. They mainly rely on matrix–vector
48 operations such as multiplications or transpose multiplications and, for robust convergence when
49 dealing with challenging high-dimensional systems, on the application of appropriate precondition-
50 ers. PETSc, as of version 3.14.0, already offers 44 different types of Krylov methods within the
51 KSP base class. Some of the most used types are KSPGMRES, KSPCG, or KSPBCGS, which respectively
52 implement the generalized minimal residual method [13] (GMRES), conjugate gradient [14] (CG),
53 or biconjugate gradient stabilized method [15] (BCGS).

54 Because Krylov methods may need long-term recurrences to mitigate round-off errors during
55 the generation of subspaces, it is common to introduce a restart parameter in order to control
56 their memory consumption and the volume of global communications needed when orthonormal-
57 izing a candidate basis vector. Such restarts may hinder convergence of iterative methods, and
58 even introduce convergence plateaus. Recycling techniques have been introduced to attenuate
59 these effects. On the one hand, PETSc implements the loose GMRES [16] (LGMRES) and the

60 deflated GMRES [17, 18] (DGMRES) as `KSP_LGMRES` and `KSP_DGMRES`. However, neither handle
61 variable preconditioning, unlike `KSP_FGMRES` or `KSP_GCR` which respectively implement the flexible
62 GMRES [19] (FGMRES) and the generalized conjugate residual method [20] (GCR). Furthermore,
63 `KSP_DGMRES` does not support complex arithmetic. Moreover, extending the recycling capabilities of
64 such methods to sequence of linear systems with smoothly varying coefficient matrices and right-
65 hand sides is not trivial. On the other hand, HPDDM offers support for the generalized conjugate
66 residual method with inner orthogonalization and deflated restarting [21] (GCRODR), which does
67 not suffer from the aforementioned limitations.

68 Another important aspect of Krylov methods is their ability to deal with multiple right-hand
69 sides simultaneously. Block Krylov methods [22] are designed for solving linear systems $AX = B$,
70 where X and B are tall-and-skinny matrices with $k \geq 1$ columns. Such methods, while having
71 higher arithmetic intensities, generate larger subspaces and typically converge in fewer iterates.
72 These methods are not currently offered in PETSc and they are less frequently implemented in
73 general purpose libraries because they require somehow more involved kernels such as matrix-
74 matrix multiplication, instead of matrix-vector product, and may involve different inner-product
75 realizations [23]. Still, systems with multiple right-hand sides are ubiquitous, e.g., in tomogra-
76 phy [24], data analytics [25], eigensolvers [26], geophysics [27], quantum chromodynamics [28], and
77 optimization with time-dependent partial differential equations as constraints.

78 Trilinos [29] is another well-known software for scientific computing and it provides iterative
79 methods through its Belos package [30]. Although having PETSc and Trilinos interoperate is
80 possible, there is currently no KSP interface to Belos. Furthermore, some of the Krylov methods
81 implemented in HPDDM and discussed in section 2.2, are not available in Belos.

82 2.2. Interfacing HPDDM Krylov methods in PETSc

83 In this section, we provide details of the interface between HPDDM Krylov methods and the
84 various PETSc classes. It is assumed that the right-hand sides, and consequently the solutions,
85 are always dense vectors or matrices.

86 `KSP_HPDDM`, the interface between HPDDM Krylov methods and PETSc, is automatically regis-
87 tered since PETSc version 3.12 when configuring PETSc with the extra flag `--download_hpddm`.
88 HPDDM linear solvers can be selected using the command line option `-ksp_type hpddm`, or pro-
89 grammatically via `KSPSetType(ksp, KSP_HPDDM)`. Then, the following Krylov methods can be ac-
90 cessed:

- 91 • pseudo-block GMRES or flexible GMRES [19];
- 92 • pseudo-block CG or flexible CG [31];
- 93 • pseudo-block GCRODR or flexible GCRODR [32];
- 94 • block GMRES or flexible GMRES [27], with deflation at each restart;
- 95 • block CG [33];
- 96 • breakdown-free block CG [34], with deflation at each iteration;
- 97 • block GCRODR or flexible GCRODR, with deflation at each restart.

98 While being mathematically equivalent to their “standard” counterparts, the pseudo-block variants
99 fuse together multiple similar operations like matrix-vector products to achieve higher arithmetic
100 intensity, or to decrease the number of global synchronizations needed for scalar products.

101 HPDDM Krylov methods may be selected using the options `-ksp_hpddm_type (gmres|cg|gcrodr|`
102 `bgmres|bcg|bfbcg|bgcrodr|preonly)`, or programmatically by using `KSP_HPDDMSetType(ksp,`
103 `KSP_HPDDMType)`. Preconditioning variants can be specified via `-ksp_hpddm_variant (left|right|`

flexible) to select whether the preconditioner is applied on the left, on the right, or if it cannot be represented as a linear operator. In this latter case, the preconditioner is always applied on the right, except for the conjugate gradient methods BCG, BFCG, and BFBCG, that only handle left preconditioning. It is also possible to set the preconditioning side through the more common PETSc option `-ksp_pc_side (left|right)`. In addition, convergence monitoring with the `KSPMonitor` interface is fully supported, as well as the specification of customized convergence testing via the `KSPSetConvergenceTest` callback.

Recycling Krylov methods try to extract convergence information by solving a small standard or a generalized dense eigenproblem at the end of each cycle or when convergence is reached, and then reuse it appropriately for subsequent solves [35, 36, 37]. Such deflation subspaces, stored as dense tall-and-skinny matrices, can be accessed with `KSPHPDDMGetDeflationSpace`. User-defined deflation subspaces can also be specified via `KSPHPDDMSetDeflationSpace`.

To fully support HPDDM solvers, in version 3.14.0 of PETSc, we added interface routines for solving systems with multiple right-hand sides, `KSPMatSolve(ksp, X, Y)`, and to apply preconditioners, `PCMatApply(pc, X, Y)`. Both input X and output Y matrices are currently limited to be dense tall-and-skinny matrices. With `KSPHPDDM`, (pseudo-)block Krylov methods will be used. Instead, when no specialized implementation is available, PETSc will perform the solution phase in a column by column fashion. Furthermore, a function `KSPSetMatSolveBlockSize` is also provided to decompose a single large block of column vectors into multiple sub-blocks. `KSPMatSolve` is then called repeatedly until all sub-blocks are traversed. Similar considerations apply to the underlying calls to `PCMatApply` and `MatProductNumeric`. This was inspired by MUMPS [38] option `ICNTL(27)`¹.

While solving linear systems, possibly with multiple right-hand sides, HPDDM will repeatedly call the following PETSc routines:

- `MatMult(A, x, y)` for $y = Ax$;
- `MatMatMult(A, X, MAT_REUSE_MATRIX, PETSC_DEFAULT, Y)`² for $Y = AX$;
- `PCApply(M, x, y)` to apply a preconditioner to x , i.e., for $y = M^{-1}x$;
- `PCMatApply(M, X, Y)` for $Y = M^{-1}X$;

The rest of the operations are performed directly inside HPDDM. Specifically, within Krylov methods using the Arnoldi process, or when recycling is requested, one has to compute QR factorizations of tall-and-skinny dense matrices to orthonormalize candidate basis vectors. Such operations are performed using the CholQR algorithm [39], or via the (modified) Gram–Schmidt method. These orthonormalization variants can be selected at runtime with the option `-ksp_hpddm_qr (cholqr|mgs|cgs)`. For Hessenberg matrices generated by the Arnoldi process, it is common to update their QR decomposition using Givens rotations [12]. For block Hessenberg matrices, Householder reflectors are used instead [40].

For matrix–matrix products, there are currently specialized implementations for the following `MatTypes`:

- `MATAIJ`: standard sequential or parallel sparse matrix, based on compressed sparse row format;
- `MATSEQBAIJ`: block sparse matrix, based on block compressed sparse row format;
- `MATSEQSBAIJ`: symmetric block sparse matrix stored in upper triangular form;
- `MATSHELL`: user-defined matrix;
- `MATNEST`: block-defined matrix with nested submatrices;

¹http://mumps.enseeiht.fr/doc/userguide_5.3.3.pdf, section 6.1

²or `MatProductNumeric(Y)` with PETSc 3.14.0 and above

147 • MATAIJCUSPARSE: sequential or parallel sparse matrix, offloaded to a NVIDIA GPU using
148 cuSPARSE [41].

149 The following preconditioners have specialized implementations for dealing efficiently with multiple
150 vectors:

- 151 1. PCKSP: embedded Krylov method;
- 152 2. PCMAT: matrix multiplication;
- 153 3. PCH2OPUS: hierarchical matrices [42, 43];
- 154 4. PCHPDDM: see section 3;
- 155 5. PCASM and PCGASM: overlapping Schwarz methods;
- 156 6. PCBJACOBI: block Jacobi;
- 157 7. PCLU or PCCHOLESKY: exact LU or Cholesky factorization;
- 158 8. PCILU or PCICC: incomplete LU or Cholesky factorization.

159 One appealing feature of domain decomposition-like preconditioners (items 4 to 6) is that they
160 most often rely on exact or inexact factorizations (items 7 and 8) as subdomain solvers. In these
161 cases, it is possible to access the so-called factored matrix F via `PCFactorGetMatrix`, and then
162 call `MatMatSolve(F , X , Y)` to take advantage of blocked forward eliminations and backward
163 substitutions from the various factorization packages interfaced with PETSc. In the case of a
164 sparse matrix, this strategy is possible with: MUMPS [38], SuiteSparse [44], MKL PARDISO
165 or CPARDISO [45], and SuperLU [46] or SuperLU_DIST [47]. For the case of a dense matrix:
166 ScaLAPACK and Elemental [48] are supported. Future work will consider extending the multigrid
167 framework PCMG, as well as other preconditioning classes to increase arithmetic intensity of the
168 preconditioner application phase for blocks of right-hand sides.

169 2.3. Applications and numerical results

170 2.3.1. Reproducibility of the results from Parks et al. [21]

171 Alongside the paper introducing GCRODR [21], a MATLAB implementation was provided
172 and is since then available³. It comes with a sequence of ten “linear systems from a finite element
173 fracture mechanics problem constructed by Philippe H. Geubelle and Spandan Maiti.” The goal
174 of this paragraph is to explain how the results can be reproduced with PETSc and KSPHPDDM:
175 the matrix files used are available at [https://gitlab.com/petsc/datafiles/-/tree/master/
176 matrices/hpddm/GCRODR](https://gitlab.com/petsc/datafiles/-/tree/master/matrices/hpddm/GCRODR) while the driver code is part of the PETSc test suite and available
177 at <https://www.mcs.anl.gov/petsc/petsc-master/src/ksp/ksp/tutorials/ex75.c.html>. In
178 order to check the correctness of the KSPHPDDM interface, the following tests are performed:

- 179 • in MATLAB, unpreconditioned GCRODR(40, 20), ICC(0) left-preconditioned GCRODR(40,
180 20), and Jacobi right-preconditioned GCRODR(40, 20);
- 181 • in PETSc with no preconditioning and no restart GMRES(∞) KSPGMRES;
- 182 • in PETSc with KSPHPDDM, all of the above.

183 The notation GCRODR(n , m) indicates a restart after n iterations and a recycling subspace of
184 dimension m . In all of the cases, convergence is declared when the initial unpreconditioned residual
185 is reduced by 10 orders of magnitude. Except for the right-preconditioned GCRODR, these tests
186 are the same as the ones from the original GCRODR paper [21]. All PETSc tests are performed
187 using four MPI processes and can be launched using the following command lines.
188

³<https://www.sandia.gov/~mlparks/GCRODR.zip>

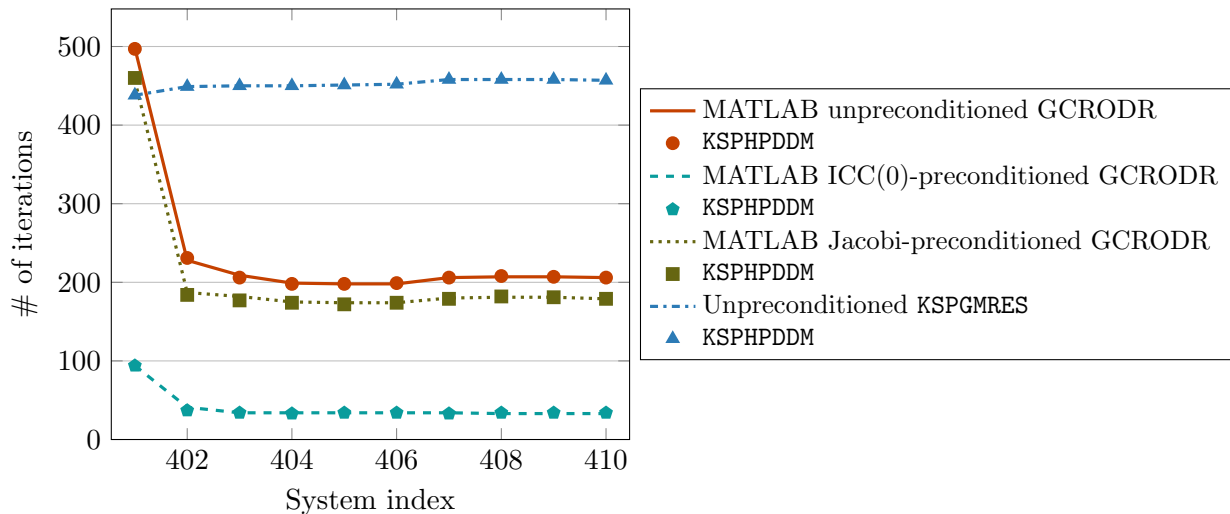


Figure 1: Number of iterations needed to converge for various configurations as originally tested by Parks et al. [21]. Note that the numbers of iterations needed by KSPHPDDM match with the respective MATLAB or PETSc reference implementation.

```

$ mpirun -n 4 ./ex75 -ksp_converged_reason -pc_type none -ksp_rtol 1e-10 -ksp_gmres_restart 40 \
-ksp_type hpddm -ksp_hpddm_type gcrodr -ksp_hpddm_recycle 20 \
-load_dir ${DATAFILES_PATH}/matrices/hpddm/GCRODR
$ mpirun -n 4 ./ex75 -ksp_converged_reason -redundant_pc_type icc -ksp_rtol 1e-10 \
-ksp_type hpddm -ksp_hpddm_type gcrodr -ksp_gmres_restart 40 -ksp_hpddm_recycle 20 \
-load_dir ${DATAFILES_PATH}/matrices/hpddm/GCRODR -pc_type redundant
$ mpirun -n 4 ./ex75 -ksp_converged_reason -pc_type jacobi -ksp_rtol 1e-10 -ksp_gmres_restart 40 \
-ksp_type hpddm -ksp_hpddm_type gcrodr -ksp_hpddm_recycle 20 \
-ksp_pc_side right -load_dir ${DATAFILES_PATH}/matrices/hpddm/GCRODR
$ mpirun -n 4 ./ex75 -ksp_converged_reason -pc_type none -ksp_rtol 1e-10 -ksp_gmres_restart 500 \
-ksp_type hpddm -load_dir ${DATAFILES_PATH}/matrices/hpddm/GCRODR

```

189 The iteration numbers needed to reach convergence are gathered in figure 1. The cases — and ●
190 (resp. — and ▲) reproduce Figure 4.2 in [21], while cases — and ● partially reproduce Figure
191 4.3.

192 2.3.2. Performance of primitives for block Krylov methods

193 The readiness of PETSc, as of version 3.14.0, is now discussed when it comes to delivering
194 efficient implementations of the core matrix–matrix multiplication needed by block Krylov methods
195 and when a restart occurs in recycling Krylov methods.

196 In particular, we consider the first three sequential matrix types from section 2.2, namely
197 MATSEQAIJ, MATSEQBAIJ, and MATSEQSBAIJ. Their distributed memory counterparts, e.g., MATMPIAIJ,
198 rely on the sequential implementations, with each process storing two such matrices, one with lo-
199 cal rows and columns, and another with local rows and “nonlocal” columns. To keep this study
200 succinct, we only evaluate the performance of the sequential implementations, which can help in
201 drawing conclusions for the intraprocess performance of the parallel formats. For this reason,
202 from now on, we drop the SEQ substring. The performance of the multiplication primitives of the
203 three aforementioned types will also be compared against those obtained with the MKL inspector–

204 executor sparse BLAS routines [45], using a single OpenMP thread. There is an ongoing effort to
 205 better integrate these routines in PETSc, but at the time of writing, direct calls to the MKL were
 206 used.















207 For benchmarking, we first discretize the Poisson equation on a cube with trilinear finite ele-
 208 ments and obtain a matrix A of dimension one million. Though the numerical values of A are of no
 209 interest here, the sparsity pattern is rather common and frequently encountered when discretizing
 210 partial differential equations. We then generate a random symmetric dense b -by- b matrix T , for
 211 $b \in \{1, 3, 6\}$, and the matrix $\mathbb{A} = A \otimes T$ is assembled into all three formats described above, using a
 212 block size of b for the block formats. The performance of the `MatProductNumeric` operation is eval-
 213 uated for tall-and-skinny dense matrices with a varying number of columns $N \in \{2, 8, 16, 32, 64\}$.
 214 For the single column case $N = 1$, the `MatMult` operation is used. All results have been obtained us-
 215 ing double-precision arithmetic and 32-bit integers. The scaled efficiency measured in GFLOP per
 216 second is reported with respect to the baseline MATAIJ implementation with $N = 1$ in figure 2, where
 217 performances have been averaged over five consecutive product operations. They can be repro-
 218 duced by using the mini-app `MatProduct.c` from <https://github.com/prj-/jolivet2020petsc>
 219 and any input MATAIJ stored in binary format, here using the default name `binaryoutput`, available
 220 at <http://jolivet.perso.enseeiht.fr/binaryoutput>.

```

$ mpicc MatProduct.c -O3 -I${PETSC_DIR}/${PETSC_ARCH}/include -I${PETSC_DIR}/include \
-L${PETSC_DIR}/${PETSC_ARCH}/lib -lpetsc -o MatProduct
$ mpirun -n 1 ./MatProduct -f binaryoutput -log_view \
-bs 1,3,6 -N 1,2,8,16,32,64 -type aij,aijmk1,baij,baijmk1,sbaij,sbaijmk1

```

221 Some conclusions may be drawn:

- 222 • with a block size of 1, top plot $b = 1$ in figure 2, using any type but MATAIJ  is counterpro-
 223 ductive when $N > 1$, see  and , even when using the MKL, see  and . With 8 or
 224 more columns, the performance of MATAIJ plateaus and the efficiency reaches approximately
 225 200%, while the performance of the MKL primitives stagnate at around 175%, see . For
 226 the single column case, MATSBAIJ and MATAIJMKL deliver slightly better performances;
- 227 • for block sizes lower or equal than 5, loops for matrix–vector and matrix–matrix multiplica-
 228 tions are unrolled by hand for block formats. This yields rather disappointing performance,
 229 except for MATAIJ, see  in middle plot $b = 3$, and its 350% efficiency, even with a moder-
 230 ate number of columns. For block formats  and , given the small value of b , it could
 231 be beneficial to switch to optimized libraries for small matrices, e.g., LIBXSMM [49]. In
 232 fact, MKL implementation for block formats is here clearly outperforming PETSc, see 
 233 and ;
- 234 • for block sizes larger than 5, block entry multiplication with PETSc block formats is per-
 235 formed using `?gemv` or `?gemm` operations. For MATAIJ  in the bottom plot corresponding
 236 to $b = 6$, the efficiency quickly caps near 400% for $N \in \{8, 16, 32, 64\}$, while reaching 500%
 237 for large number of columns for the block formats, see  and  for $N = 64$. With these
 238 numbers of columns, PETSc and MKL perform similarly.

239 The mini-app is then used to benchmark intranode performance of matrix–matrix multiplications,
 240 either with multiple OpenMP threads, or by offloading the operation to a NVIDIA GPU and using
 241 cuSPARSE [41] as interfaced in PETSc.

```

$ export OMP_NUM_THREADS=20 && export MKL_NUM_THREADS=20
$ mpirun -n 1 ./MatProduct -f binaryoutput -log_view \
-bs 1,3,6 -N 1,2,8,16,32,64 -type aijmk1,baijmk1,sbaijmk1,aijcusparse

```

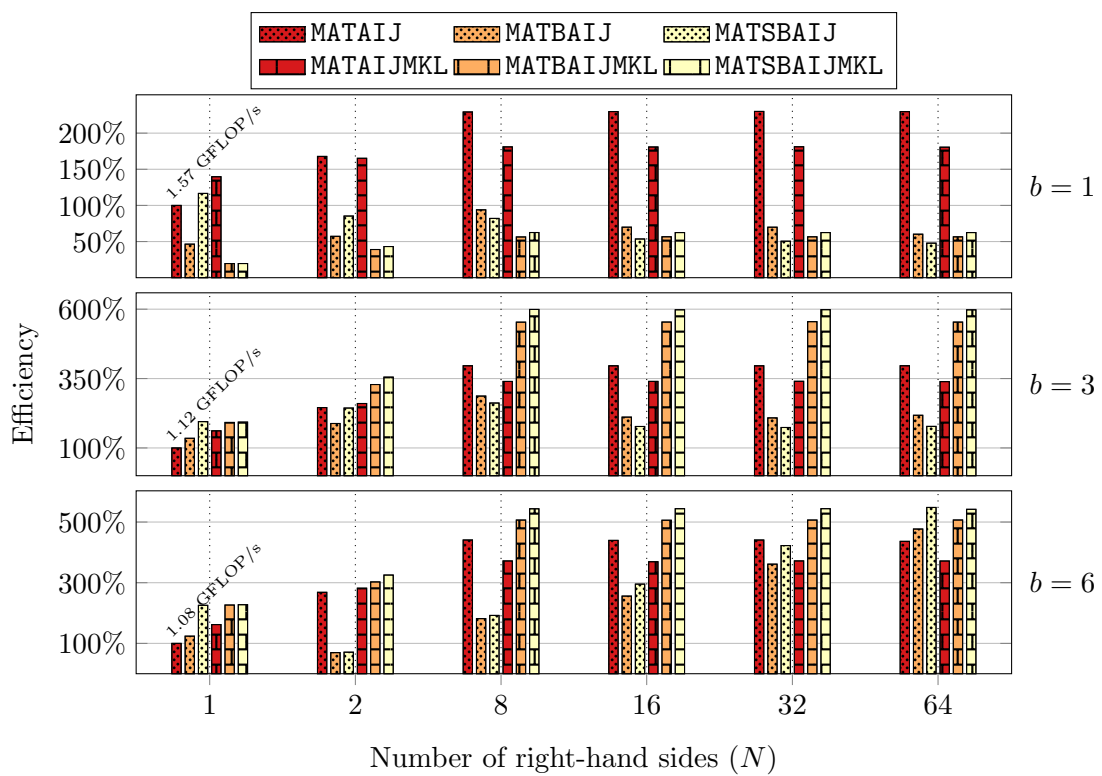



Figure 2: Performance of the matrix–matrix multiplication for different sparse matrix formats.

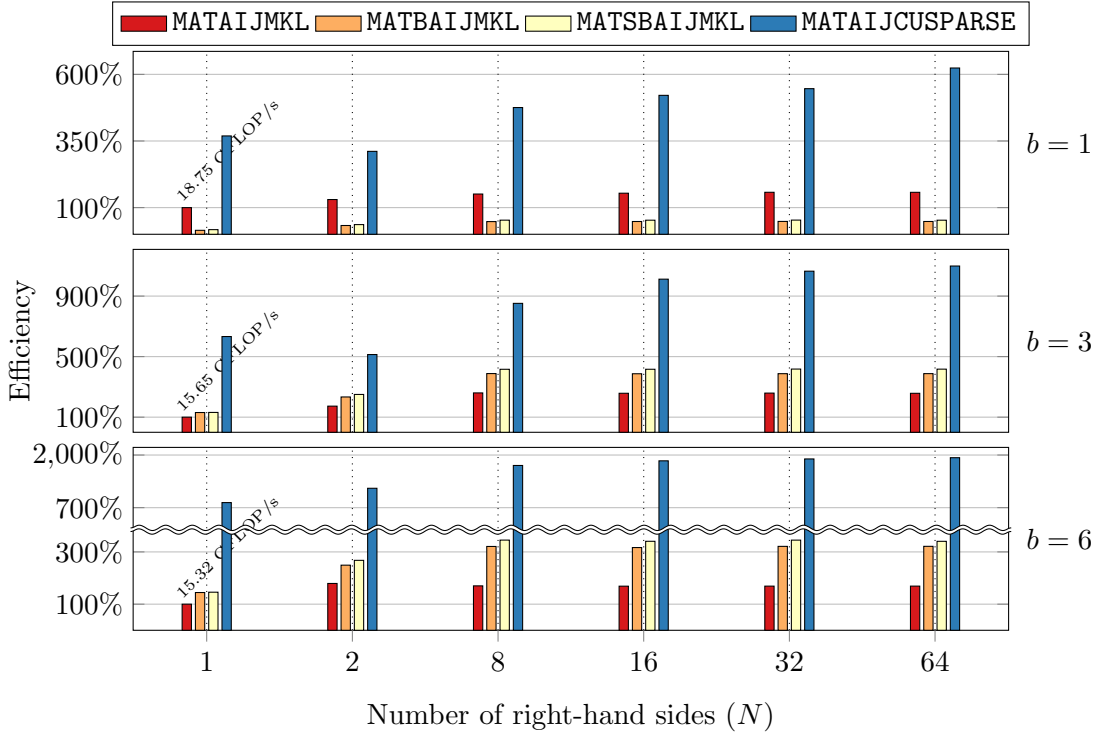


Figure 3: Performance of the matrix–matrix multiplication as implemented in PETSc using intranode parallelism.

242 Results reported in figure 3 have been scaled with respect to the baseline implementation MATAIJMKL
 243 with $N = 1$. They have been obtained on a single node of Jean Zay, a system composed of 261 nodes
 244 with two 20-core Intel Xeon Gold 6248 clocked at 2.5 GHz and four NVIDIA Tesla V100 SXM2.
 245 Neither transfers between host and device nor time spent in `mkl_sparse_optimize` have been taken
 246 into account. For the sake of completeness, the performance of the matrix–vector multiplication
 247 with MATAIJMKL and different numbers of threads is also reported in figure 4. Clearly, it is not
 248 advised to use the full socket to perform this type of workload for such small sparse matrices and
 249 skinny dense matrices. However, these results help in drawing a fair comparison between a full
 250 CPU socket and a GPU device.

251 The CPU configuration which reaches the highest percentage of peak is MATSBAIJMKL with
 252 $b = 3$ and $N = 64$, rightmost middle plot in figure 3, with approximately 63 GFLOP/s. This format
 253 is not yet available in PETSc, but it is handled by the mini-app. Still, it is less than 1% of peak for
 254 an Intel Xeon Gold 6248. On the GPU, MATAIJCUSPARSE with $b = 6$ and $N = 64$, rightmost
 255 bottom plot in figure 3, performs at around 306 GFLOP/s, about 4% of peak for an NVIDIA
 256 Tesla V100 SXM2. Future work may consider using interlaced layouts for storing the dense right-
 257 hand sides and extend the current PETSc functionality in order to maximize performance while
 258 maintaining interface flexibility and the user friendliness of the library.

259 *2.3.3. Linear stability analysis*

260 In this section, we apply recycling Krylov methods in the context of linear stability analysis
 261 and consider the solution of the following generalized eigenvalue problem:

$$J(q_b)x = \lambda \begin{bmatrix} M & 0 \\ 0 & 0 \end{bmatrix} x, \quad (1)$$

262 where J is the Jacobian of the incompressible steady-state Navier–Stokes equation evaluated using
 263 a given base flow $q_b = \begin{bmatrix} u_b \\ p_b \end{bmatrix}$ and M is the discretization of the mass matrix on the space of
 264 velocities. The mini-app employed for the numerical results is built on top of FreeFEM [50] and it
 265 is available at <https://github.com/prj-/moulin2019a1>. Interested readers are referred to [51]
 266 for more details and for larger runs.

267 There are different methods to compute the eigenvalues of equation (1) near a complex-valued
 268 shift σ . In this paragraph we consider a Krylov–Schur method [52], which, for interior eigenvalues,
 269 relies on spectral transformations and on the solution of successive linear systems such as:

$$\left(J(q_b) - \sigma \begin{bmatrix} M & 0 \\ 0 & 0 \end{bmatrix} \right) x_i = b_i, \quad (2)$$

270 which, in SLEPc, are parameterized using the `-st_` prefix. In this work, the above linear system
 271 is preconditioned using a modified augmented Lagrangian approach [53, 51]

272 The effectiveness of subspace recycling is shown for finding the 5 eigenpairs closest to the shift
 273 $\sigma = 10^{-6} + 0.6i$ for a flow past a cylinder at Reynolds 100. Results can be reproduced using the
 274 following four commands. We note here that the first two commands are merely used to generate
 275 the base flow q_b with a SNES, a PETSc object used to solve nonlinear problems, using a continuation
 276 method on the Reynolds number.
 277

```

$ mpirun -n 4 FreeFem++-mpi Nonlinear-solver.edp -Re 50 -v 0
$ mpirun -n 4 FreeFem++-mpi Nonlinear-solver.edp -Re 100 -v 0
$ mpirun -n 4 FreeFem++-mpi Eigensolver.edp -Re 100 -v 0 -st_ksp_rtol 1.0e-4 \
-st_ksp_type fgmres -st_ksp_gmres_restart 200 -st_ksp_converged_reason \
$ mpirun -n 4 FreeFem++-mpi Eigensolver.edp -Re 100 -v 0 -st_ksp_rtol 1.0e-4 \
-st_ksp_type hpddm -st_ksp_gmres_restart 200 -st_ksp_converged_reason \
-st_ksp_hpddm_variant flexible -st_ksp_hpddm_recycle 10 -st_ksp_hpddm_type gcrodr
  
```

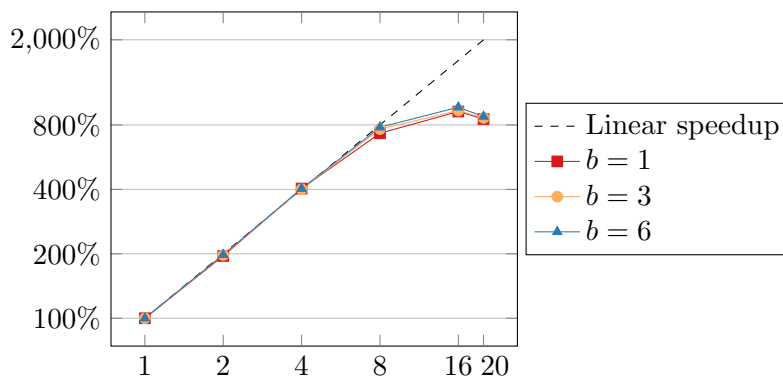


Figure 4: Scalability of MatMult with MATAIJMKL and different number of threads.

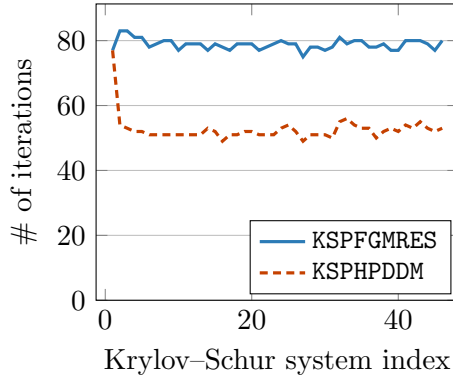


Figure 5: Number of inner iterations for systems equation (2) with - - - or without — recycling.

278 The Krylov-Schur algorithm converges in six outer iterations (restarts), with a total of 46 inner
 279 solves equation (2). Recycling of Krylov subspaces with KSPHPDDM provides algorithmic speedup by
 280 lowering the number of iterations per inner solve, as reported in figure 5, from 80 with KSPFGMRES
 281 to 45. A similar speedup can be observed in terms of runtime, with the time spent for the solution
 282 of the eigenvalue problem being reduced from 25.1 min to 17.5 min. These timings, and all that
 283 follow, have been obtained on Irène, a system composed of 1,656 nodes with two 24-core Intel Xeon
 284 Platinum 8168 clocked at 2.7 GHz.

285 2.3.4. Blocking inside SLEPc

286 The implementations of the locally optimal block preconditioned conjugate gradient [26] (LOBPCG)
 287 and the contour integral spectrum slicing method [54] (CISS) in SLEPc were previously not using
 288 blocking when applying the preconditioner or solving linear systems. In version 3.14.0, these solvers
 289 have been adapted to employ `KSPMatSolve` and `PCMatApply`. In the case of LOBPCG, which is
 290 a purely blocked method, all steps were already implemented as block operations, except the ap-
 291 plications of the preconditioner, which undermined the benefits of blocking. The performance of
 292 LOBPCG implemented in SLEPc as `EPSLOBPCG` is studied using three different preconditioners:

- 293 • **PCASM**: one-level overlapping Schwarz method with one level of overlap and exact Cholesky
 294 factorizations in each subdomain;
- 295 • **PCHPDDM**: multilevel overlapping Schwarz method, see section 3;
- 296 • **PCGAMG**: algebraic multigrid method [55].

297 More details on these solvers are given in the next section where focus is put on preconditioning
 298 rather than Krylov methods. We note here that only PCASM and PCHPDDM currently handle blocking,
 299 see the list section 2.2, and that PCHPDDM and PCGAMG have lower contraction factors than the
 300 cheaper alternative PCASM.

As a testbed, we consider the following generalized eigenvalue problem on a three-dimensional cube:

$$-\nabla \cdot (\kappa \nabla u) = \lambda u.$$

301 This continuous equation is discretized with third-order Lagrange finite elements using FreeFEM.
 302 The script `blocking-slepc.edp` available at <https://github.com/prj-/joliviet2020petsc>
 303 can be used for reproducibility. The following timings and convergence histories have been obtained
 304 on 2,304 processes of Irène, for solving a problem of dimension $7.46 \cdot 10^7$.

305 By default, SLEPc applies at most 5 iterations of GMRES on each column of a so-called set of
 306 active columns. Since the option `-eps_lobpcg_blocksize 40` is used for computing 20 eigenpairs,
 307 the size of the set varies between 40 to 1. This size corresponds to the number of right-hand sides
 308 that will be solved for throughout LOBPCG iterations. Instead of using KSPGMRES, which does
 309 not handle blocking, KSPHPDDM is used, with the pseudo-block GMRES. Thus, assuming the size of
 310 the set of active columns is 40, one LOBPCG iteration performs at most 5 pseudo-block GMRES
 311 iterations with 40 vectors simultaneously, using `MatProductNumeric` and `PCMatApply`, instead of
 312 doing at most 40×5 successive GMRES iterations, using `MatMult` and `PCApply`.

313 Results are gathered in table 1. In the first row, the numbers of outer LOBPCG iterations
 314 are reported with the three different inner preconditioners mentioned above. On the one hand,
 315 PCASM is known to yield a preconditioned operator whose condition number grows as the number
 316 of subdomains, here processes, increases. Thus, the number of outer iterations is higher than with
 317 PCHPDDM or PCGAMG since the inner solves are not converging to meaningful solutions in just 5
 318 iterations. On the other hand, both multilevel preconditioners perform similarly, with a minimal
 319 difference in the number of outer iterations. The second and third rows record the number of times
 320 `PCApply` and `PCMatApply` are called. Since PCGAMG does not handle blocking, at each pseudo-
 321 block GMRES iteration, `PCApply` is called for each active column separately. PCASM and PCHPDDM
 322 handle blocking, so they only rely on `PCMatApply`. In the fourth row, the time spent setting up
 323 the preconditioners is given. PCASM is extremely cheap but not very robust numerically, while
 324 both multilevel preconditioners have a similar setup time. The time spent in the full eigensolver
 325 is reported in the last row. Clearly, having both a KSP and a PC which can efficiently deal with
 326 multiple vectors is mandatory to achieve reasonable performance with such an outer solver that
 heavily relies on blocking.

Table 1: Performance of EPSSLOBPCG with three different inner preconditioners: PCASM and PCHPDDM, which utilize `PCMatApply`, and PCGAMG, which utilizes `PCApply`.

	PCASM	PCHPDDM	PCGAMG
# of outer iterations	54	14	15
<code>PCApply</code>	—	—	4,049
<code>PCMatApply</code>	378	98	—
<code>PCSetUp</code> (sec)	2.9	21.6	26.5
<code>EPSSolve</code> (sec)	265.7	121.6	391.8

327 The script available at <https://github.com/prj-/joliviet2020petsc> may be used to solve
 328 the same eigenproblem using EPSCISS by providing the additional command line argument `-eps_type`
 329 `ciss`. However, this eigensolver is mostly suited for finding interior eigenpairs, **whereas the left-**
 330 **most part of the spectrum, closest to 0 for a symmetric positive definite problem, is here sought.**
 331 **In this scenario, EPSCISS is not a suitable solver and** for fairness, its **algorithmic** performance
 332 is not reported, but the options from the script can be readily used for problems where it is an
 333 appropriate choice. We finally note that blocking support is also available and tested in SLEPc for
 334 EPSSUBSPACE since version 3.14.0.

336 3. Robust multilevel overlapping Schwarz preconditioners: PCHPDDM

337 3.1. Related work

338 Domain decomposition methods are, alongside multigrid methods, one of the dominant paradigms
 339 for defining efficient and robust preconditioners in modern large-scale applications dealing with
 340 partial differential equations. There are many monographs on domain decomposition methods
 341 [56, 57, 58, 59, 60, 61].

342 Basic one-level methods such as the block Jacobi and the (overlapping) additive Schwarz method
 343 are implemented in PETSc as PCBJACOBI and PCASM respectively, with some additional customiza-
 344 tion [62] available for PCASM. However, none of these methods provide automatic support for a
 345 coarse-level solver, which is mandatory to obtain algorithmic scalability with large numbers of
 346 processes. Furthermore, PETSc provides non-overlapping domain decomposition methods. PCNN
 347 implements a basic balancing Neumann–Neumann method [63], while more advanced precondition-
 348 ing techniques are available with PCBDDC [64], including adaptive selection of primal constraints [65],
 349 support for $H(\text{curl})$ [66] and $H(\text{div})$ [67] conforming finite elements, and isogeometric analysis [68].
 350 For multigrid, PCMG provides a unified framework for geometric or algebraic preconditioners, which
 351 is used for the implementation of the smoothed-aggregation PCGAMG [55]. There are also interfaces
 352 to other well-known multigrid packages such as PCHYPRE [4] or PCML [69] from Trilinos. Of course,
 353 there are many other available domain decomposition preconditioners, either accessible through
 354 high-level libraries other than PETSc, e.g., FROSch [70] in Trilinos, or as stand-alone packages,
 355 e.g., BDDCML [71] or FEMPAR-BDDC [72].

356 HPDDM implements the same one-level overlapping Schwarz methods as PETSc as well as
 357 optimized Schwarz methods [73], which may be used in PETSc with PCSetModifySubMatrices,
 358 see [74]. It also provides support for defining robust two-level methods using the generalized
 359 eigenvalue problem on the overlap (GenEO) framework, for finite elements [75, 76] and boundary
 360 elements [77]. Results in this work present the first high-level access to GenEO from PETSc and
 361 its composable solver infrastructure [78]. We note that GenEO has been recently implemented in
 362 other libraries as well [79].

363 3.2. Interfacing HPDDM overlapping Schwarz methods in PETSc

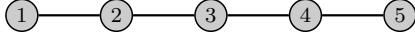
364 PCHPDDM, the interface between HPDDM overlapping Schwarz methods and PETSc, is au-
 365 tomatically registered since PETSc version 3.12 when configuring PETSc with the extra flag
 366 `--download-hpddm --download-slepc`. It is then possible to select the corresponding PC using
 367 the command line option `-pc_type hpddm`, or the routine `PCSetType(pc, PCHPDDM)`.

368 In this section, we are interested in constructing a preconditioner for a given coefficient matrix
 369 A . Without user intervention, PCHPDDM is strictly equivalent to PCASM. That is, if A is distributed
 370 among N processes, the action of the preconditioner M^{-1} is:

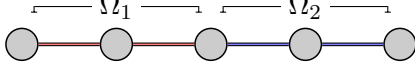
$$371 M^{-1} = \sum_{i=1}^N \tilde{R}_i^T (R_i A R_i^T)^{-1} R_i, \quad (3)$$

372 where $\{R_i\}_{i=1}^N$ are restriction operators that act on a global vector and return a local vector on
 373 each process, possibly with some overlap. $\{\tilde{R}_i\}_{i=1}^N$ are the same operators except that coefficients
 374 on the overlap are set to 0 [62]. The action of each $\{R_i A R_i^T\}_i^{-1}$ can be parameterized through
 375 a local KSP. When A is the discretization of a linear partial differential operator \mathcal{L} on a domain
 Ω , it is extremely common to exhibit parallelism by distributing Ω on N processes, possibly with

Domain and global unknown numbering

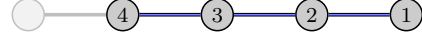
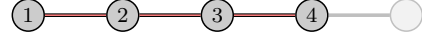


Two-way element-based partitioning



$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Local numbering with overlap



$$R_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} \quad \tilde{R}_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\dot{A}_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix} \quad R_2 A R_2^T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix}$$

Figure 6: Notation and basics of overlapping Schwarz methods.

376 some overlap, and to construct the $\{R_i\}_{i=1}^N$ so that they map global unknowns to unknowns local
377 to each process.

378 In addition to the local operators $\{R_i A R_i^T\}_{i=1}^N$, GenEO needs users to supply the local unassem-
379 bled matrices $\{\dot{A}_i\}_{i=1}^N$, also known as Neumann matrices, representing the discretization of \mathcal{L} on the
380 extended local subdomain with overlap, endowed with natural boundary conditions. In the case of
381 the one-dimensional Poisson equation with homogeneous Dirichlet boundary conditions discretized
382 using trilinear finite elements, an explicit representation of these operators is given figure 6 for a
383 two-domain decomposition.

384 With extra Neumann information at hand, it is then possible to enrich the original one-level
385 preconditioner from equation (3) using a spectral coarse grid built in the following way:

386 1. have SLEPc solve the local generalized eigenvalue problem concurrently:

$$\dot{A}_i y_i = \lambda_i \tilde{R}_i R_i^T (R_i A R_i^T) R_i \tilde{R}_i^T y_i; \quad (4)$$

387 2. retrieve the ν_i smallest eigenpairs $\{y_{i,j}, \lambda_{i,j}\}_{j=1}^{\nu_i}$ and assemble a local deflation dense matrix

$$388 \quad W_i = \begin{bmatrix} \tilde{R}_i R_i^T y_{i,1} & \cdots & \tilde{R}_i R_i^T y_{i,\nu_i} \end{bmatrix};$$

389 3. define a global deflation matrix $P = [R_1^T W_1 \quad \cdots \quad R_N^T W_N]$ and a new two-level precondi-
390 tioner using the Galerkin product of A and P :

$$M_{\text{additive}}^{-1} = P (P^T A P)^{-1} P^T + \sum_{i=1}^N \tilde{R}_i^T (R_i A R_i^T)^{-1} R_i. \quad (5)$$

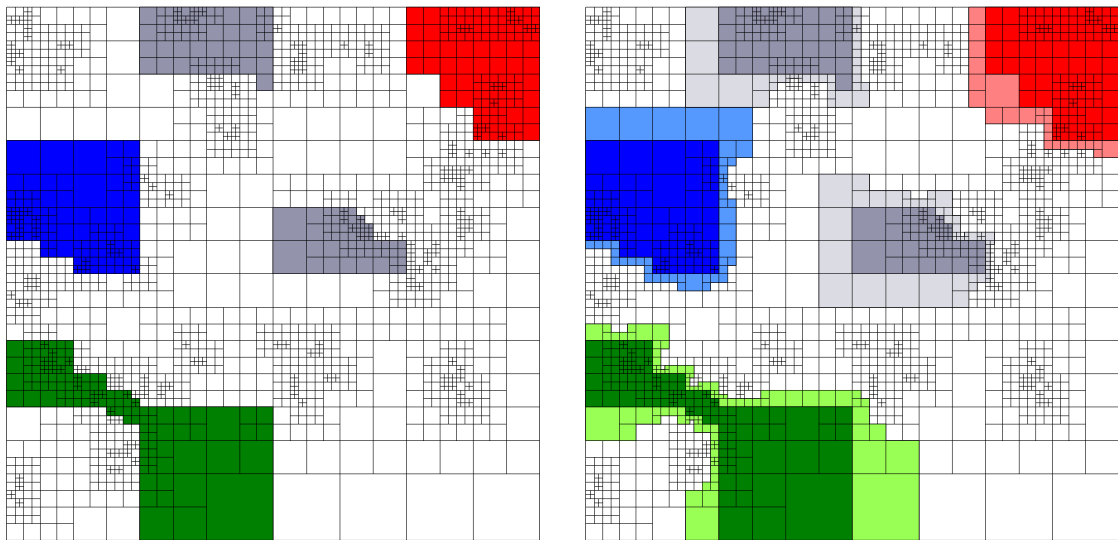
Note that $\{\tilde{R}_i R_i^T\}_{i=1}^N$ defines a partition of unity, i.e.,

$$\sum_{i=1}^N R_i^T \tilde{R}_i R_i^T R_i = I.$$

391 Such a construction of a two-level method yields a preconditioner with which it is possible to
392 bound the condition number of the preconditioned operator. Thus, the convergence rate of a
393 preconditioned Krylov method such as the conjugate gradient can be guaranteed a priori.

394 While step #3 is common to most domain decomposition or multigrid preconditioners, HPDDM
 395 takes advantage of the special block sparse structure of the deflation matrix P . Moreover, the size
 396 of the coarse operator $P^T A P$ is much lower than the size of A , and coarsening is much more
 397 aggressive than when using algebraic or geometric multigrid methods. It is thus natural to remap
 398 the coarse operator on a subset of the processes used to decompose the original matrix A . In
 399 HPDDM, the coarse operator is computed and redistributed simultaneously. Interested readers
 400 are referred to the paper describing the technical details of the implementation [7]. Note that
 401 similar techniques have been proven beneficial for other types of solvers in PETSc [80].

402 In order to use GenEO from PETSc, the auxiliary operators $\{\hat{A}_i\}_{i=1}^N$ and the correspond-
 403 ing local to global map of degrees of freedom, see figure 6, can be provided using the routine
 404 `PCHPDDMSetAuxiliaryMat`. In special cases where the matrix has been constructed using the
 405 PETSc discretization infrastructure, the relevant information is automatically computed by PETSc.
 406 Figure 7a (resp. figure 7b) is an example of a non-overlapping (resp. overlapping) decomposition
 407 using a non-conforming grid with DMPLEX [81]. The mesh has been constructed internally by DMPLEX
 408 using a random pattern of non-conforming 2:1 refinements and a space-filling curve distribution by
 409 `p4est` [6]. The required elements needed to properly assemble the Neumann operators in a conform-
 410 ing way are pictured with a lighter color. Other DMTypes, including user-defined, may be supported
 411 by implementing the proper `DMCreateNeumannOverlap` callback. Support for the structured grid
 infrastructure (DMDA) could be easily added.



(a) Initial non-overlapping decomposition.

(b) Overlapping decomposition on which auxiliary operators $\{\hat{A}_i\}_{i=1}^N$ are assembled.

Figure 7: Automatic generation of overlapping subdomains when using a DMPLEX. For clarity, only four subdomains, one of which is disconnected, are colored.

412 With the local matrix pencils available, new options are automatically registered such as
 413 `-pc_hpddm_levels_1_eps_nev` or `-pc_hpddm_levels_1_eps_threshold`, which may be used to cus-
 414 tomize the eigenpairs SLEPc will compute in step #2. The default `-eps_nev` value is 20, i.e., the
 415 dimension of the coarse space is $20 \times N$ under the assumption that all concurrent SLEPc solves
 416

417 converge.

418 For performance, it is better to compute the same number of eigenvectors on each process,
 419 since in this case, the coarse operator will be assembled using symmetric or general blocked matrix
 420 formats. With usually large block sizes, this has consequent impacts on memory, floating-point,
 421 and message passing performances. In addition, standard eigensolvers such as EPskrylovSchur
 422 from SLEPc, cannot inexpensively evaluate the exact number of eigenpairs in a given interval of
 423 \mathbb{R}^+ . Instead, it is often more efficient to provide an upper bound on the number of eigenpairs.

424 The option `-pc_hpddm_coarse_p` can be used to provide the size of the subcommunicator that
 425 will be used to remap the coarse operator. With the default value of 1, the coarse matrix $P^T AP$
 426 is centralized on a single process. The action of the inverses in equation (5) can be parameterized
 427 through the `-pc_hpddm_coarse_ksp_type` and `-pc_hpddm_levels_1_ksp_type` options respectively.

428 3.2.1. Coarse corrections customization

Some other parameters may be adjusted to define how the one-level preconditioner and the
 coarse-grid operator interact. Besides the additive correction shown in equation (5), end-users
 may select more numerically efficient corrections [82, 83] as:

$$\begin{aligned} M_{\text{deflated}}'^{-1} &= Q + M^{-1} (I - AQ) \quad (\text{default}) \\ M_{\text{balanced}}'^{-1} &= Q + (I - AQ) M^{-1} (I - AQ), \end{aligned}$$

429 with $Q = P (P^T AP)^{-1} P^T$ and M^{-1} defined in equation (3). This is parameterized using the option
 430 `-pc_hpddm_coarse_correction (deflated|additive|balanced)` or the routine
 431 `PCHPDDMSetCoarseCorrectionType`. All of the correction formulas can be applied to either a
 432 single vector or a block of multiple vectors optimally. Indeed, the one-level preconditioner M^{-1} is
 433 applied with either `PCApply` or `PCMatApply`, and the restriction-correction-interpolation operator
 434 Q is applied in a block fashion in HPDDM, and not column by column.

435 3.2.2. Extension of the GenEO framework

The local generalized eigenvalue problems equation (4) can also be adjusted. In addition to
 the local unassembled Neumann matrices $\{\mathring{A}_i\}_{i=1}^N$, users can prescribe additional local operators
 $\{B_i\}_{i=1}^N$, defined on the overlapping decomposition, via the routine `PCHPDDMSetRHSMat(pc, B_i)`. In
 the case of local matrices with optimized Robin transmission conditions, the following generalized
 eigenvalue problems, called GenEO-2 in the literature [84], are then solved in each subdomain:

$$\mathring{A}_i y_i = \lambda_i B_i y_i.$$

436 This may be useful when dealing with nearly indefinite systems, e.g., nearly incompressible elas-
 437 ticity or the system of Stokes. Indeed, for such equations, one needs to compute a large number
 438 of eigenvectors from the classical GenEO eigenproblem equation (4) to generate a robust precon-
 439 ditioner. GenEO-2 alleviates this phenomenon. In the case where the local operators $\{B_i\}_{i=1}^N$ are
 440 the discrete mass matrices on the subdomain interfaces, the so-called Dirichlet-to-Neumann coarse
 441 operator is constructed. This has proven to be efficient for solving the heterogeneous Helmholtz
 442 equation [85].

443 3.2.3. Multilevel extension

444 In the previous sections, it was shown how to supply the required information to PCHPDDM
 445 in order to build robust two-level overlapping domain decomposition preconditioners. Since the

446 dimension of the coarse operator linearly depends on the number of subdomains of the fine level
 447 decomposition, switching to a multilevel scheme may alleviate the increasing cost of solving coarse
 448 linear systems. A multilevel extension of the GenEO framework has been recently proposed [86],
 449 with the advantage that the multilevel hierarchy can be automatically constructed without any
 450 additional information from the user.

451 PCHPDDM follows the same numbering of PCBDDC: the finest level is always numbered with 1,
 452 and the level index increases as the hierarchy is traversed, up until the coarsest level, whose solver
 453 options are prefixed by `coarse_`. In order to register an additional level $l' = l + 1$, the following
 454 conditions must be met at level l :

- 455 • there must be more than one subdomain;
- 456 • at least one local eigenvector must be computed per coarse subdomain.

457 For example, using $N = 16$ subdomains on the finest level, the options

458 `-pc_hpddm_levels_1_eps_threshold 0.4 -pc_hpddm_coarse_p 8`

459 will define a two-level method with the coarse operator being distributed among 8 processes,
 460 assuming that there is globally enough λ_i in equation (4) smaller than the prescribed threshold
 461 0.4. Similarly, the options

462 `-pc_hpddm_levels_1_eps_nev 10 -pc_hpddm_levels_2_p 8`

463 `-pc_hpddm_levels_2_eps_nev 10 -pc_hpddm_coarse_p 2`

464 will define a three-level method with the second level distributed among 8 processes, while the
 465 coarsest level will be built using 10 eigenvalues per subdomain from the second level and remapped
 466 onto 2 processes. Additional examples for the solver customization are provided in section 3.3

467 3.2.4. PCHPDDM for non-overlapping domain decomposition

468 Because of the intrinsic nature of balancing domain decomposition methods, PCNN and PCBDDC
 469 must be supplied with a `MatIS` which stores local unassembled matrices and local to global map-
 470 pings. These local matrices are equivalent to the $\{\hat{A}_i\}_{i=1}^N$ defined section 3.2, assuming the domain
 471 decomposition is without overlap, see for example Ω_1 and Ω_2 from figure 6 in a finite element con-
 472 text. Since the assembled form of these operators can be reconstructed, it is possible to obtain the
 473 Dirichlet operators $\{R_i A R_i^T\}_{i=1}^N$ from section 3.2 as well. All tools needed by the GenEO framework
 474 are thus readily available. Note however that one-level overlapping Schwarz methods are known for
 475 converging slowly when there is no overlap. The proposed methodology is not strictly equivalent
 476 to the BDD-GenEO method [87], in which local Schur complements on subdomain interfaces are
 477 computed explicitly, and then used in concurrent dense generalized eigenvalue problems.

478 3.3. Applications and numerical results

479 3.3.1. System of elasticity

We first report on solving the three-dimensional system of linear elasticity with highly hetero-
 geneous elastic modulus. Its strong formulation is given by:

$$\begin{aligned}
 \operatorname{div} \sigma(u) + f &= 0 & \text{in } \Omega, \\
 u &= 0 & \text{on } \Gamma_D, \\
 \sigma(u) \cdot n &= 0 & \text{on } \Gamma_N.
 \end{aligned} \tag{6}$$

480 The physical domain Ω is a beam of dimensions $[0, 6] \times [0, 1] \times [0, 1]$. The Cauchy stress tensor σ
 481 is given by Hooke's law: it can be expressed in terms of Young's modulus E and Poisson's ratio ν ,

$$\sigma_{ij}(u) = \begin{cases} 2\mu\varepsilon_{ij}(u) & i \neq j, \\ 2\mu\varepsilon_{ii}(u) + \lambda\text{div}(u) & i = j, \end{cases}$$

482 where

$$\varepsilon_{ij}(u) = \frac{1}{2} \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right), \quad \mu = \frac{E}{2(1+\nu)}, \quad \text{and} \quad \lambda = \frac{E\nu}{1-2\nu}.$$

483 Γ_D is the subset of the boundary of Ω corresponding to $x = 0$. Γ_N is defined as the complementary
 484 of Γ_D with respect to the boundary of Ω . Equation (6) is discretized using trilinear finite elements
 485 resulting in 593×10^6 unknowns with approximately 45 nonzero coefficients per row in the discrete
 486 coefficient matrix. The physical domain Ω is decomposed in 13,824 subdomains using the automatic
 487 graph partitioner ParMETIS [88]. There are heterogeneities due to jumps in E and ν . The following
 488 discontinuous piecewise constant values are considered: $(E_1, \nu_1) = (2 \times 10^{11}, 0.25)$ in blue regions,
 489 while $(E_2, \nu_2) = (10^7, 0.45)$ in red regions, see figure 8. The code used to produce these results was
 490 borrowed from the original paper about the multilevel extension of GenEO [86] and it is available
 at <https://github.com/prj-/aldaas2019multi>.

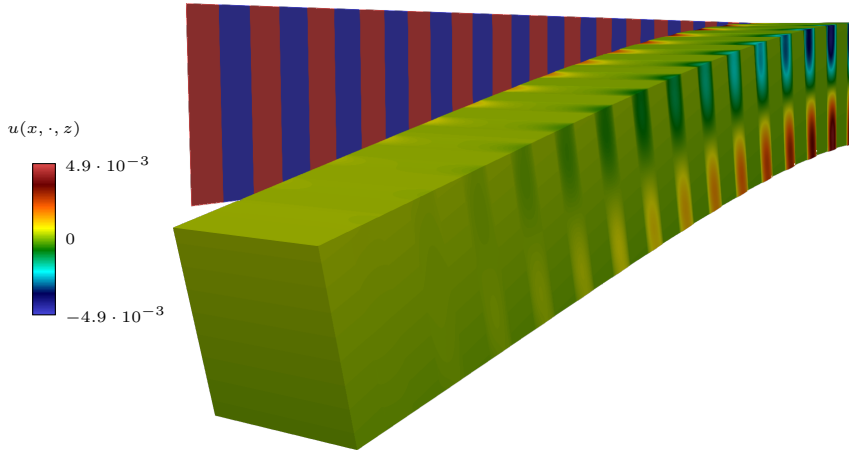


Figure 8: Deformed geometrical configuration of a 3D clamped beam subject to gravity. The striped plane in the background shows a cut of the resting configuration with the jumps in the coefficient $E_1 = 10^7$ and $E_2 = 2 \times 10^{11}$, aligned with those of ν .

491 The following preconditioners are compared:
 492

- 493 1. PCGAMG;
- 494 2. PCHPDDM with an exact coarse solver;
- 495 3. PCHPDDM with inexact coarse solvers using GenEO multilevel extension.

496 The flexible GMRES is used and customized with the options `-ksp_gmres_modifiedgramschmidt`
 497 `-ksp_gmres_restart 50 -ksp_type fgmres`. The options specific to each solver described above
 498 are given below.

499

```

# cf. item 1
-pc_type gamg
-prefix_push pc_gamg_
-threshold 0.03
-square_graph 4
-sym_graph true
-asm_use_agg true
-repartition true
-prefix_pop
-prefix_push mg_levels_
-pc_asm_overlap 0
-sub_pc_type cholesky
-prefix_pop
-prefix_push mg_coarse_
-pc_type redundant
-redundant_pc_type cholesky
-prefix_pop

```

```

# cf. item 2
-pc_type hpddm
-prefix_push pc_hpddm_
-prefix_push levels_1_
-pc_type asm
-eps_nev 40
-sub_pc_type cholesky
-sub_pc_factor_mat_solver_type mumps
-st_pc_factor_mat_solver_type mumps
-prefix_pop
-prefix_push coarse_
-p 24
-pc_factor_mat_solver_type mkl_cpardiso
-prefix_pop
-define_subdomains
-has_neumann
-prefix_pop

```

500 By default, the coarse problem in PCHPDDM is solved using an exact LU or Cholesky factorization,
501 in this case using 24 processes and MKL CPARDISO. Switching to an inexact solver using the
502 multilevel extension of GenEO, cf. item 3, is straightforward.

503

```

# cf. item 3
-prefix_push pc_hpddm_levels_2_
-p M
-ksp_type gmres
-ksp_rtol 1.0e-2
-ksp_pc_side right
-pc_type asm
-eps_nev 80
-sub_pc_type cholesky
-sub_pc_factor_mat_solver_type mkl_pardiso
-st_pc_factor_mat_solver_type mkl_pardiso
-prefix_pop

```

504 In this third configuration, M is the number of subdomains used to define the second-level domain
505 decomposition, which is an aggregation of first-level subdomains.

506 Considering the parametrization used for the PCHPDDM solvers, items 2 and 3 yield the same
507 number of outer iterates. In figure 9a, the convergence history of PCGAMG and PCHPDDM are reported.
508 In figure 9b, the number of inner coarse iterations are reported for various values of M . In the
509 case of a two-level method (item 2) this number is equal to one and is thus not reported. Note
510 that in this test case, PCGAMG is faster than all PCHPDDM alternatives. On the one hand, for PCGAMG,
511 the setup phase takes 70 seconds, while the solution phase requires 19 seconds. On the other
512 hand, the fastest PCHPDDM configuration, which corresponds to the configuration with 256 second-
513 level subdomains ----, requires 64 seconds to setup and 36 seconds for the solution of the linear
514 system. Out of the 64 seconds needed for PCHPDDM setup, 31 are spent in SLEPc EPSSolve for
515 solving GenEO at each level but the coarsest. With respect to coarsening, PCGAMG has an operator
516 complexity of 1.501, while for PCHPDDM, it is 1.015.

517 3.3.2. Liouville–Bratu–Gelfand equation

518 The strong formulation of this nonlinear problem is given by:

$$-\nabla \cdot (\kappa \nabla u) - 6.2e^u = 0, \quad (7)$$

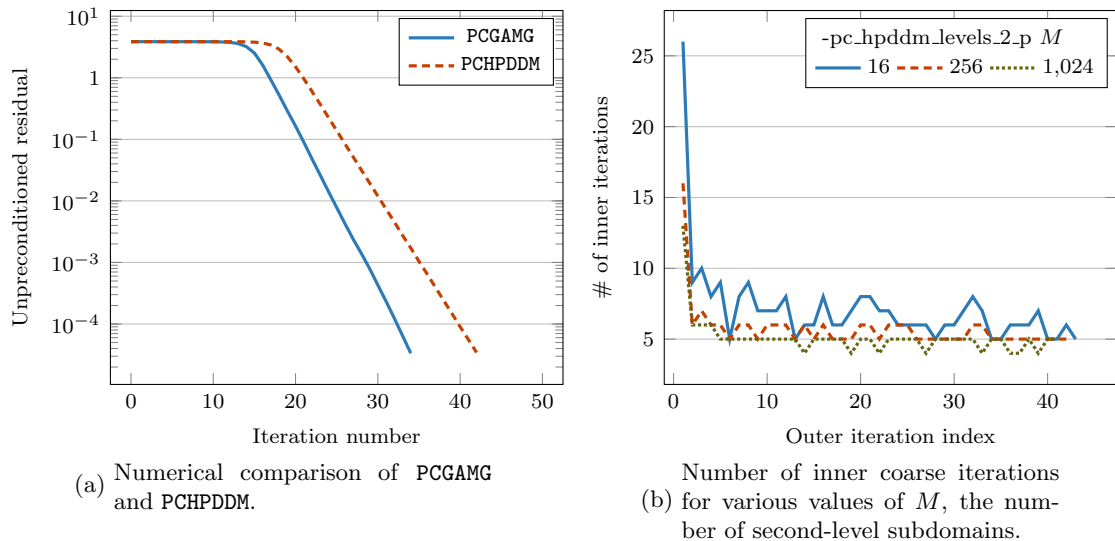


Figure 9: Convergence histories of PCGAMG and PCHPDDM for solving the 3D system of linear elasticity. For PCHPDDM, a comparison of the number of inner coarse iterations is displayed when using an inexact method, cf. item 3.

519 where κ is a heterogeneous coefficient distribution, see figure 10a. This equation may model the
 520 temperature distribution in combustion models. Here, it is merely used to test PCHPDDM in the
 521 context of solving successive linearized equations. The physical domain Ω is the unit cube. It is
 522 decomposed in 10,272 subdomains. Equation (7) is discretized using second-order Lagrange finite
 523 elements. The number of unknowns is 217×10^6 , with approximately 29 nonzero coefficients per
 524 row. It is solved using SNES, with PCHPDDM being the preconditioner for solving each linearized
 525 system, see figure 10b. The GenEO framework has seldom been used in the context of nonlinear
 526 problems, but here, it is shown that it can solve the linearized equations from equation (7) in few
 527 iterates. There are two strategies to define the preconditioner:

- 528 • whenever the Jacobian is being updated in the `SNESetJacobian` function, new unassembled
 529 Neumann matrices at the current linearization point can be updated as well and supplied via
 530 `PCHPDDMSetAuxiliaryMat`
- 531 • by reusing the PCHPDDM hierarchy assembled when solving the first linearized system, using
 532 the `SNESetLagPreconditioner` option.

533 Since the system is not too stiff here, both strategies lead to the same convergence history, which
 534 is displayed in figure 11a. The complete set of options is given in figure 11b. In the case where the
 535 preconditioner is being rebuilt at each of the four linearization steps, 81.1s are spent in `PCSetUp`
 536 and 27.7s in `KSPSolve`. Setting up the preconditioner only once is in this scenario a compelling way
 537 of amortizing this cost. Results can be reproduced using the FreeFEM script `bratu.edp` available
 538 at <https://github.com/prj-/jolivet2020petsc>.

539 3.3.3. Using PCHPDDM as a coarse grid solver for PCBDDC

A last problem shows how one may switch between a multilevel BDDC solver to a two-level
 BDDC solver using PCHPDDM for solving the coarse problem. The problem is generated using
 the MFEM [89] definite Maxwell example available at <https://github.com/mfem/mfem/blob/master/examples/petsc/ex3p.cpp>. The strong formulation of the continuous problem is given

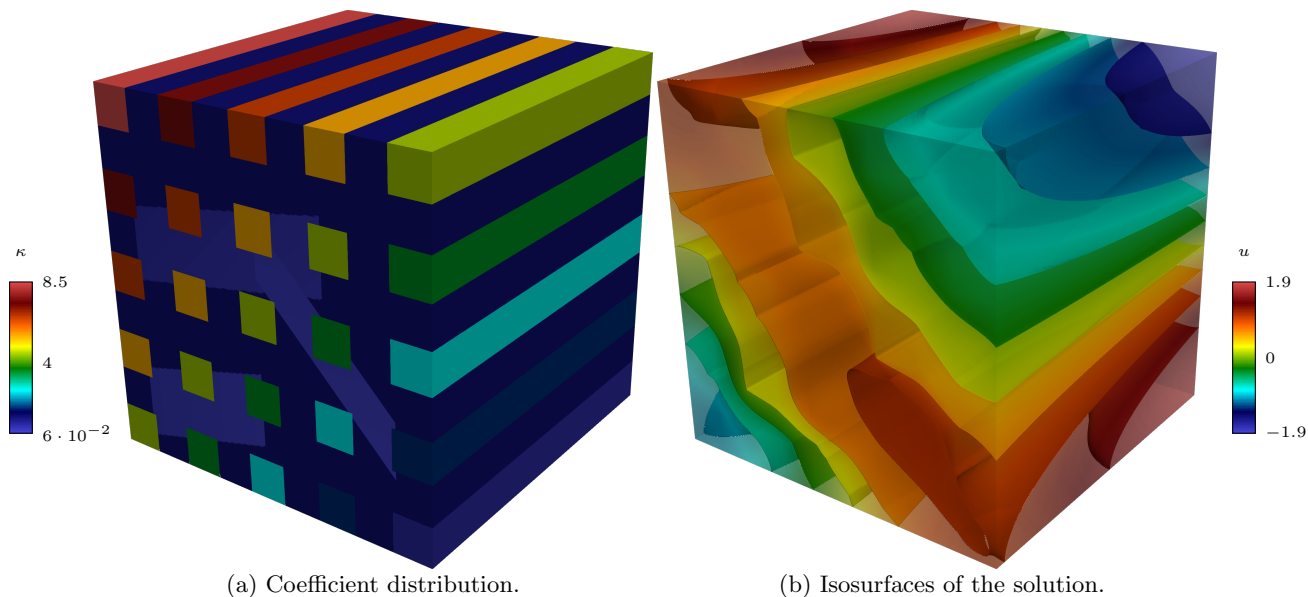


Figure 10: Solving the Liouville–Bratu–Gelfand equation using SNESNEWTONLS and PCHPDDM.

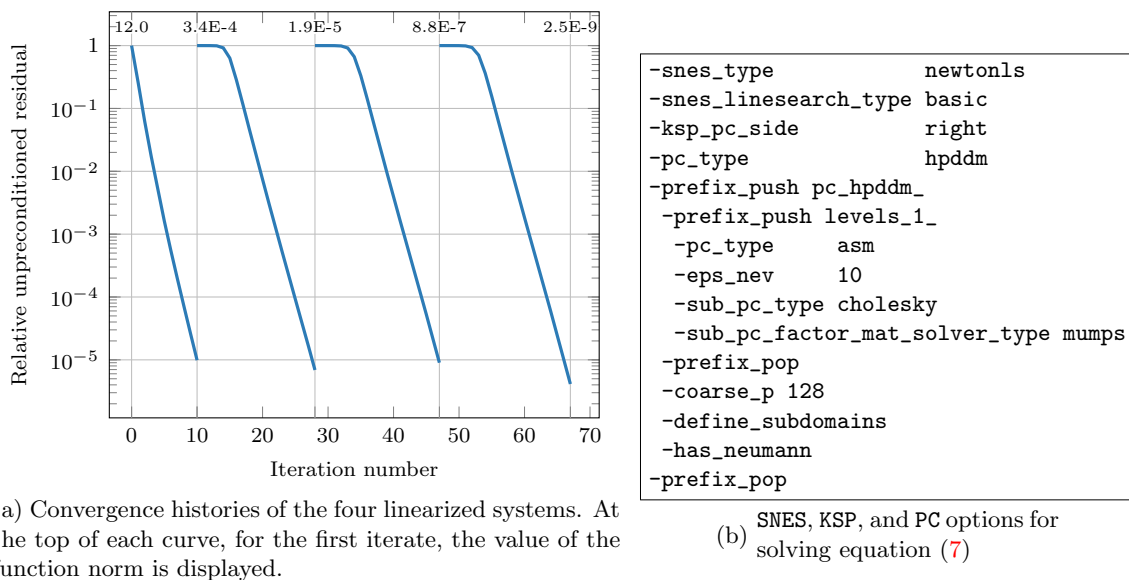


Figure 11: Numerical performance of PCHPDDM in a nonlinear context for solving the Liouville–Bratu–Gelfand equation. Default PETSc tolerances are used: 10^{-5} decrease of relative residuals for linear solves, 10^{-8} decrease of function norms for nonlinear solves.

by:

$$\begin{aligned} \nabla \times (\nabla \times E) + E &= 0 \quad \text{in } \Omega, \\ E \times n &= (1 + 16\pi^2) [\sin 4\pi y \quad \sin 4\pi z \quad \sin 4\pi x]^T \quad \text{on } \partial\Omega. \end{aligned} \tag{8}$$

540 It is discretized using first order Nédélec finite elements and PCBDDC is automatically customized
 541 to obtain a stable method with these elements [66]. The number of unknowns is 5.6×10^6 . The
 542 following command line options are used.

543

```
$ mpirun -n 512 ./ex3p -m ../../data/fichera.mesh -f 4 --petscopts rc_ex3p_bddc --nonoverlapping
```

544 The option file shows how PCHPDDM may be composed into PCBDDC.

545

<pre>\$ cat rc_ex3p_bddc -ksp_type fgmres -ksp_norm_type unpreconditioned -ksp_rtol 1.0e-8 -prefix_push pc_bddc_ -use_deluxe_scaling -levels 1 -adaptive_threshold 2.0 -coarsening_ratio 8 -neumann_pc_factor_mat_solver_type mumps -dirichlet_pc_factor_mat_solver_type mumps # continue on the right column</pre>	<pre># continued from the left column -prefix_push coarse_ -ksp_converged_reason -ksp_type gmres -ksp_max_it 100 -ksp_rtol 1.0e-1 -pc_type hpddm -ksp_norm_type preconditioned -prefix_push pc_hpddm_ -levels_1_pc_type asm -levels_1_eps_nev 10 -levels_1_sub_pc_type cholesky -coarse_pc_type cholesky -prefix_pop -prefix_pop -prefix_pop</pre>
--	---

546 Deluxe scaling is used to build an adaptive second level with PCBDDC. On this second level with
 547 16,284 unknowns, new subdomains are built by aggregating the coarse element matrices of 8 fine-
 548 level subdomains. The BDDC coarse problem composed of $\frac{512}{8} = 64$ subdomains is then solved
 549 using PCHPDDM, which itself builds another adaptive level using 10 local eigenvectors per subdomain.
 550 The coarsest level is aggregated on a single process and solved using an exact Cholesky factoriza-
 551 tion. The magnitude of the electric field is plotted in figure 12a. The convergence history of the
 552 solver is shown in figure 12b. The outer solver reaches the prescribed convergence tolerance in 17
 553 iterations ----. The number of inner iterations for solving the BDDC coarse problem is reported
 554 as well —.

555 4. Conclusion and perspectives

556 In this paper, we presented the interface between PETSc and HPDDM and discussed the new
 557 Krylov and overlapping Schwarz methods, providing several numerical examples and describing
 558 various runtime options. Overall, the interface paves the way for having recycling and block
 559 Krylov methods plus robust overlapping Schwarz methods using the GenEO framework in PETSc.

560 Concerning PETSc, the infrastructure for dealing with blocks of vectors has been laid out.
 561 However, only a little fraction of the built-in matrix types and preconditioners can currently handle
 562 them efficiently, and additional storage formats may be considered to maximize performance.

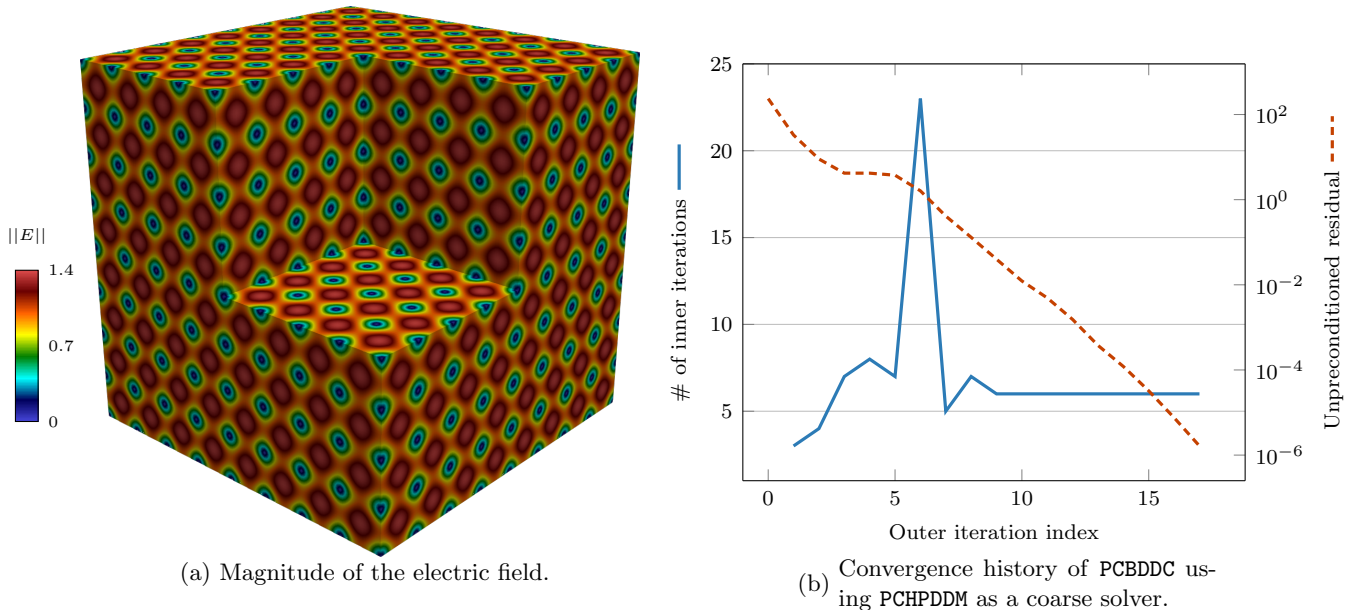


Figure 12: MFEM example ex3p for solving the Fichera corner problem.

563 Concerning SLEPc, we plan to implement new solvers or extend existing ones to further exploit
 564 block solve primitives, for instance, block Krylov–Schur [90]. The current code for Arnoldi iterations
 565 would then handle block Arnoldi iterations, and shift-and-invert solves would operate on the whole
 566 block instead of column by column.

567 Concerning HPDDM, its integration inside PETSc offers a much-needed flexibility compared
 568 to the standard implementation. Indeed, it is now possible to decide at runtime many solvers
 569 parameters that are instead determined at compile-time in a stand-alone HPDDM implementation.

570 The interface between PETSc and HPDDM has room for future improvements. First, HPDDM
 571 handles mixed-precision, e.g., using single-precision scalars for assembling coarse operators while
 572 using double-precision scalars at the fine level. The topic of mixed-precision is under scrutiny for
 573 the next major overhaul of PETSc. Second, HPDDM does not handle GPU efficiently currently,
 574 but the interface with PETSc provides a very thorough testbed with Mat and PC implementations
 575 that do exploit GPU.

576 Acknowledgments

577 The authors would like to thank S. Balay, J. Brown, V. Hapla, M. Knepley, and B. Smith
 578 for reviewing the successive merge requests in PETSc repository and for their feedback on this
 579 manuscript. This work was granted access to the GENCI-sponsored HPC resources of:

- 580 • TGCC@CEA under allocation A0070607519;
- 581 • IDRIS@CNRS under allocation AP010611780.

582 Jose E. Roman was supported by the Spanish Agencia Estatal de Investigación (AEI) under project
 583 SLEPc-DA (PID2019-107379RB-I00).

584 Appendix A. Code reproducibility

585 In order to reproduce the various results from this paper, a short how-to is provided in this
586 appendix to get a minimalist functioning set of required libraries. First, PETSc version 3.14.2
587 can be downloaded at [https://gitlab.com/petsc/petsc/-/archive/v3.14.2/petsc-v3.14.2.](https://gitlab.com/petsc/petsc/-/archive/v3.14.2/petsc-v3.14.2.zip)
588 [zip](https://gitlab.com/petsc/petsc/-/archive/v3.14.2/petsc-v3.14.2.zip). While some adjustments may be needed to ensure that the proper MPI implementation and
589 BLAS/LAPACK libraries are used by PETSc build system, the configure line should be somehow
590 similar to:

```
./configure --download-hypre --download-metis  
--download-slepc --download-hpddm --download-mfem  
--with-scalar-type=real PETSC_ARCH=real-build
```

591 MFEM will then be available for reproducing results from section 3.3.3. As a stand-alone library,
592 PETSc can be used to reproduce results from sections 2.3.1 and 2.3.2. After building PETSc with
593 real scalars, it must also be built with complex scalars.

```
./configure --with-metis-dir=real-build  
--download-slepc --download-hpddm  
--with-scalar-type=complex PETSC_ARCH=complex-build
```

594 Eventually, one can download FreeFEM version 4.7-1 at [https://github.com/FreeFem/FreeFem-sources/](https://github.com/FreeFem/FreeFem-sources/archive/v4.7-1.zip)
595 [archive/v4.7-1.zip](https://github.com/FreeFem/FreeFem-sources/archive/v4.7-1.zip) and use the following configure line:

```
./configure --with-petsc=${PETSC_DIR}/real-build/lib  
--with-petsc_complex=${PETSC_DIR}/complex-build/lib
```

596 After building FreeFEM, results from sections 2.3.3, 2.3.4, 3.3.1 and 3.3.2 can be reproduced.

597 References

- 598 [1] R. Bartlett, I. Demeshko, T. Gamblin, G. Hammond, M. A. Heroux, J. Johnson, A. Klinvex, X. S. Li, L. C.
599 McInnes, J. D. Moulton, D. Osei-Kuffuor, J. Sarich, B. F. Smith, J. Willenbring, U. M. Yang, [xSDK foundations:](#)
600 [Toward an extreme-scale scientific software development kit](#), Supercomputing Frontiers and Innovations 4 (1)
601 (2017).
602 URL <https://xsdk.info>
- 603 [2] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, A. Dener, V. Eijkhout,
604 W. D. Gropp, D. Karpeyev, D. Kaushik, M. G. Knepley, D. A. May, L. C. McInnes, R. T. Mills, T. Munson,
605 K. Rupp, P. Sanan, B. F. Smith, S. Zampini, H. Zhang, H. Zhang, [PETSc web page](#) (2020).
606 URL <http://www.mcs.anl.gov/petsc>
- 607 [3] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, A. Dener, V. Eijkhout,
608 W. D. Gropp, D. Karpeyev, D. Kaushik, M. G. Knepley, D. A. May, L. C. McInnes, R. T. Mills, T. Munson,
609 K. Rupp, P. Sanan, B. F. Smith, S. Zampini, H. Zhang, H. Zhang, PETSc users manual, Tech. Rep. ANL-95/11
610 - Revision 3.13, Argonne National Laboratory (2020).
- 611 [4] R. Falgout, U. M. Yang, [hypre: A library of high performance preconditioners](#), Computational Science—ICCS
612 2002 (2002) 632–641.
613 URL <https://www.llnl.gov/casc/hypre>
- 614 [5] H. Si, [TetGen: A quality tetrahedral mesh generator and 3D Delaunay triangulator](#), Tech. Rep. 13 (2013).
615 URL <http://wias-berlin.de/software/tetgen>
- 616 [6] C. Burstedde, L. C. Wilcox, O. Ghattas, [p4est: Scalable algorithms for parallel adaptive mesh refinement on](#)
617 [forests of octrees](#), SIAM Journal on Scientific Computing 33 (3) (2011) 1103–1133.
618 URL <http://www.p4est.org>

- 619 [7] P. Jolivet, F. Hecht, F. Nataf, C. Prud'homme, Scalable domain decomposition preconditioners for heterogeneous
620 elliptic problems, in: Proceedings of the International Conference on High Performance Computing, Networking,
621 Storage and Analysis, SC13, ACM, 2013.
- 622 [8] P. Jolivet, P.-H. Tournier, Block iterative methods and recycling for improved scalability of linear solvers, in:
623 Proceedings of the 2016 International Conference for High Performance Computing, Networking, Storage and
624 Analysis, SC16, IEEE, 2016.
- 625 [9] T. Hoefer, R. Belli, Scientific benchmarking of parallel computing systems: Twelve ways to tell the masses
626 when reporting performance results, in: Proceedings of the 2015 International Conference for High Performance
627 Computing, Networking, Storage and Analysis, SC15, 2015.
- 628 [10] V. Hernandez, J. E. Roman, V. Vidal, [SLEPc: A scalable and flexible toolkit for the solution of eigenvalue
629 problems](#), ACM Transactions on Mathematical Software 31 (3) (2005) 351–362.
630 URL <https://slepc.upv.es>
- 631 [11] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine,
632 H. Van der Vorst, Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, SIAM,
633 1994.
- 634 [12] Y. Saad, Iterative Methods for Sparse Linear Systems, 2nd Edition, SIAM, 2003.
- 635 [13] Y. Saad, M. H. Schultz, GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear
636 systems, SIAM Journal on Scientific and Statistical Computing 7 (3) (1986) 856–869.
- 637 [14] M. R. Hestenes, E. Stiefel, Methods of conjugate gradients for solving linear systems, Journal of Research of
638 the National Bureau of Standards 49 (6) (1952) 409–436.
- 639 [15] H. A. Van der Vorst, Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of non-
640 symmetric linear systems, SIAM Journal on Scientific and Statistical Computing 13 (2) (1992) 631–644.
- 641 [16] A. H. Baker, E. R. Jessup, T. Manteuffel, A technique for accelerating the convergence of restarted GMRES,
642 SIAM Journal on Matrix Analysis and Applications 26 (4) (2005) 962–984.
- 643 [17] J. Erhel, K. Burrage, B. Pohl, Restarted GMRES preconditioned by deflation, Journal of Computational and
644 Applied Mathematics 69 (2) (1996) 303–318.
- 645 [18] D. N. Wakam, F. Pacull, Memory efficient hybrid algebraic solvers for linear systems arising from compressible
646 flows, Computers & Fluids 80 (2013) 158–167.
- 647 [19] Y. Saad, A flexible inner-outer preconditioned GMRES algorithm, SIAM Journal on Scientific Computing 14 (2)
648 (1993) 461–469.
- 649 [20] S. C. Eisenstat, H. C. Elman, M. H. Schultz, Variational iterative methods for nonsymmetric systems of linear
650 equations, SIAM Journal on Numerical Analysis 20 (2) (1983) 345–357.
- 651 [21] M. L. Parks, E. de Sturler, G. Mackey, D. D. Johnson, S. Maiti, Recycling Krylov subspaces for sequences of
652 linear systems, SIAM Journal on Scientific Computing 28 (5) (2006) 1651–1674.
- 653 [22] M. H. Gutknecht, Block Krylov space methods for linear systems with multiple right-hand sides: An introduc-
654 tion, in: A. Siddiqui, I. Duff, O. Christensen (Eds.), Modern Mathematical Models, Methods and Algorithms
655 for Real World Systems, 2006, pp. 420–447.
- 656 [23] A. Frommer, K. Lund, D. B. Szyld, Block Krylov subspace methods for functions of matrices, Electronic
657 Transactions on Numerical Analysis 47 (2017) 100–126.
- 658 [24] P.-H. Tournier, I. Aliferis, M. Bonazzoli, M. de Buhan, M. Darbas, V. Dolean, F. Hecht, P. Jolivet, I. El Kanfoud,
659 C. Migliaccio, F. Nataf, C. Pichot, S. Semenov, Microwave tomographic imaging of cerebrovascular accidents
660 by using high-performance computing, Parallel Computing 85 (2019) 88–97.
- 661 [25] V. Kalantzis, A. C. I. Malossi, C. Bekas, A. Curioni, E. Gallopoulos, Y. Saad, A scalable iterative dense linear
662 system solver for multiple right-hand sides in data analytics, Parallel Computing 74 (2018) 136–153.
- 663 [26] A. V. Knyazev, Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate
664 gradient method, SIAM Journal on Scientific Computing 23 (2) (2001) 517–541.
- 665 [27] H. Calandra, S. Gratton, J. Langou, X. Pinel, X. Vasseur, Flexible variants of block restarted GMRES methods
666 with application to geophysics, SIAM Journal on Scientific Computing 34 (2) (2012) A714–A736.
- 667 [28] T. Sakurai, H. Tadano, Y. Kuramashi, Application of block Krylov subspace algorithms to the Wilson–Dirac
668 equation with multiple right-hand sides in lattice QCD, Computer Physics Communications 181 (1) (2010)
669 113–117.
- 670 [29] M. A. Heroux, R. A. Bartlett, V. E. Howle, R. J. Hoekstra, J. J. Hu, T. G. Kolda, R. B. Lehoucq, K. R. Long,
671 R. P. Pawlowski, E. T. Phipps, et al., [An overview of the Trilinos project](#), ACM Transactions on Mathematical
672 Software (TOMS) 31 (3) (2005) 397–423.
673 URL <https://trilinos.github.io>
- 674 [30] E. Bavier, M. Hoemmen, S. Rajamanickam, H. Thornquist, Amesos2 and Belos: Direct and iterative solvers for

- 675 large sparse linear systems, *Scientific Programming* 20 (3) (2012) 241–255.
- 676 [31] Y. Notay, Flexible conjugate gradients, *SIAM Journal on Scientific Computing* 22 (4) (2000) 1444–1460.
- 677 [32] L. M. Carvalho, S. Gratton, R. Lago, X. Vasseur, A flexible generalized conjugate residual method with inner
678 orthogonalization and deflated restarting, *SIAM Journal on Matrix Analysis and Applications* 32 (4) (2011)
679 1212–1235.
- 680 [33] D. P. O’Leary, The block conjugate gradient algorithm and related methods, *Linear Algebra and its Applications*
681 29 (1980) 293–322.
- 682 [34] H. Ji, Y. Li, A breakdown-free block conjugate gradient method, *BIT Numerical Mathematics* 57 (2) (2017)
683 379–403.
- 684 [35] Y. Saad, M. Yeung, J. Erhel, F. Guyomarc’h, A deflated version of the conjugate gradient algorithm, *SIAM*
685 *Journal on Scientific Computing* 21 (5) (2000) 1909–1926.
- 686 [36] A. Stathopoulos, A. Abdel-Rehim, K. Orginos, Deflation for inversion with multiple right-hand sides in QCD,
687 in: *Journal of Physics: Conference Series*, Vol. 180, IOP Publishing, 2009.
- 688 [37] K. M. Soodhalter, D. B. Szyld, F. Xue, Krylov subspace recycling for sequences of shifted linear systems, *Applied*
689 *Numerical Mathematics* 81 (2014) 105–118.
- 690 [38] P. Amestoy, I. Duff, J.-Y. L’Excellent, J. Koster, [A fully asynchronous multifrontal solver using distributed
691 dynamic scheduling](#), *SIAM Journal on Matrix Analysis and Applications* 23 (1) (2001) 15–41.
692 URL <http://mumps.enseeiht.fr>
- 693 [39] A. Stathopoulos, K. Wu, A block orthogonalization procedure with constant synchronization requirements,
694 *SIAM Journal on Scientific Computing* 23 (6) (2002) 2165–2182.
- 695 [40] M. H. Gutknecht, T. Schmelzer, Updating the QR decomposition of block tridiagonal and block Hessenberg
696 matrices, *Applied Numerical Mathematics* 58 (6) (2008) 871–883.
- 697 [41] NVIDIA, cuSPARSE web page, <https://docs.nvidia.com/cuda/cuspars> (2020).
- 698 [42] W. Boukaram, G. Turkiyyah, D. Keyes, Hierarchical matrix operations on GPUs: Matrix–vector multiplication
699 and compression, *ACM Transactions on Mathematical Software* 45 (2019).
- 700 [43] I. Ambartsumyan, W. Boukaram, T. Bui-Thanh, O. Ghattas, D. Keyes, G. Stadler, G. Turkiyyah, S. Zampini,
701 Hierarchical matrix approximations of Hessians arising in inverse problems governed by PDEs, *SIAM Journal*
702 *on Scientific Computing* 42 (5) (2020) A3397–A3426.
- 703 [44] T. A. Davis, Algorithm 832: UMFPACK—an unsymmetric–pattern multifrontal method, *ACM Transactions on*
704 *Mathematical Software* 30 (2) (2004) 196–199.
- 705 [45] Intel, MKL web page, [https://software.intel.com/content/www/us/en/develop/tools/
706 math-kernel-library.html](https://software.intel.com/content/www/us/en/develop/tools/math-kernel-library.html) (2020).
- 707 [46] X. S. Li, An overview of SuperLU: Algorithms, implementation, and user interface, *ACM Transactions on*
708 *Mathematical Software* 31 (3) (2005) 302–325.
- 709 [47] X. S. Li, J. Demmel, SuperLU_DIST: A scalable distributed-memory sparse direct solver for unsymmetric linear
710 systems, *ACM Transactions on Mathematical Software* 29 (2) (2003) 110–140.
- 711 [48] J. Poulson, B. Marker, R. A. Van de Geijn, J. R. Hammond, N. A. Romero, Elemental: A new framework for
712 distributed memory dense matrix computations, *ACM Transactions on Mathematical Software* 39 (2) (2013).
- 713 [49] A. Heinecke, G. Henry, M. Hutchinson, H. Pabst, [LIBXSMM: Accelerating small matrix multiplications by
714 runtime code generation](#), in: *Proceedings of the 2016 International Conference for High Performance Computing,*
715 *Networking, Storage and Analysis, SC16, IEEE, 2016.*
716 URL <https://github.com/hfp/libxsmm>
- 717 [50] F. Hecht, [New development in FreeFem++](#), *Journal of Numerical Mathematics* 20 (3-4) (2012) 251–266.
718 URL <http://freefem.org>
- 719 [51] J. Moulin, P. Jolivet, O. Marquet, [Augmented Lagrangian preconditioner for large-scale hydrodynamic stability
720 analysis](#), *Computer Methods in Applied Mechanics and Engineering* 351 (2019) 718–743.
721 URL <https://github.com/prj-moulin2019a1>
- 722 [52] G. W. Stewart, A Krylov–Schur algorithm for large eigenproblems, *SIAM Journal on Matrix Analysis and*
723 *Applications* 23 (3) (2002) 601–614.
- 724 [53] M. Benzi, M. A. Olshanskii, Z. Wang, Modified augmented Lagrangian preconditioners for the incompressible
725 Navier–Stokes equations, *International Journal for Numerical Methods in Fluids* 66 (4) (2011) 486–508.
- 726 [54] T. Sakurai, H. Sugiura, A projection method for generalized eigenvalue problems using numerical integration,
727 *Journal of Computational and Applied Mathematics* 159 (1) (2003) 119–128.
- 728 [55] M. Adams, H. H. Bayraktar, T. M. Keaveny, P. Papadopoulos, Ultrascalable implicit finite element analyses in
729 solid mechanics with over a half a billion degrees of freedom, in: *Proceedings of the 2004 ACM/IEEE Conference*
730 *on Supercomputing, SC04, IEEE Computer Society, 2004, pp. 34:1–34:15.*

- 731 [56] B. F. Smith, P. Bjørstad, W. D. Gropp, *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial*
732 *Differential Equations*, Cambridge University Press, 2004.
- 733 [57] V. Dolean, P. Jolivet, F. Nataf, *An Introduction to Domain Decomposition Methods: Algorithms, Theory and*
734 *Parallel Implementation*, SIAM, 2015.
- 735 [58] A. Toselli, O. B. Widlund, *Domain decomposition methods: algorithms and theory*, Vol. 34 of *Series in Com-*
736 *putational Mathematics*, Springer, 2005.
- 737 [59] C. Pechstein, *Finite and Boundary Element Tearing and Interconnecting Solvers for Multiscale Problems*, Vol. 90
738 of *Lecture Notes in Computational Science and Engineering*, Springer, 2012.
- 739 [60] A. Quarteroni, A. Valli, *Domain Decomposition Methods for Partial Differential Equations*, Vol. 10, Clarendon
740 Press, 1999.
- 741 [61] T. Mathew, *Domain Decomposition Methods for the Numerical Solution of Partial Differential Equations*,
742 Vol. 61, Springer Science & Business Media, 2008.
- 743 [62] X.-C. Cai, M. Sarkis, A restricted additive Schwarz preconditioner for general sparse linear systems, *SIAM*
744 *Journal on Scientific Computing* 21 (2) (1999) 792–797.
- 745 [63] J. Mandel, Balancing domain decomposition, *Communications in Numerical Methods in Engineering* 9 (3)
746 (1993) 233–241.
- 747 [64] S. Zampini, [PCBDDC: A class of robust dual–primal methods in PETSc](https://www.mcs.anl.gov/petsc/petsc-master/docs/manualpages/PC/PCBDDC.html), *SIAM Journal on Scientific Computing*
748 38 (5) (2016) S282–S306.
749 URL <https://www.mcs.anl.gov/petsc/petsc-master/docs/manualpages/PC/PCBDDC.html>
- 750 [65] C. Pechstein, C. R. Dohrmann, A unified framework for adaptive BDDC, *Electronic Transactions on Numerical*
751 *Analysis* 46 (2017) 273–336.
- 752 [66] S. Zampini, P. Vassilevski, V. Dobrev, T. Kolev, Balancing domain decomposition by constraints algorithms
753 for curl-conforming spaces of arbitrary order, in: *International Conference on Domain Decomposition Methods*,
754 Springer, 2017, pp. 103–116.
- 755 [67] D.-S. Oh, O. B. Widlund, S. Zampini, C. Dohrmann, BDDC algorithms with deluxe scaling and adaptive
756 selection of primal constraints for Raviart–Thomas vector fields, *Mathematics of Computation* 87 (310) (2018)
757 659–692.
- 758 [68] L. B. Da Veiga, L. F. Pavarino, S. Scacchi, O. B. Widlund, S. Zampini, Isogeometric BDDC preconditioners
759 with deluxe scaling, *SIAM Journal on Scientific Computing* 36 (3) (2014) A1118–A1139.
- 760 [69] M. W. Gee, C. M. Siefert, J. J. Hu, R. S. Tuminaro, M. G. Sala, [ML 5.0 smoothed aggregation user’s guide](https://trilinos.github.io/ml.html),
761 Tech. Rep. SAND2006-2649, Sandia National Laboratories (2006).
762 URL <https://trilinos.github.io/ml.html>
- 763 [70] A. Heinlein, A. Klawonn, O. Rheinbach, A parallel implementation of a two-level overlapping Schwarz method
764 with energy-minimizing coarse space based on Trilinos, *SIAM Journal on Scientific Computing* 38 (6) (2016)
765 C713–C747.
- 766 [71] J. Šítek, J. Mandel, B. Sousedík, P. Burda, [Parallel implementation of multilevel BDDC](http://users.math.cas.cz/~sistek/software/bddcml.html), in: *Numerical Math-*
767 *ematics and Advanced Applications 2011*, Springer, 2013, pp. 681–689.
768 URL <http://users.math.cas.cz/~sistek/software/bddcml.html>
- 769 [72] S. Badia, A. F. Martín, J. Principe, A highly scalable parallel implementation of balancing domain decomposition
770 by constraints, *SIAM Journal on Scientific Computing* 36 (2) (2014) C190–C218.
- 771 [73] M. J. Gander, Optimized Schwarz Methods, *SIAM Journal on Numerical Analysis* 44 (2) (2006) 699–731.
- 772 [74] M. J. Gander, S. Van Criekingen, New coarse corrections for optimized restricted additive Schwarz using PETSc,
773 in: *International Conference on Domain Decomposition Methods*, Springer, 2019.
- 774 [75] N. Spillane, V. Dolean, P. Hauret, F. Nataf, C. Pechstein, R. Scheichl, A robust two-level domain decomposition
775 preconditioner for systems of PDEs, *Comptes Rendus Mathématique* 349 (23) (2011) 1255–1259.
- 776 [76] N. Spillane, V. Dolean, P. Hauret, F. Nataf, C. Pechstein, R. Scheichl, Abstract robust coarse spaces for systems
777 of PDEs via generalized eigenproblems in the overlaps, *Numerische Mathematik* 126 (4) (2013) 741–700.
- 778 [77] P. Marchand, X. Claeys, P. Jolivet, F. Nataf, P.-H. Tournier, Two-level preconditioning for h -version boundary
779 element approximation of hypersingular operator with GenEO, *Numerische Mathematik* 146 (2020) 597–628.
- 780 [78] J. Brown, M. G. Knepley, D. A. May, L. Curfman McInnes, B. F. Smith, Composable linear solvers for mul-
781 tiphysics, in: *2012 11th International Symposium on Parallel and Distributed Computing, IEEE*, 2012, pp.
782 55–62.
- 783 [79] R. Butler, T. Dodwell, A. Reinarz, A. Sandhu, R. Scheichl, L. Seelinger, High-performance DUNE modules for
784 solving large-scale, strongly anisotropic elliptic problems with applications to aerospace composites, *Computer*
785 *Physics Communications* 249 (2020).
- 786 [80] D. A. May, P. Sanan, K. Rupp, M. G. Knepley, B. F. Smith, Extreme-scale multigrid components within PETSc,

- 787 in: Proceedings of the Platform for Advanced Scientific Computing Conference, 2016.
- 788 [81] M. G. Knepley, D. A. Karpeev, Mesh algorithms for PDE with Sieve I: Mesh distribution, *Scientific Programming*
789 17 (3) (2009) 215–230.
- 790 [82] J. Tang, R. Nabben, C. Vuik, Y. Erlangga, Comparison of two-level preconditioners derived from deflation,
791 domain decomposition and multigrid methods, *Journal of Scientific Computing* 39 (3) (2009) 340–370.
- 792 [83] P. Bastian, G. Wittum, W. Hackbusch, Additive and multiplicative multi-grid—a comparison, *Computing* 60 (4)
793 (1998) 345–364.
- 794 [84] R. Haferssas, P. Jolivet, F. Nataf, An additive Schwarz method type theory for Lions’s algorithm and a sym-
795 metrized optimized restricted additive Schwarz method, *Journal on Scientific Computing* 39 (4) (2017) A1345–
796 A1365.
- 797 [85] L. Conen, V. Dolean, R. Krause, F. Nataf, A coarse space for heterogeneous Helmholtz problems based on the
798 Dirichlet-to-Neumann operator, *Journal of Computational and Applied Mathematics* 271 (2014) 83–99.
- 799 [86] H. Al Daas, L. Grigori, P. Jolivet, P.-H. Tournier, [A multilevel Schwarz preconditioner based on a hierarchy of](#)
800 [robust coarse spaces](#), *Journal of Scientific Computing* (2019) submitted for publication.
801 URL <https://github.com/prj-/aldaas2019multi>
- 802 [87] N. Spillane, D. J. Rixen, Automatic spectral coarse spaces for robust finite element tearing and interconnecting
803 and balanced domain decomposition algorithms, *International Journal for Numerical Methods in Engineering*
804 95 (11) (2013) 953–990.
- 805 [88] G. Karypis, V. Kumar, [A fast and high quality multilevel scheme for partitioning irregular graphs](#), *SIAM Journal*
806 *on Scientific Computing* 20 (1) (1998) 359–392.
807 URL <http://glaros.dtc.umn.edu/gkhome/metis/parmetis/overview>
- 808 [89] R. Anderson, J. Andrej, A. Barker, J. Bramwell, J.-S. Camier, J. Cerveny, V. Dobrev, Y. Dudouit, A. Fisher,
809 T. Kolev, W. Pazner, M. Stowell, V. Tomov, I. Akkerman, J. Dahm, D. Medina, S. Zampini, [MFEM: A modular](#)
810 [finite element methods library](#), *Computers & Mathematics with Applications* 81 (2021) 42–74.
811 URL <http://mfem.org>
- 812 [90] Y. Zhou, Y. Saad, Block Krylov–Schur method for large symmetric eigenvalue problems, *Numerical Algorithms*
813 47 (4) (2008) 341–359.